A UNIFIED OPTIMIZATION FRAMEWORK OF ANN-SNN CONVERSION: TOWARDS OPTIMAL MAPPING FROM ACTIVATION VALUES TO FIRING RATES

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking Neural Networks (SNNs) have attracted great attention as a primary candidate for running large-scale deep artificial neural networks (ANNs) in real-time due to their distinctive properties of energy-efficient and event-driven fast computation. Training an SNN directly from scratch is usually difficult because of the discreteness of spikes. Converting an ANN to an SNN, i.e., ANN-SNN conversion, is an alternative method to obtain deep SNNs. The performance of the converted SNN is determined by both the ANN performance and the conversion error. The existing ANN-SNN conversion methods usually redesign the ANN with a new activation function instead of the regular ReLU, train the tailored ANN and convert it to an SNN. The performance loss between the regular ANN with ReLU and the tailored ANN has never been considered, which will be inherited to the converted SNN. In this work, we formulate the ANN-SNN conversion as a unified optimization problem which considers the performance loss between the regular ANN and the tailored ANN, as well as the conversion error simultaneously. Following the unified optimization framework, we propose the SlipReLU activation function to replace the regular ReLU activation function in the tailored ANN. The SlipReLU is a weighted sum of the threhold-ReLU and the step function, which improves the performance of either as an activation function alone. The SlipReLU method covers a family of activation functions mapping from activation values in source ANNs to firing rates in target SNNs; most of the state-of-the-art optimal ANN-SNN conversion methods are special cases of our proposed SlipReLU method. We demonstrate through two theorems that the expected conversion error between SNNs and ANNs can theoretically be zero on a range of shift values $\delta \in [-\frac{1}{2}, \frac{1}{2}]$ rather than a fixed shift term $\frac{1}{2}$, enabling us to achieve converted SNNs with high accuracy and ultra-low latency. We evaluate our proposed SlipReLU method on CIFAR-10 dataset, and the results show that the SlipReLU outperforms the state-of-the-art ANN-SNN conversion in both accuracy and latency. To our knowledge, this is the first work to explore high-performance ANN-SNN conversion method considering the ANN performance and the conversion error simultaneously.

1 INTRODUCTION

Spiking neural networks (SNNs) are biologically-inspired neural networks based on biological plausible spiking neuron models to process real-time signals (Hodgkin & Huxley, 1952; Izhikevich, 2003). With the significant advantages of low power consumption and fast inference on neuromorphic hardware (Roy et al., 2019), SNNs are therefore becoming a primary candidate to run large-scale deep artificial neural networks (ANNs) in real-time. The most commonly used neuron model in SNNs is the Integrate-and-Fire (IF) neuron model (Liu & Wang, 2001). Each neuron in the SNNs emits a spike only when its accumulated membrane potential exceeds the threshold voltage, otherwise, it stays inactive in the current time-step. This setting makes SNNs more similar to biological neural networks. Compared to ANNs, event-driven SNNs have binarized/spiking activation values, resulting in low energy consumption when implemented on specialized neuromorphic hardware. Another significant property of SNNs is the pseudo-simultaneity of their inputs and outputs

Table 1: Summary of notations in this paper.

Symbol	Definition	Symbol	Definition
N	Quasi-time-steps of ANNs	$\mathcal{F}_{\text{ANN}}(\cdot)$	ANN activation function
T	Total time-steps of SNNs	$\mathcal{F}_{\text{SNN}}(\cdot)$	SNN activation function
$\boldsymbol{a}^{(\ell)}$	Activation values of ANNs	$\mathbf{s}^{(\ell)}(t)$	Spike outputs of SNN
$\bar{\mathbf{x}}^{(\ell)}$	Average PSP of SNNs	$\mathbf{x}^{(\ell)}(t)$	PSP released by l-th layer
$\theta^{(\ell)}$	Trainable threshold in ANNs	$\mathbf{v}^{(\ell)}(t)$	Membrane potential after firing
$V_{th}^{(\ell)}$	Firing threshold in SNNs	$\mathbf{W}^{(\ell)}$	Weight matrix

for making inferences in a spatial-temporal paradigm. Compared to conventional ANNs that present a whole input vector at once, and process layer-by-layer to produce one output value, the forwarding pass in SNN can efficiently process streaming time-varying inputs.

Generally, there are two distinct routes to obtain an SNN: (1) training an SNN from scratch (Wu et al., 2018; Neftci et al., 2019; Zenke & Vogels, 2021), and (2) ANN-SNN conversion (i.e., converting ANNs to SNNs) (Cao et al., 2015; Diehl et al., 2015; Deng & Gu, 2021). Training from scratch uses a gradient-based supervised optimization method in back-propagation, pretending that SNNs are specialized ANNs. Due to the non-differentiability of the binary activation function in SNNs, surrogate gradients are usually used (Neftci et al., 2019), but it essentially optimizes different networks in forward and backward passes. This method can only train SNNs on small- and moderate-size datasets (Li et al., 2021). ANN-SNN conversion is an effective method to obtain deep SNNs, with comparable performance as ANNs on large-scale datasets. There are two main types of ANN-SNN conversion mechanism: (1) one-step conversion, which converts the pre-trained ANN to SNN without changing the architecture of the pre-trained ANN, for example (Diehl et al., 2015; Li et al., 2021), and (2) two-step conversion, which involves redesigning the ANN, training it and converting it to SNN, for example Cao et al. (2015); Deng & Gu (2021); Bu et al. (2021).

In this work, we investigate the two-step ANN-SNN conversion methods, where we usually redesign the ANN by replacing the regular ReLU activation function to a new activation function, train the tailored ANN and convert it to an SNN. A tailored ANN that deviates too much from the regular ANN will degrade its performance, resulting in a performance loss which will be inherited to the converted SNN. However, the performance degradation between the regular ANN and the tailored ANN has never been considered in the existing ANN-SNN conversion studies. To achieve high-accuracy and low-latency SNNs (e.g., 2 time-steps), we are the first to consider the performance loss between the regular ANN with ReLU and the tailored ANN, as well as the conversion error simultaneously. Our main contributions are summarized as follows:
(1) We formulate the ANN-SNN conversion as a unified optimization problem which considers the ANN performance as well as the conversion error simultaneously.
(2) We propose to use the SlipReLU activation function in the tailored ANN, in order to minimize the layer-wise conversion error and keep tailored ANN performance as good as the regular ANN.
(3) The SlipReLU method covers a family of activation functions mapping from activation values in source ANNs to firing rates in target SNNs; most of the state-of-the-art optimal ANN-SNN conversion methods are special cases of our proposed SlipReLU method.
(4) We demonstrate through two theorems that the expected conversion error between SNNs and ANNs can theoretically be zero on a range of shift values $\delta \in [-\frac{1}{2}, \frac{1}{2}]$ rather than a fixed shift $\frac{1}{2}$. Experiment results also demonstrate the effectiveness of the proposed SlipReLU method.

2 PRELIMINARIES

Given a classification problem on an image dataset $(\boldsymbol{x}, y) \in \mathcal{D}$, where $y \in \{1, \cdots, C\}$ is the true class label for image $\boldsymbol{x} \in \mathbb{R}^m$, we train a neural network $f : \boldsymbol{x} \to f(\boldsymbol{x})$ in the form of an ANN/SNN, by optimizing the standard cross-entropy (CE) loss, $L_{\text{CE}}(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{j=1}^{C} \boldsymbol{y}_c \log(\boldsymbol{p}_c)$, where $\boldsymbol{y}_c$ and $\boldsymbol{p}_c$ are the c-th elements of the label $\boldsymbol{y}$ and the network prediction $\boldsymbol{p} = f(\boldsymbol{x})$. Since the infrastructures of the source ANN and target SNN are the same, we use the same f notation when it is unambiguous. And f_{ANN} or f_{SNN}, otherwise. For the notations, refer to Table 1.

ANN Neuron Model. In conventional ANN, a whole input vector is presented to the network at one time, and processed layer-by-layer through continuous activation to produce one output value. In ANNs, the forwarding computation of analog neurons is formulated as

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\mathrm{ANN}}(\mathbf{z}^{(\ell)}) = \mathcal{F}_{\mathrm{ANN}}(\mathbf{W}^{(\ell)}\boldsymbol{a}^{(\ell-1)}) \ , \tag{1}$$

where $\mathbf{z}^{(\ell)}$ and $\boldsymbol{a}^{(\ell)}$ are the pre-activation and post-activation vectors of the ℓ-th layer considered, $\mathbf{W}^{(\ell)}$ denotes the weight matrix, and $\mathcal{F}_{\mathrm{ANN}}(\cdot)$ is the activation function of the ANN.

SNN Neuron Model. Compared with ANN, SNN employs binary activation (i.e. spikes) in each layer. To compensate the weak representation capacity of the binary activation, the time dimension (or the latency) is introduced to SNN, where the inputs of the forwarding pass in SNN are presented as streams of events, by repeating the forwarding pass T time-steps to get the final result.

Here we consider the Integrate-and-Fire (IF) neuron model (Cao et al., 2015; Bu et al., 2021; Deng & Gu, 2021) for SNNs. We derive the forward propagation of PSP through layers in the target SNN which is equivalent to the forwarding computation of the analog neurons in the source ANN. Suppose at time-step t the IF neuron in ℓ-th layer receive its binary input $\mathbf{x}^{(\ell-1)}(t)$ from the previous layer, the IF neuron will temporarily update its membrane potential by

$$\mathbf{u}^{(\ell)}(t) = \mathbf{v}^{(\ell)}(t-1) + \mathbf{W}^{(\ell)}\mathbf{x}^{(\ell-1)}(t) \ , \tag{2}$$

where $\mathbf{v}^{(\ell)}(t)$ denotes the membrane potential at time step t, $\mathbf{u}^{(\ell)}(t)$ denotes the temporary intermediate variable that would be used to determine the update from $\mathbf{v}^{(\ell)}(t-1)$ to $\mathbf{v}^{(\ell)}(t)$. For each IF neuron, if the temporary intermediate potential $u_i^{(\ell)}(t)$ exceeds the membrane threshold $V_{\mathrm{th}}^{(\ell)}$, it would produce a spike output $s_i^{(\ell)}(t)$. Otherwise, it would release no spikes $s_i^{(\ell)}(t) = 0$.

$$s_i^{(\ell)}(t) = H(u_i^{(\ell)}(t) - V_{\mathrm{th}}^{(\ell)}) = \begin{cases} 1, & \text{if } u_i^{(\ell)}(t) \geqslant V_{\mathrm{th}}^{(\ell)}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The vector $\mathbf{s}^{(\ell)}(t) = \{s_i^{(\ell)}(t)\}$ collects spikes of all neurons of ℓ-th layer at time t. Note that $V_{\mathrm{th}}^{(\ell)}$ can be different in each layer. We update the membrane potential by the reset-by-subtraction mechanism (Rueckauer et al., 2017; Han et al., 2020), which means the temporary membrane potential $u_i^{(\ell)}(t)$ is subtracted by the threshold value $V_{\mathrm{th}}^{(\ell)}$ if the neuron fires, $s_i^{(\ell)}(t) = 1$. That is

$$\mathbf{v}^{(\ell)}(t) = \mathbf{u}^{(\ell)}(t) - \mathbf{s}^{(\ell)}(t)V_{\mathrm{th}}^{(\ell)} \ . \tag{4}$$

Similar to Deng & Gu (2021); Bu et al. (2021), if the neuron in the current ℓ layer fires a spike, then it will release an unweighted PSP (postsynaptic potential) $\mathbf{x}^{(\ell)}(t)$ as input to the next layer,

$$\mathbf{x}^{(\ell)}(t) = \mathbf{s}^{(\ell)}(t)V_{\mathrm{th}}^{(\ell)} \ .$$

As for the input to the first layer and the output of the last layer, we do not employ any spiking mechanism. We use the first layer to direct encode the static image to temporal dynamic spikes, this can prevent the undesired information loss of the Poission encoding. For the last layer output, we only integrate the pre-synaptic input and does not firing any spikes.

3 Unified Optimization Framework of ANN-SNN Conversion

In this section, we formulate the ANN-SNN conversion problem as an optimization problem determined by two terms: one is used to make the tailored ANN not far away from the regular ANN with ReLU, the other one is used to control the ANN-SNN conversion error.

Starting from the first ANN-SNN conversion work (Cao et al., 2015), all the previous ANN-SNN conversion methods (Cao et al., 2015; Diehl et al., 2015; Deng & Gu, 2021) have focused on optimizing the ANN-SNN conversion error. The performance of the converted SNN is determined by both the ANN performance and the conversion error. However, no research work has considered the ANN performance in the two-step ANN-SNN conversions, which is considered in our unified optimization framework. In the two-step conversion method, we need to redesign the new activation function of the regular ANN to get a tailored ANN, train the tailored ANN and convert it to SNN. By considering the performance loss between the tailored ANN and the regular ANN, the new activation function should not deviate too much from the regular ReLU.

3.1 ANN-SNN CONVERSION IN A UNIFIED OPTIMIZATION FRAMEWORK

Definition 1 (Unified Optimization Framework of ANN-SNN Conversion)**.** *The ANN-SNN conversion can be formulated into a unified optimization framework with an implicit variable, T,*

$$\min_{\mathcal{F},T} \quad \{w\mathbb{E}_{\mathbf{z}}\left(|f(\mathbf{z};\mathbf{W},\mathrm{ReLU}) - f(\mathbf{z};\mathbf{W},T,\mathcal{F}_{\mathrm{ANN}})|\right) + (1-w)\mathbb{E}_{\mathbf{z}}\left(|f(\mathbf{z};\mathbf{W},\mathcal{F}_{\mathrm{ANN}}) - f(\mathbf{z};\mathbf{W},T,\mathcal{F}_{\mathrm{SNN}})|\right)\} \ . \tag{5}$$

where $w \in [0,1]$. Specially, if $\mathcal{F}_{\mathrm{ANN}}$ is designed by considering the deviation from the regular ReLU, the layer-wise conversion error becomes

$$\mathbb{E}_{\mathbf{z}}\left(\left|\mathrm{Err}^{(\ell)}\right|\right) = \mathbb{E}_{\mathbf{z}}\left(\left|\mathcal{F}_{\mathrm{ANN}}(\boldsymbol{a}^{(\ell-1)};\mathbf{W}^{(\ell)}) - \mathcal{F}_{\mathrm{SNN}}(\bar{\mathbf{x}}^{(\ell-1)};\mathbf{W}^{(\ell)},T)\right|\right) \ . \tag{6}$$

Here f denotes the same neural network infrastructures shared by the source ANN and target SNN (see description in Sect. 2), $f(\mathbf{z};\mathbf{W},\mathrm{ReLU})$ denotes the regular ANN with ReLU, $f(\mathbf{z};\mathbf{W},T,\mathcal{F}_{\mathrm{ANN}})$ is the tailored ANN with activation function $\mathcal{F}_{\mathrm{ANN}}$, $f(\mathbf{z};\mathbf{W},T,\mathcal{F}_{\mathrm{SNN}})$ is the converted SNN, $\mathbf{z}$ is the input to the neural network, $\mathbf{W} = \{\mathbf{W}^{(\ell)}\}$ are the weight matrix trained from the tailored ANN and copied to the target SNN, $\mathcal{F} = \mathcal{F}_{\mathrm{ANN}} \cup \mathcal{F}_{\mathrm{SNN}}$ is the space of activation functions of the tailored ANNs and the target SNNs, and the latency T (or time-steps) is seen as an implicit variable inherently inherited from the target SNNs. Moreover, the latency also allows the flexibility of adjusting T to balance between the latency and the accuracy of the converted SNN for different applications.

Remark 1. *(A) An effective activation function $\mathcal{F}_{\mathrm{ANN}}$ of the tailored ANN should address the performance lose caused by the deviation from the regular ReLU. (B) When $\mathcal{F}_{\mathrm{ANN}}$ is designed by considering the deviation from the regular ReLU, the layer-wise error Eq. (6) would come from any mismatch of the three parts: (1) different activation values from source ANNs and target SNNs, i.e. $\boldsymbol{a}^{(\ell)}$ and $\bar{\mathbf{x}}^{(\ell)}$, (2) different activation functions, i.e. $\mathcal{F}_{\mathrm{ANN}}(\cdot)$ and $\mathcal{F}_{\mathrm{SNN}}(\cdot)$, and (3) the latency variable T which implicitly affects both the activation values and activation functions. (C) Whenever the conversion error $\mathbb{E}_{\mathbf{z}}(|\mathrm{Err}^{(\ell)}|)$ achieves its minimum, it is called an "optimal" ANN-SNN conversion. For example, Deng & Gu (2021) achieves optimal minimum error of $\frac{(V_{\mathrm{th}}^{(\ell)})^2}{4T}$, whereas Bu et al. (2021) can theoretically achieve optimal minimum error of* 0.

3.2 ANN-SNN CONVERSION ERROR ANALYSIS

3.2.1 FIRING RATES IN SNNS AND ACTIVATION VALUES IN ANNS

To make the layer-wise error as small as possible, ideally, the converted SNN is expected to have approximately the same activation values as the source ANN for each layer, i.e.,

$$\boldsymbol{a}^{(\ell)} \approx \bar{\mathbf{x}}^{(\ell)} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{x}^{(\ell)}(t) = \frac{1}{T}\sum_{t=1}^{T}\mathbf{s}^{(\ell)}(t)V_{\mathrm{th}}^{(\ell)} = V_{\mathrm{th}}^{(\ell)}\bar{\mathbf{s}}^{(\ell)} \ . \tag{7}$$

Here $\boldsymbol{a}^{(\ell)}$ denotes activation value of the ANN, and $\bar{\mathbf{x}}^{(\ell)}$ is activation value of the SNN which is actually the average postsynaptic potential (i.e. average PSP) released by the ℓ-th layer as input to the next layer. Note $\bar{\mathbf{s}}^{(\ell)}$ is the firing rate over latency T of ℓ-th layer. Note that the thresholding $V_{\mathrm{th}}^{(\ell)}$ in SNN can be different from layer to layer, we make it a trainable parameter that can be learned in the source ANN and copied to the target SNN. Any mismatch between the activation values $\boldsymbol{a}^{(\ell)}$ and $\bar{\mathbf{x}}^{(\ell)}$ can lead to conversion error.

3.2.2 ACTIVATION FUNCTION IN SNNS

We use the derivation in Deng & Gu (2021); Li et al. (2021) to deduce the SNN activation function $\mathcal{F}_{\mathrm{SNN}}$ which gives the relationship between activation values $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$ of successive layers of SNN. By combining Eq. (2) and Eq. (4) and summing up the time-step from 1 to T, then we get

$$\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0) = \mathbf{W}^{(\ell)}\sum_{t=1}^{T}\mathbf{x}^{(\ell-1)}(t) - \sum_{t=1}^{T}\mathbf{s}^{(\ell)}(t)V_{\mathrm{th}}^{(\ell)} \ .$$

The accumulated spikes are $\mathbf{m} = \sum_{t=1}^{T} \mathbf{s}^{(\ell)}(t) = \{m_i\}$ where each $m_i \in \{0, 1, 2, \cdots, T\}$ denotes the total number of spikes of neuron i. Further assume $\mathbf{v}^{(\ell)}(T) \in [\mathbf{0}, \mathbf{V}_{\text{th}}^{(\ell)})$. Therefore, we have

$$\frac{T\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} < \mathbf{m} \leqslant \frac{T\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \quad \text{with shift} \quad \boldsymbol{\delta} = \frac{\mathbf{v}^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}} .$$

Then, we use the clip and floor functions to determine $\mathbf{m}$,

$$\mathbf{m} = \text{clip}\left(\left\lfloor \frac{T\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, T\right) .$$

We provide the detailed analysis of the activation function of SNNs in Appendix A. Here the $\text{clip}(x, a, b)$ function sets the lower bound a and upper bound b. Floor function $\lfloor x \rfloor$ gives the greatest integer that is less than or equal to x. With $\bar{\mathbf{x}}^{(\ell)} = V_{\text{th}}^{(\ell)}\bar{\mathbf{s}}^{(\ell)} = \mathbf{m}V_{\text{th}}^{(\ell)}/T$, finally, the SNN activation function gives the relationship between activation values $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$ as follows,

$$\bar{\mathbf{x}}^{(\ell)} = \mathcal{F}_{\text{SNN}}\left(\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)}\right) = V_{\text{th}}^{(\ell)}\text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, 1\right) . \tag{8}$$

The SNN activation function $\mathcal{F}_{\text{SNN}}(\cdot)$ is a step function in interval $[0, V_{\text{th}}^{(\ell)}]$ with a step size $\frac{V_{\text{th}}^{(\ell)}}{T}$ (see the green curve in Fig. 1). Since the SNN output is discrete while the ANN output is continuous, there actually would be an intrinsic difference between $\boldsymbol{a}^{(\ell)}$ and $\bar{\mathbf{x}}^{(\ell)}$ as shown in Fig. 1.

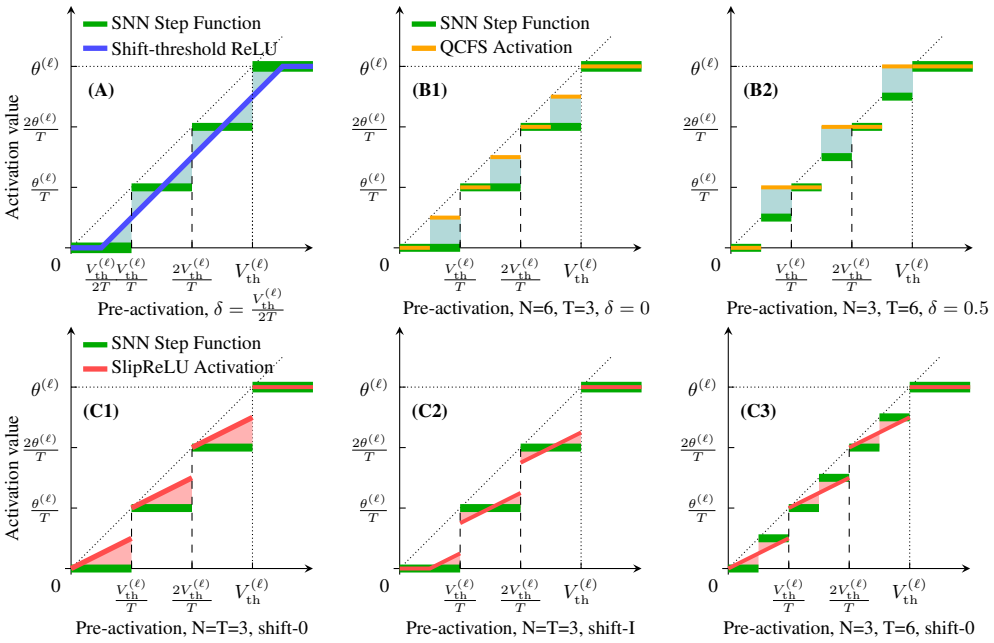

Figure 1: Activation functions of source ANNs, i.e., the shift-threshold-ReLU (blue curve) (Deng & Gu, 2021) in (A), quantization clip-floor-shift (QCFS) activation (orange curve) (Bu et al., 2021) in (B1)-(B2), and our proposed SlipReLU (red curve) in (C1)-(C3). The activation functions of target SNNs is the step function (green curve). The error between the activation function (of ANNs) and the step function (of SNNs) is obtained by summing up of all the shaded area together, which is the ANN-SNN conversion error.

4 Proposed SlipReLU: Optimal ANN-SNN Conversion

4.1 The SlipReLU Activation Function

In this section, by following our unified optimization framework, we will exploit the two-step conversion mechanism. We redesign the ANN with to get a tailored ANN, train the tailored ANN

and convert it to SNN by copying the weights from the tailored ANN to the target SNN. Performance loss will occur if the the new activation function of the tailored ANN deviates too much from the regular ReLU activation function, and we need to minimize the conversion error at the same time. By keeping that in mind, an effective activation function $\mathcal{F}_{\text{ANN}}$ of the tailored ANN should not deviate too much from the regular ReLU and be close to the step function. Therefore, we propose the SlipReLU activation function which is a weighted sum of the threshold-ReLU and the step function to balance the trade-off between the regular ReLU and step function. Assume that both ANN and SNN receive the same input from the previous layer, $\boldsymbol{a}^{(\ell-1)} = \bar{\mathbf{x}}^{(\ell-1)}$. Denote $\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)} = \mathbf{W}^{(\ell)}\boldsymbol{a}^{(\ell-1)}$.

Proposed SlipReLU activation function. Following the unified optimization framework Sect. 3, new activation functions of the tailored ANNs are designed by minimizing the mismatch to the step function of the target SNNs, and minimizing the deviation from the regular ReLU. We propose the SlipReLU activation function for the tailored ANN (see the red curves in (C1)-(C3) of Fig. 1),

$$\mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) . \tag{9}$$

From definition in Eq. (9), the SlipReLu activation function is just a weighted sum of the threshold-ReLU (first part) and the step function (second part), with the slope $0 \leqslant c \leqslant 1$ balancing its weight.

With some linear algebra, the SlipReLU can be formulated as a piece-wise linear function with a constant slope c (see the red curves in (C1)-(C3) of Fig. 1 and detailed derivation in Appendix B), $\text{SlipReLU}(\mathbf{z}^{(\ell)}) = c\mathbf{z}^{(\ell)} + (1-c)\frac{k\boldsymbol{\theta}^{(\ell)}}{N}$, $\frac{k\boldsymbol{\theta}^{(\ell)}}{N} \leqslant \mathbf{z}^{(\ell)} < \frac{(k+1)\boldsymbol{\theta}^{(\ell)}}{N}$, $k = 0, 1, \cdots, N-1$. Here $0 \leqslant c \leqslant 1$ is the constant slope of the piece-wise linear function. From the definition and red curves in (C1)-(C3) of Fig. 1, we see that the new proposed function is very similar to a slippery step function with a slope, hence the name "SlipReLU".

SlipReLU extension The SlipReLU extension with shift (see Appendix B) can be formulated as

$$\mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta}_1, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta}\right\rfloor, 0, 1\right) . \tag{10}$$

An application of the unified optimization framework. Recall the step activation function of target SNNs in Eq. (8), by setting $c = 0$ the SlipReLU becomes the step function,

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = \theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta}\right\rfloor, 0, 1\right) . \tag{11}$$

Note that Eq. (11) is exactly the step function in Eq. (8), except $V_{\text{th}}^{(\ell)} \leftarrow \theta^{(\ell)}, T \leftarrow N$. Because the latency T in Eq. (8) is an inherent property of the target SNNs, so it cannot be used in the source ANN. Therefore, instead of T, we use quasi-time-steps N in SNN. As mentioned in Sect. 3.2.1, the threshold $V_{\text{th}}^{(\ell)}$ in SNN can be different from layer to layer, and we make it a trainable value $\theta^{(\ell)}$ in the ANN which can be learned and copied to the target SNN. Coincidentally, Bu et al. (2021) uses this function defined in Eq. (11) as the activation function in the source ANNs, and they name it the quantization clip-floor-shift (QCFS) activation function and call N the quantization steps.

Special cases of SlipReLU Here we list some related works which fall in to our proposed unified optimization framework and are special cases of the SlipReLU. (1) When $c = 0, \boldsymbol{\delta} = [\frac{1}{2}]$, the proposed SlipRLU activation function becomes the quantization clip-floor-shift in Bu et al. (2021). It only considers to be close to the SNN step function, but neglects the deviation from the regular ReLU. (2) When $c = 1, \boldsymbol{\delta}_1 = [-\frac{1}{2N}]$, the proposed SlipRLU activation function becomes shift-threshold ReLU in Deng & Gu (2021). It considers the deviation from the regular ReLU but neglects the closeness to the step function of the target SNN. Our proposed SlipReLU balances the trade-off between the regular ReLU and the step function. Refer to Appendix B for the details.

4.2 Theorems on the conversion error

The following two theorems gives the conversion error of the proposed unified method.

Theorem 1. *An ANN trained with SlipReLU activation function Eq. (10) is converted to an SNN with the same weights. Let* $V_{\text{th}}^{(\ell)} = \theta^{(\ell)}$, $\mathbf{v}^{(\ell)}(0) = V_{\text{th}}^{(\ell)}\boldsymbol{\delta}$, $c = 0$. *Then for arbitrary* T *and* N, *the expectation of the conversion error of the proposed unified method reaches* 0, *i.e.,*

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\text{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \mathbf{0}\,, \tag{12}$$

holds for any the shift term δ *in the source ANN when* $\delta \in [-\frac{1}{2}, \frac{1}{2}]$.

Theorem 1 indicates that for $c = 0$ the expected conversion error reaches zero even though $N \neq T$ provided that the shift term $\delta \in [-\frac{1}{2}, \frac{1}{2}]$. The proof is in Appendix C.

Theorem 2. *An ANN trained with SlipReLU activation function Eq. (10) is converted to an SNN with the same weights. Let* $V_{\text{th}}^{(\ell)} = \theta^{(\ell)}$, $\mathbf{v}^{(\ell)}(0) = V_{\text{th}}^{(\ell)}\boldsymbol{\delta}$, $\boldsymbol{\delta}_1 = [\frac{\delta - 1/2}{T}]$. *Then for arbitrary* T *and* N *and arbitrary* $c \in [0, 1]$, *the expectation of the conversion error of the proposed unified method reaches the optimal* $\frac{c(V_{\text{th}}^{(\ell)})^2}{4T}$, *i.e.,*

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\text{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \frac{c(V_{\text{th}}^{(\ell)})^2}{4T}\,, \tag{13}$$

holds for any the shift term δ *in the source ANN when* $\delta \in [-\frac{1}{2}, \frac{1}{2}]$.

Theorem 2 indicates that for any $\forall\, c \in [0, 1]$, the expectation of the conversion error can reach the minimum $\frac{c(V_{\text{th}}^{(\ell)})^2}{4T}$, provided that the shift term δ in the source ANN is in the interval $[-\frac{1}{2}, \frac{1}{2}]$, and $\delta_1 = \frac{\delta - 1/2}{T}$. The proof is in Appendix C. These results indicate we can achieve high-performance converted SNN at ultra-low time-steps.

4.3 Algorithm for Training SlipReLU Activation Function in Backpropagation

Training an ANN with SlipReLU activation instead of ReLU is also a challenging problem. Although the SlipReLU has a constant slope as its derivative from Eq. (9), however, small slope values $c \in [0, 1]$ can cause the gradient vanishing problem. Therefore, inspired by Bu et al. (2021); Bengio et al. (2013), we use the surrogate gradient as the derivative of the floor function $\frac{d\lfloor x \rfloor}{x} = 1$. The overall derivation rule is given as follows,

$$\frac{d\mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)})}{d\mathrm{z}_i^{(\ell)}} = \begin{cases} 1, & \text{if } \mathrm{z}_i^{(\ell)} \in D_1 \cup D_2 \\ 0, & \text{otherwise} \end{cases}$$

where $D_1 = [-\delta_1\theta, \theta - \delta_1\theta]$, $D_2 = [-\delta\theta, \theta - \delta\theta]$, and $\mathrm{z}_i^{(\ell)}$ is the i-th element of $\mathbf{z}^{(\ell)}$. Then we can train the ANN with SlipReLU activation using Stochastic Gradient Descent algorithm, and convert it to the SNN. Refer to Appendix D for the pseudo-code of our proposed ANN-SNN conversion algorithm.

5 Related Work

The first study of ANN-SNN conversion is proposed by Cao et al. (2015), which convert the ANNs with the ReLU activation function to SNNs. Afterwards, Diehl et al. (2015) proposed data-based and model-based weight-normalization method to convert a three-layer CNN to an SNN. However, it usually requires hundreds of time-steps for the converted SNN to get accurate results due to the error analyzed in Sect. 3. To address the potential information loss, the "reset-by-subtraction" mechanism (Rueckauer et al., 2017), also called "soft-reset" (Han et al., 2020) rather than "reset-to-zero" is proposed. Recently, many methods and algorithms have been proposed to eliminate the conversion error. Sengupta et al. (2019) proposed a novel weight-normalization technique which considers the actual SNN operation in the conversion step. For direct conversion from a pre-trained ANN to an SNN, Ding et al. (2021) proposed Rate Norm Layer to replace the ReLU activation function in source ANN training, and Li et al. (2021) proposed calibration for weights and biases using quantized fine-tuning to correct the error layer-by-layer. Our work share similarity with Deng & Gu (2021); Bu et al. (2021) which are also on optimal conversion. Deng & Gu (2021) minimized the layer-wise error by shift-threshold ReLU which only considers the deviation from the

Table 2: Comparison between the proposed SlipReLU method and other methods on CIFAR10

Architecture	Method	ANN	T=1	T=2	T=4	T=8	T=16	T=32	T=64	T=128
ResNet-20	RTS	91.46	10.00	10.00	10.00	10.00	10.00	19.77	90.34	90.54
ResNet-18	RNL	79.71	10.00	12.20	14.30	32.20	58.70	70	73.20	74.50
VGG-16	HC	92.54	10.00	10.00	10.00	10.00	10.00	11.13	83.23	91.71
ResNet-18	QCFS	96.00	48.40	61.10	75.30	87.47	93.44	**95.63**	**96.01**	**96.14**
ResNet-18	Ours	**96.19**	**55.55**	**65.34**	**77.97**	**89.22**	**93.88**	95.53	95.86	95.94

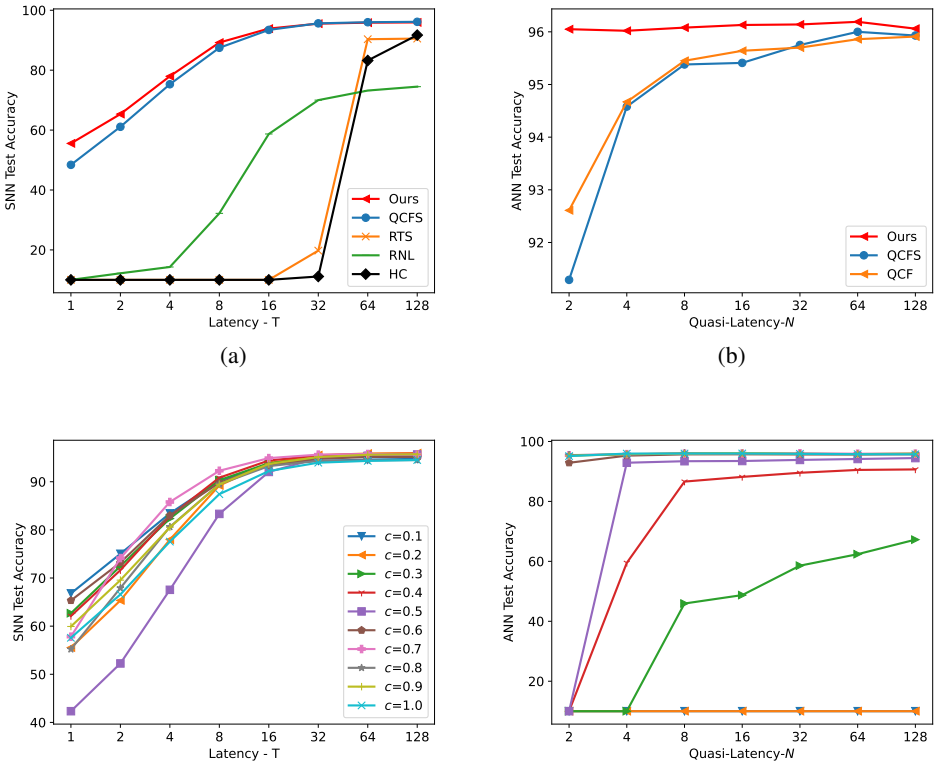

Figure 2: (a) SNN accuracy of the proposed SlipReLU method compared against state-of-the-art methods over different latency values. (b) shows the impact of Quasi-latency N on test accuracy of the trained ANN model. (c) SNN accuracy of the proposed models with different slope c values. (d) Test accuracy of the ANN model with shift term $\delta_1 = -\frac{1}{2N}, \delta = 0$.

ReLU in the unified optimization framework in Sect. 3. Bu et al. (2021) proposed to use the quantization clip-floor-shift activation function to train ANNs and the clip-floor-shift activation function only minimizes the conversion error neglecting the performance loss of the tailored ANN with new activation function. They all got the theoretical "optimal" results with some fixed shift term. In comparison, our proposed unified framework gives more flexibility for different application scenarios to covert ANN into SNN with techniques eliminating the conversion error and keeping the ANN performance with less deviation from the regular ANN with ReLU. Our SlipReLU can balance the trade-off between the ANN performance and the conversion error simultaneously.

6 Experiments

In this section, we validate our unified framework and compare our SlipReLU method with existing state-of-the-art approaches for image classification task on CIFAR-10 (LeCun et al., 1998). Similar

to previous works, we utilize VGG-16, ResNet-18, and ResNet-20 network structures for source ANNs. We compare our method with the state-of-the-art ANN-SNN conversion methods, including Hybrid-Conversion (HC) from Rathi et al. (2020), RNL from Ding et al. (2021), ReLUThreshold-Shift(RTS) from Deng & Gu (2021) and optimal ANN-SNN conversion with quantization clip-floor-shift activation function(QCFS) from Bu et al. (2021). Refer to Appendix E for the details of the network infrastructure and training setups.

6.1 Comparison with the state-of-the-art conversion methods

We evaluate and compare the performance of the proposed SlipReLU with other methods in terms of SNN accuracy and results are shown in Table 2. Our proposed SlipReLU has the best SNN accuracy compared to existing state-of-the-art ANN-SNN conversion methods when we consider low-latency inference ($T \leq 16$). Our SlipReLU method is able to achieve an accuracy of 55.55% compared to the next best baseline QCFS (48.40%) with a good margin when the latency $T = 1$. Moreover, the ANN testing accuracy is the best among all the other models, as our SlipReLU method is designed to consider both the ANN accuracy and the conversion error simultaneously.

6.2 Impact of quasi-latency N and impact of the slope c

We give a clear figure on the SNN accuracy of our proposed SlipReLU method and other state-of-the-art method in (a) of Fig. 2). Our proposed SlipReLU method indeed gives the best SNN accuracy. The most competitive method of our SlipReLU is the QCFS and QCF methods, however, they do not give as high performance as our SlipReLU with respect to the ANN accuracy, which can be seen from (b) of Fig. 2). The reason is that the QCFS only considers the conversion error but not the ANN performance, whereas, our SlipReLU is proposed to consider the conversion error as well as the ANN performance.

We have also investigated impact of the slope c on the SNN accuracy over different latency length T (see (c) in Fig. 2). It shows the slop c can affect the SNN accuracy when the latency T is small, which gives the flexibility to apply our SlipReLU to different scenarios. We have evaluated the impact of the slope c on the ANN accuracy of the trained ANN models over different quasi-latency N. (see (b) and (d) in Fig. 2). We do not want the ANN model be effected by the value of quasi-latency N, as this may further impact the performance of the converted SNN model. From (b) and (d) of Fig. 2 it can be observed that, test ANN accuracy of our model is not effected by the quasi-latency value. Whereas, the baselines QCFS and QCF (QCFS without shift) are impacted by the value of quasi-latency.

7 Discussion and Conclusion

The performance of the converted SNN is determined by both the ANN performance and the conversion error. The performance loss between the regular ANN with regular ReLU and the tailored ANN has never been considered in the existing ANN-SNN conversion methods, which will be inherited to the converted SNN. In this work, we formulate the ANN-SNN conversion as a unified optimization problem which considers the performance loss between the regular ANN with and the tailored ANN, as well as the conversion error simultaneously. Following the unified optimization framework, we propose the SlipReLU activation function to replace the regular ReLU activation function in the tailored ANN. The SlipReLU is a weighted sum of the shift-threhold-ReLU and the step function, which improves the performance of either as an activation function alone. The SlipReLU method covers a family of activation functions mapping from activation values in source ANNs to firing rates in target SNNs; most of the state-of-the-art optimal ANN-SNN conversion methods are special cases of our proposed SlipReLU method. We demonstrate through two theorems that the expected conversion error between SNNs and ANNs can theoretically be zero on a range of shift values $\delta \in [-\frac{1}{2}, \frac{1}{2}]$ rather than a fixed shift term $\frac{1}{2}$, enabling us to achieve converted SNNs with high accuracy and ultra-low latency. We evaluate our proposed SlipReLU method on CIFAR-10 dataset, and the results show that our proposed SlipReLU outperforms the state-of-the-art ANN-SNN conversion in both accuracy and latency. To our knowledge, this is the first work to explore high-performance ANN-SNN conversion method considering the ANN performance and the conversion error simultaneously.

REFERENCES

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *International Conference on Learning Representations*, 2021.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. ieee, 2015.

Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *In International Joint Conference on Artificial Intelligence*, pp. 2328–2336, 2021.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13558–13567, 2020.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *https://www.cs.toronto.edu/ kriz/cifar.html*, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021.

Ying-Hui Liu and Xiao-Jing Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of computational neuroscience*, 10(1):25–45, 2001.

Patrick Mehlitz and Matúš Benko. On implicit variables in optimization theory. *Journal of Nonsmooth Analysis and Optimization*, 2, 2021.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. 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, 2019.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

NOTATIONS IN THE APPENDIX

Through out this Appendix, we use the following notations. Bold-face lower-case letters refer to vectors, and normal-face letters refer to scalars. Note $\mathbf{V}_{\text{th}}^{(\ell)}$ and $\boldsymbol{\theta}^{(\ell)}$ are vectors whose dimensions match the number of neurons in the layer of interest, and denote $\mathbf{V}_{\text{th}}^{(\ell)} = [V_{\text{th}}^{(\ell)}]$ and $\boldsymbol{\theta}^{(\ell)} = [\theta^{(\ell)}]$ respectively. Namely, vector $\mathbf{V}_{\text{th}}^{(\ell)} = [V_{\text{th}}^{(\ell)}]$ means that each element is the same $V_{\text{th}}^{(\ell)}$. Denote $\boldsymbol{\delta} = [\delta]$.

A ANALYSIS OF ACTIVATION FUNCTION IN SNNS

We will derive the activation function of SNN, $\mathcal{F}_{\text{SNN}}(\cdot)$ in this section.

The activation function of SNN gives the relationship between activation values $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$ of successive layers of SNN, which defines input-output function mapping for adjacent layers.

Specifically, we can get the potential update equation by combining Eq. (2) and Eq. (4),

$$\mathbf{v}^{(\ell)}(t) = \mathbf{v}^{(\ell)}(t-1) + \mathbf{W}^{(\ell)}\mathbf{x}^{(\ell-1)}(t) - \mathbf{s}^{(\ell)}(t)V_{\text{th}}^{(\ell)} \,. \tag{A.1}$$

By summing the time-step from time 1 to T, then we get

$$\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0) = \mathbf{W}^{(\ell)}\sum_{t=1}^{T}\mathbf{x}^{(\ell-1)}(t) - \sum_{t=1}^{T}\mathbf{s}^{(\ell)}(t)V_{\text{th}}^{(\ell)} \,. \tag{A.2}$$

Due to the spike-in-spike-out property of the IF neurons in SNN, the output at each time-step can be ether 0 or 1. For each neuron i, let $m_i = \sum_{t=1}^{T} s_i^{(\ell)}(t)$, and each $m_i \in \{0, 1, 2, \cdots, T\}$ denotes the total number of spikes of each neuron i. Then $\mathbf{m} = \{m_i\}$ is the vector collecting all the number of spikes of all neurons in the ℓ-th layer. The accumulated spikes $\mathbf{m} = \sum_{t=1}^{T}\mathbf{s}^{(\ell)}(t)$ denotes the total number of spikes. According to the above equations, we have

$$\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0) = \mathbf{W}^{(\ell)}T \cdot \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{m}V_{\text{th}}^{(\ell)} \,. \tag{A.3}$$

Then we get

$$\mathbf{m}V_{\text{th}}^{(\ell)} = T\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)} - (\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0)) \,. \tag{A.4}$$

A.1 ELEMENT-WISE VERSION DERIVATION

Denote

$$\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)} \,.$$

We use $\mathrm{z}_i^{(\ell)}$, $\mathrm{v}_i^{(\ell)}(T)$, $\mathrm{v}_i^{(\ell)}(0)$, and m_i to denote the i-th element in vector $\mathbf{z}^{(\ell)}$, $\mathbf{v}^{(\ell)}(T)$, $\mathbf{v}^{(\ell)}(0)$, and $\mathbf{m}$ respectively. That is, $\mathbf{z}^{(\ell)} = \{\mathrm{z}_i^{(\ell)}\}$, $\mathbf{v}^{(\ell)}(T) = \{\mathrm{v}_i^{(\ell)}(T)\}$, $\mathbf{v}^{(\ell)}(0) = \{\mathrm{v}_i^{(\ell)}(0)\}$, and $\mathbf{m} = \{m_i\}$.

Then we have

$$\begin{aligned}
&\mathbf{m}V_{\text{th}}^{(\ell)} = T\mathbf{z}^{(\ell)} - (\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0)) \\
\iff & m_i V_{\text{th}}^{(\ell)} = T\mathrm{z}_i^{(\ell)} - (\mathrm{v}_i^{(\ell)}(T) - \mathrm{v}_i^{(\ell)}(0)) \ \text{(For each neuron } i \text{ with } \mathbf{m} = \{m_i\},\ \mathbf{z}^{(\ell)} = \{\mathrm{z}_i^{(\ell)}\})\ .
\end{aligned}$$

Note that we assume the terminal membrane potential $\mathrm{v}_i^{(\ell)}(T)$ lies within the range $[0, V_{\text{th}}^{(\ell)})$, by further assuming $\mathrm{v}_i^{(\ell)}(0) = 0$, we get

$$\begin{aligned}
& 0 \leqslant \mathrm{v}_i^{(\ell)}(T) < V_{\text{th}}^{(\ell)} \\
\iff & -V_{\text{th}}^{(\ell)} < -\mathrm{v}_i^{(\ell)}(T) \leqslant 0 \quad \text{(adding } T\mathrm{z}_i^{(\ell)} \text{ to each term)} \\
\iff & T\mathrm{z}_i^{(\ell)} - V_{\text{th}}^{(\ell)} < T\mathrm{z}_i^{(\ell)} - \mathrm{v}_i^{(\ell)}(T) \leqslant T\mathrm{z}_i^{(\ell)} \quad (m_i = T\mathrm{z}_i^{(\ell)} - \mathrm{v}_i^{(\ell)}(T)) \\
\iff & T\mathrm{z}_i^{(\ell)} - V_{\text{th}}^{(\ell)} < m_i V_{\text{th}}^{(\ell)} \leqslant T\mathrm{z}_i^{(\ell)} \\
\iff & \frac{T\mathrm{z}_i^{(\ell)} - V_{\text{th}}^{(\ell)}}{V_{\text{th}}^{(\ell)}} < m_i \leqslant \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}}\ .
\end{aligned}$$

Then we use floor operation and clip operation to determine the totoal number of spikes, m_i,

$$\begin{aligned}
m_i &= \text{clip}\left(\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} \right\rfloor, 0, T\right) \ (\text{ and } m_i = T\bar{\mathrm{s}}_i^{(\ell)}) \\
\bar{\mathrm{s}}_i^{(\ell)} &= \text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} \right\rfloor, 0, 1\right) \ (\text{ and } \bar{\mathrm{x}}_i^{(\ell)} = V_{\text{th}}^{(\ell)}\bar{\mathrm{s}}_i^{(\ell)}) \\
\bar{\mathrm{x}}_i^{(\ell)} &= V_{\text{th}}^{(\ell)}\text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} \right\rfloor, 0, 1\right)\ .
\end{aligned}$$

The assumption $\mathrm{v}_i^{(\ell)}(0) = 0$ may be too strong, without it, we will get

$$\begin{aligned}
\iff & T\mathrm{z}_i^{(\ell)} - V_{\text{th}}^{(\ell)} + \mathrm{v}_i^{(\ell)}(0) < m_i V_{\text{th}}^{(\ell)} \leqslant T\mathrm{z}_i^{(\ell)} + \mathrm{v}_i^{(\ell)}(0) \\
\iff & \frac{T\mathrm{z}_i^{(\ell)} - V_{\text{th}}^{(\ell)} + \mathrm{v}_i^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}} < m_i \leqslant \frac{T\mathrm{z}_i^{(\ell)} + \mathrm{v}_i^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}} \\
\iff & \frac{T\mathrm{z}_i^{(\ell)} - V_{\text{th}}^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \delta < m_i \leqslant \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \delta \quad \text{with} \quad \delta = \frac{\mathrm{v}_i^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}}\ .
\end{aligned}$$

Denote $\delta = \frac{\mathrm{v}_i^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}}$. Then we have

$$\begin{aligned}
m_i &= \text{clip}\left(\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \delta \right\rfloor, 0, T\right) \ (\text{ and } m_i = T\bar{\mathrm{s}}_i^{(\ell)}) \\
\bar{\mathrm{s}}_i^{(\ell)} &= \text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \delta \right\rfloor, 0, 1\right) \ (\text{ and } \bar{\mathrm{x}}_i^{(\ell)} = V_{\text{th}}^{(\ell)}\bar{\mathrm{s}}_i^{(\ell)}) \\
\bar{\mathrm{x}}_i^{(\ell)} &= V_{\text{th}}^{(\ell)}\text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \delta \right\rfloor, 0, 1\right)\ .
\end{aligned}$$

The relationship between activation values $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$ of successive layers of SNN can be formulated as

$$\bar{\mathrm{x}}_i^{(\ell)} = V_{\text{th}}^{(\ell)}\text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathrm{z}_i^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \delta \right\rfloor, 0, 1\right)\ .$$

A.2 VECTOR VERSION DERIVATION

The accumulated spikes $\mathbf{m} = \sum_{t=1}^{T} \mathbf{s}^{(\ell)}(t)$ denotes the total number of spikes, and $\mathbf{m} = \{m_i\}$ is the vector collecting all the number of spikes of all neurons in the ℓ-th layer. Each $m_i \in \{0, 1, 2, \cdots, T\}$ denotes the total number of spikes of each neuron i. According to the above equations, we have

$$\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0) = \mathbf{W}^{(\ell)} T \cdot \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{m} V_{\text{th}}^{(\ell)} \,. \tag{A.5}$$

Then we get

$$\mathbf{m} V_{\text{th}}^{(\ell)} = T \mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - (\mathbf{v}^{(\ell)}(T) - \mathbf{v}^{(\ell)}(0)) \,. \tag{A.6}$$

Note that we assume the terminal membrane potential $\mathbf{v}^{(\ell)}(T)$ lies within the range $[\mathbf{0}, \mathbf{V}_{\text{th}}^{(\ell)})$, by further assuming $\mathbf{v}^{(\ell)}(0) = \mathbf{0}$, we get

$$\begin{aligned}
& \mathbf{0} \leqslant \mathbf{v}^{(\ell)}(T) < \mathbf{V}_{\text{th}}^{(\ell)} \\
\Longleftrightarrow & -\mathbf{V}_{\text{th}}^{(\ell)} < -\mathbf{v}^{(\ell)}(T) \leqslant \mathbf{0} \\
\Longleftrightarrow & T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)} < T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{v}^{(\ell)}(T) \leqslant T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} \\
\Longleftrightarrow & T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)} < \mathbf{m} V_{\text{th}}^{(\ell)} \leqslant T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} \\
\Longleftrightarrow & \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)}}{V_{\text{th}}^{(\ell)}} < \mathbf{m} \leqslant \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} \,.
\end{aligned}$$

Then we use floor operation and clip operation to determine the totoal number of spikes, $\mathbf{m}$,

$$\begin{aligned}
\mathbf{m} &= \text{clip}\left(\left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} \right\rfloor, 0, T \right) \quad (\text{ and } \mathbf{m} = T\bar{\mathbf{s}}^{(\ell)}) \\
\bar{\mathbf{s}}^{(\ell)} &= \text{clip}\left(\frac{1}{T} \left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} \right\rfloor, 0, 1 \right) \quad (\text{ and } \bar{\mathbf{x}}^{(\ell)} = V_{\text{th}}^{(\ell)} \bar{\mathbf{s}}^{(\ell)}) \\
\bar{\mathbf{x}}^{(\ell)} &= V_{\text{th}}^{(\ell)} \text{clip}\left(\frac{1}{T} \left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} \right\rfloor, 0, 1 \right) \,.
\end{aligned}$$

The assumption $\mathbf{v}^{(\ell)}(0) = \mathbf{0}$ may be too strong, without it, we will get

$$\begin{aligned}
\Longleftrightarrow & T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)} + \mathbf{v}^{(\ell)}(0) < \mathbf{m} V_{\text{th}}^{(\ell)} \leqslant T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} + \mathbf{v}^{(\ell)}(0) \\
\Longleftrightarrow & \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)} + \mathbf{v}^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}} < \mathbf{m} \leqslant \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} + \mathbf{v}^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}} \\
\Longleftrightarrow & \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)} - \mathbf{V}_{\text{th}}^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} < \mathbf{m} \leqslant \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \quad \text{with} \quad \boldsymbol{\delta} = \frac{\mathbf{v}^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}} \,.
\end{aligned}$$

Denote $\boldsymbol{\delta} = \frac{\mathbf{v}^{(\ell)}(0)}{V_{\text{th}}^{(\ell)}}$. Note $\boldsymbol{\delta}$ is a vector whose dimension matches the number of neurons in that layer. Then we have

$$\begin{aligned}
\mathbf{m} &= \text{clip}\left(\left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, T \right) \quad (\text{ and } \mathbf{m} = T\bar{\mathbf{s}}^{(\ell)}) \\
\bar{\mathbf{s}}^{(\ell)} &= \text{clip}\left(\frac{1}{T} \left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, 1 \right) \quad (\text{ and } \bar{\mathbf{x}}^{(\ell)} = V_{\text{th}}^{(\ell)} \bar{\mathbf{s}}^{(\ell)}) \\
\bar{\mathbf{x}}^{(\ell)} &= V_{\text{th}}^{(\ell)} \text{clip}\left(\frac{1}{T} \left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, 1 \right) \,.
\end{aligned}$$

The relationship between activation values $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$ of successive layers of SNN can be formulated as

$$\bar{\mathbf{x}}^{(\ell)} = V_{\text{th}}^{(\ell)} \text{clip}\left(\frac{1}{T} \left\lfloor \frac{T\mathbf{W}^{(\ell)} \bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, 1 \right) \,.$$

Note $\mathbf{V}_{\text{th}}^{(\ell)}$ is a vector whose dimension matches the number of neurons in that layer, and $\mathbf{V}_{\text{th}}^{(\ell)} = [V_{\text{th}}^{(\ell)}]$ means each element is the same $V_{\text{th}}^{(\ell)}$.

Denote

$$\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)} .$$

Then

$$\bar{\mathbf{x}}^{(\ell)} = V_{\text{th}}^{(\ell)}\text{clip}\left(\frac{1}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta}\right\rfloor, 0, 1\right) .$$

B Derivation of SlipReLU activation function

In this section, we will give detailed derivation of the proposed SlipReLU activation function in Eq. (9) and its extension in Eq. (10) with different shift modes. In ANNs, denote

$$\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{x}^{(\ell-1)} .$$

Then the forward propagation of activation values through layers in the ANN is

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = \mathcal{F}_{\text{ANN}}(\mathbf{W}^{(\ell)}\mathbf{x}^{(\ell-1)}) .$$

Derivation of SlipReLU activation function in Eq. (9). We start with the initial definition of the SlipReLU function in Eq. (B.1),

$$\text{SlipReLU}(\mathbf{z}^{(\ell)}) = \begin{cases} \mathbf{0} & \text{if } \mathbf{z}^{(\ell)} < \mathbf{0} \\ c\mathbf{z}^{(\ell)} + (1-c)\frac{k\boldsymbol{\theta}^{(\ell)}}{N} & \text{if } \frac{k\boldsymbol{\theta}^{(\ell)}}{N} \leqslant \mathbf{z}^{(\ell)} < \frac{(k+1)\boldsymbol{\theta}^{(\ell)}}{N} \\ \boldsymbol{\theta}^{(\ell)} & \text{if } \mathbf{z}^{(\ell)} \geqslant \boldsymbol{\theta}^{(\ell)} \end{cases} . \tag{B.1}$$

Here $k = 0, 1, \cdots, N-1$. Note $\boldsymbol{\theta}^{(\ell)}$ should be a vector whose dimension matches the number of neurons in that layer, $\boldsymbol{\theta}^{(\ell)} = [\theta^{(\ell)}]$.

Then we can rewrite it to

$$\text{SlipReLU}(\mathbf{z}^{(\ell)}) = \mathbf{y}_{\text{temp}} + c\cdot(\mathbf{z}_{\text{temp}} - \mathbf{y}_{\text{temp}}) = c\mathbf{z}_{\text{temp}} + (1-c)\mathbf{y}_{\text{temp}}$$
$$\text{where} \quad \mathbf{z}_{\text{temp}} = \theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right) \text{, and } \mathbf{y}_{\text{temp}} = \frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\cdot\mathbf{z}_{\text{temp}}}{\theta^{(\ell)}}\right\rfloor .$$

Here

$$\begin{aligned} \mathbf{y}_{\text{temp}} &= \frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathbf{z}_{\text{temp}}}{\theta^{(\ell)}}\right\rfloor \\ \Longleftrightarrow \mathbf{y}_{\text{temp}} &= \frac{\theta^{(\ell)}}{N}\left\lfloor N\cdot\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right)\right\rfloor \\ \Longleftrightarrow \mathbf{y}_{\text{temp}} &= \theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor N\cdot\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) \\ \Longleftrightarrow \mathbf{y}_{\text{temp}} &= \theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) . \end{aligned}$$

Then Eq. (B.1) can be written as follows,

$$\begin{aligned} \boldsymbol{a}^{(\ell)} &= \text{SlipReLU}(\mathbf{z}^{(\ell)}) \\ &= c\mathbf{z}_{\text{temp}} + (1-c)\mathbf{y}_{\text{temp}} \\ &= c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) . \end{aligned}$$

That is the SlipReLU activation function in Eq. (9),

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) .$$

Derivation of SlipReLU extension in Eq. (10) with shift As mentioned in Sect. 4, the SlipReLU activation function in Eq. (9) in a weighted combination of the threshold-ReLU (first part) and the step function (second part), with the slope $0 \leqslant c \leqslant 1$ balancing the weight, then any shift to these two parts will lead to shift in the SlipReLU activation function. The SlipReLU extension with in Eq. (10) can be formulated as follows,

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta}_1, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta}\right\rfloor, 0, 1\right) .$$

The shift term $\delta_1 \in [-N, 0]$ and $\delta \in [-\frac{1}{2}, \frac{1}{2}]$ for the source ANNs. And $\boldsymbol{\delta}_1 = [\delta_1]$, $\boldsymbol{\delta} = [\delta]$.

Here we list several examples of the proposed SlipReLU with different shift modes.

Mode 0 We set $\delta_1 = \delta = 0$, then

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) .$$

Mode 1 We set $\delta_1 = -\frac{1}{2}, \delta = 0$ then

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} - [\frac{1}{2}], 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}\right\rfloor, 0, 1\right) .$$

Mode 2 We set $\delta_1 = 0, \delta = \frac{1}{2}$, then

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}, 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + [\frac{1}{2}]\right\rfloor, 0, 1\right) .$$

Mode 3 We set $\delta_1 = -\frac{1}{2N}, \delta = -\frac{1}{2}$, then

$$\boldsymbol{a}^{(\ell)} = \mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} - [\frac{1}{2N}], 0, 1\right) + (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} - [\frac{1}{2}]\right\rfloor, 0, 1\right) .$$

Derivation of SlipReLU special cases Here we list four different special cases of the proposed SlipReLU.

Threshold-ReLU When $c = 1$ and $\delta_1 = 0$, the SlipReLU becomes the threshold ReLU activation function which is studied in Deng & Gu (2021).

Shift-threshold-ReLU When $c = 1$ and $\delta_1 = -1/(2N)$, the SlipReLU becomes the shift-threshold ReLU activation function which is studied in Deng & Gu (2021).

Quantization clip-floor (QCF) When $c = 0$ and $\delta = 0$, the SlipReLU becomes the quantization clip-floor (QCF) activation function which is studied in Bu et al. (2021).

Quantization clip-floor-shift (QCFS) When $c = 0$ and $\delta = 1/2$, the SlipReLU becomes the quantization clip-floor-shift (QCFS) activation function which is studied in Bu et al. (2021).

C Proof of Theorems

Before we proof Theorem 1 and Theorem 2, we first introduce an important Lemma.

Lemma 1. *If a random variable $x \in [0, \theta]$ is uniformly distributed in every small interval (m_t, m_{t+1}) with p_t $(t = 0, 1, \cdots, T)$, where $m_0 = 0, m_{T+1} = \theta, m_t = \frac{(2t-1)\theta}{2T}$ for $t = 1, 2, \cdots, T$, $p_0 = p_T$. For any value $\delta \in [-\frac{1}{2}, \frac{1}{2}]$, then we can conclude that*

$$\mathbb{E}_x\left(\left|x - \frac{\theta}{T}\left\lfloor \frac{Tx}{\theta} + \delta\right\rfloor\right|\right) = 0 . \tag{C.1}$$

Proof. We consider x in different small intervals (m_t, m_{t+1}).

(1) For $x \in \left(0, \frac{\theta}{2T}\right)$,

$$0 < x < \frac{\theta}{2T} \Longleftrightarrow \delta < \frac{Tx}{\theta} + \delta < \frac{1}{2} + \delta \Longleftrightarrow \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor = 0 \, .$$

(2) For $x \in \left(\frac{(2t-1)\theta}{2T}, \frac{(2t+1)\theta}{2T}\right)$, and $t = 1, 2, \cdots, T-1$

$$\frac{(2t-1)\theta}{2T} < x < \frac{(2t+1)\theta}{2T} \Longleftrightarrow t - \frac{1}{2} + \delta < \frac{Tx}{\theta} + \delta < t + \frac{1}{2} + \delta \Longleftrightarrow \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor = t \, .$$

(3) For $x \in \left(\frac{(2T-1)\theta}{2T}, \theta\right)$,

$$\frac{(2T-1)\theta}{2T} < x < \theta \Longleftrightarrow T - \frac{1}{2} + \delta < \frac{Tx}{\theta} + \delta < T + \frac{1}{2} + \delta \Longleftrightarrow \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor = T \, .$$

Then we have

$$\begin{aligned}
&\mathbb{E}_x \left(\left| x - \frac{\theta}{T} \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor \right| \right) \\
=& \int_0^{\theta/2T} p_0 \left| x - \frac{\theta}{T} \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor \right| dx + \sum_{t=1}^{T-1} p_t \int_{(2t-1)\theta/2T}^{(2t+1)\theta/2T} \left| x - \frac{\theta}{T} \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor \right| dx \\
&+ \int_{(2T-1)\theta/2T}^{\theta} p_T \left| x - \frac{\theta}{T} \left\lfloor \frac{Tx}{\theta} + \delta \right\rfloor \right| dx \\
=& p_0 \int_0^{\theta/2T} |x| \, dx + \sum_{t=1}^{T-1} p_t \int_{(2t-1)\theta/2T}^{(2t+1)\theta/2T} \left| x - \frac{t\theta}{T} \right| dx + p_T \int_{(2T-1)\theta/2T}^{\theta} |x - \theta| \, dx \\
=& p_0 \int_0^{\theta/2T} x dx + \sum_{t=1}^{T-1} p_t \int_{(2t-1)\theta/2T}^{(2t+1)\theta/2T} \left(x - \frac{t\theta}{T} \right) dx + p_T \int_{(2T-1)\theta/2T}^{\theta} (x - \theta) \, dx \\
=& p_0 \frac{\theta^2}{8T^2} + 0 - p_T \frac{\theta^2}{8T^2} = 0 \, .
\end{aligned}$$

□

Lemma 2. *Let* P *be a probability distribution on* $\mathbb{R}$*. If a random variable* $\mathbf{z} \in \mathbb{R}^m$ *and* $\mathbf{z} \sim \mathrm{P}$*, a function* $\boldsymbol{g} : \mathbf{z} \to \boldsymbol{g}(\mathbf{z}) \in \mathbb{R}^n$ *and* $\boldsymbol{g}(\mathbf{z}) \geqslant \mathbf{0}$ *almost surely for* $\forall\, \mathbf{z} \in D$*, and*

$$\mathbb{E}_{\mathbf{z}} \left| \boldsymbol{g}(\mathbf{z}) \right| = 0 \, ,$$

then we have

$$\mathbb{E}_{\mathbf{z}} \left\| \boldsymbol{g}(\mathbf{z}) \right\|_2 = 0 \, .$$

Proof. By the definition of L_2-norm, we have

$$\left\| \boldsymbol{g}(\mathbf{z}) \right\|_2 = \sqrt{g_1^2(z) + g_2^2(z) + \cdots + g_n^2(z)} \leqslant |g_1(z)| + |g_2(z)| + \cdots + |g_n(z)| \, .$$

Then, we can get

$$\begin{aligned}
\mathbb{E}_{\mathbf{z}} \left\| \boldsymbol{g}(\mathbf{z}) \right\|_2 &\leqslant \mathbb{E}_{\mathbf{z}} |g_1(z)| + \mathbb{E}_{\mathbf{z}} |g_2(z)| + \cdots + \mathbb{E}_{\mathbf{z}} |g_n(z)| \\
&= \mathbb{E}_{\mathbf{z}} g_1(z) + \mathbb{E}_{\mathbf{z}} g_2(z) + \cdots + \mathbb{E}_{\mathbf{z}} g_n(z) = 0 \, .
\end{aligned}$$

Then

$$\mathbb{E}_{\mathbf{z}} \left\| \boldsymbol{g}(\mathbf{z}) \right\|_2 = 0 \, .$$

□

C.1 PROOF OF THEOREM 1

For Theorem 1, we need to prove

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\mathrm{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \mathbf{0}\,.$$

Proof. The activation function of the SNN is

$$\mathcal{F}_{\mathrm{SNN}}(\mathbf{z}^{(\ell)}) = V_{\mathrm{th}}^{(\ell)}\mathrm{clip}\left(\frac{1}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor, 0, 1\right)\,.$$

For $c=0$, the SlipReLU activation function used in the source ANN then becomes

$$\mathcal{F}_{\mathrm{ANN}}(\mathbf{z}^{(\ell)}) = \theta^{(\ell)}\mathrm{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor, 0, 1\right)\,.$$

With $V_{\mathrm{th}}^{(\ell)} = \theta^{(\ell)}$, then the error becomes

$$\mathrm{Err}^{(\ell)} = \mathcal{F}_{\mathrm{SNN}}(\mathbf{z}^{(\ell)}) - \mathcal{F}_{\mathrm{ANN}}(\mathbf{z}^{(\ell)}) = \frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\,.$$

Then

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}}\left(\left|\mathrm{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]}\\
=&\mathbb{E}_{\mathbf{z}}\left(\left|\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right|\right)\\
\leqslant&\,\mathbb{E}_{\mathbf{z}}\left(\left|\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor - \mathbf{z}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} + \mathbb{E}_{\mathbf{z}}\left(\left|\mathbf{z}^{(\ell)} - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right|\right)\,.
\end{aligned}$$

Denote $\mathrm{v}_i^{(\ell)}(0)$ and z_i the i-th element of vector $\mathbf{v}^{(\ell)}(0)$ and $\mathbf{z}$. Denote $\boldsymbol{\delta} = [\delta]$. Then we need to consider every element of vector $\mathbf{z}$.

$$\begin{aligned}
&\mathbb{E}_{\mathrm{z}_i}\left(\left|\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathrm{z}_i^{(\ell)}}{\theta^{(\ell)}}+\delta\right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathrm{z}_i^{(\ell)}+\mathrm{v}_i^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right|\right)\\
\leqslant&\,\mathbb{E}_{\mathrm{z}_i}\left(\left|\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathrm{z}_i^{(\ell)}}{\theta^{(\ell)}}+\delta\right\rfloor - \mathrm{z}_i^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} + \mathbb{E}_{\mathrm{z}_i}\left(\left|\mathrm{z}_i^{(\ell)} - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathrm{z}_i^{(\ell)}+\mathrm{v}_i^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right|\right)\,.
\end{aligned} \tag{C.2}$$

Then according to Lemma 1, we have

$$\begin{aligned}
&\mathbb{E}_{\mathrm{z}_i}\left(\left|\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathrm{z}_i^{(\ell)}}{\theta^{(\ell)}}+\delta\right\rfloor - \mathrm{z}_i^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = 0\\
&\mathbb{E}_{\mathrm{z}_i}\left(\left|\mathrm{z}_i^{(\ell)} - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathrm{z}_i^{(\ell)}+\mathrm{v}_i^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right|\right)\Big|_{\mathrm{v}_i^{(\ell)}(0)=\delta V_{\mathrm{th}}^{(\ell)}} = 0\,.
\end{aligned}$$

This holds for any shift value δ in the ANNs when $-\frac{1}{2}\leqslant\delta\leqslant\frac{1}{2}$, which gives the conclusion of the Theorem 1.

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\mathrm{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \mathbf{0}\,.$$

□

C.2 PROOF OF THEOREM 2

For Theorem 2, we need to prove,

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\mathrm{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \frac{c(V_{\mathrm{th}}^{(\ell)})^2}{4T}\,, \tag{C.3}$$

Proof. The activation function of the SNN is

$$\mathcal{F}_{\mathrm{SNN}}(\mathbf{z}^{(\ell)}) = V_{\mathrm{th}}^{(\ell)}\mathrm{clip}\left(\frac{1}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor, 0, 1\right)\,.$$

For arbitrary $c\in[0,1]$, the SlipReLU activation function used in the source ANN then becomes

$$\mathcal{F}_{\mathrm{ANN}}(\mathbf{z}^{(\ell)}) = c\theta^{(\ell)}\mathrm{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}_1, 0, 1\right) + (1-c)\theta^{(\ell)}\mathrm{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor, 0, 1\right)\,.$$

With $V_{\mathrm{th}}^{(\ell)} = \theta^{(\ell)}$, then the error becomes,

$$\begin{aligned}
&\mathrm{Err}^{(\ell)} = \mathcal{F}_{\mathrm{ANN}}(\mathbf{z}^{(\ell)}) - \mathcal{F}_{\mathrm{SNN}}(\mathbf{z}^{(\ell)})\\
&= c\left\{\theta^{(\ell)}\mathrm{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}_1, 0, 1\right) - V_{\mathrm{th}}^{(\ell)}\mathrm{clip}\left(\frac{1}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor, 0, 1\right)\right\}\\
&\qquad + (1-c)\left\{\theta^{(\ell)}\mathrm{clip}\left(\frac{1}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor, 0, 1\right) - V_{\mathrm{th}}^{(\ell)}\mathrm{clip}\left(\frac{1}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor, 0, 1\right)\right\}\\
&= c\left\{\mathbf{z}^{(\ell)}+\boldsymbol{\delta}_1\theta^{(\ell)} - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right\} \quad (\text{with } \mathbf{v}^{(\ell)}(0) = V_{\mathrm{th}}^{(\ell)}\boldsymbol{\delta}, V_{\mathrm{th}}^{(\ell)} = \theta^{(\ell)})\\
&\qquad + (1-c)\left\{\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{\theta^{(\ell)}}\right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right\}\\
&= c\left\{\mathbf{z}^{(\ell)}+V_{\mathrm{th}}^{(\ell)}\boldsymbol{\delta}_1 - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}}{V_{\mathrm{th}}^{(\ell)}}+\boldsymbol{\delta}\right\rfloor\right\} \triangleq c\cdot\mathrm{Err}_1 \qquad (\mathrm{C.4})\\
&\qquad + (1-c)\left\{\frac{\theta^{(\ell)}}{N}\left\lfloor\frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}}+\boldsymbol{\delta}\right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}+\mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right\} \triangleq (1-c)\cdot\mathrm{Err}_2 \qquad (\mathrm{C.5})
\end{aligned}$$

Then

$$\begin{aligned}
&\mathrm{Err}^{(\ell)} \triangleq c\cdot\mathrm{Err}_1 + (1-c)\cdot\mathrm{Err}_2\\
\Longrightarrow &\left|\mathrm{Err}^{(\ell)}\right| = |c\cdot\mathrm{Err}_1 + (1-c)\cdot\mathrm{Err}_2| \leqslant c\cdot|\mathrm{Err}_1| + (1-c)\cdot|\mathrm{Err}_2|\,.
\end{aligned}$$

So we can minimize the whole error by minimized each of the two terms.

Let $\delta_1 = \frac{\phi+\delta}{T}$. For Eq. (C.4), we have

$$\begin{aligned}
|\mathrm{Err}_1| &\triangleq \left|\mathbf{z}^{(\ell)} + \frac{V_{\mathrm{th}}^{(\ell)}}{T}(\boldsymbol{\phi}+\boldsymbol{\delta}) - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}}{V_{\mathrm{th}}^{(\ell)}}+\boldsymbol{\delta}\right\rfloor\right|\\
&= \left|\mathbf{z}^{(\ell)} + \frac{V_{\mathrm{th}}^{(\ell)}}{T}\boldsymbol{\phi} - \frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor\right|\,. \qquad (\mathrm{C.6})
\end{aligned}$$

Here

$$\begin{aligned}
&\mathbf{z}^{(\ell)} + \frac{V_{\mathrm{th}}^{(\ell)}}{T}\boldsymbol{\phi} && \text{is the activation function of ANN}\\
&\frac{V_{\mathrm{th}}^{(\ell)}}{T}\left\lfloor\frac{T\mathbf{z}^{(\ell)}}{V_{\mathrm{th}}^{(\ell)}}\right\rfloor && \text{is the step activation function of SNN}\,.
\end{aligned}$$

This Eq. (C.6) recovers the loss of the shift-threshold ReLU (with a shift value ϕ) and the step function, which is the same as Deng & Gu (2021). And as shown in (A) of Fig. 1, the conversion error is the shaded area. The error between the activation function (of ANNs) and the step function (of SNNs) is obtained by summing up of all the shaded area together, which is the ANN-SNN conversion error.

Then the objective becomes minimize

$$\min_{\phi} \{\mathbb{E}_{\mathbf{z}} |\mathrm{Err}_1|\} = \min_{\phi} \frac{T}{2} \left[\left(\frac{V_{\mathrm{th}}^{(\ell)}}{T} + \frac{V_{\mathrm{th}}^{(\ell)}}{T}\phi \right)^2 + \left(\frac{V_{\mathrm{th}}^{(\ell)}}{T}\phi \right)^2 \right] = \frac{(V_{\mathrm{th}}^{(\ell)})^2}{4T} \Longrightarrow \phi = -\frac{1}{2} \,.$$

Then

$$\delta_1 = \frac{-1/2 + \delta}{T} \,. \tag{C.7}$$

And the minimum L_2-norm of the first error becomes

$$\mathbb{E}_{\mathbf{z}} \left(|\mathrm{Err}_1| \right) = \frac{(V_{\mathrm{th}}^{(\ell)})^2}{4T} \,.$$

For Eq. (C.5), with $\mathbf{v}^{(\ell)}(0) = V_{\mathrm{th}}^{(\ell)}\boldsymbol{\delta}, V_{\mathrm{th}}^{(\ell)} = \theta^{(\ell)}$, we have

$$|\mathrm{Err}_2| \triangleq \left| \frac{\theta^{(\ell)}}{N} \left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta} \right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T} \left\lfloor \frac{T\mathbf{z}^{(\ell)} + \mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}} \right\rfloor \right|$$

From Lemma 1 and Theorem 1, we have

$$\begin{aligned} \mathbb{E}_{\mathbf{z}} \left(|\mathrm{Err}_2| \right) &= \mathbb{E}_{\mathbf{z}} \left(\left| \frac{\theta^{(\ell)}}{N} \left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta} \right\rfloor - \frac{V_{\mathrm{th}}^{(\ell)}}{T} \left\lfloor \frac{T\mathbf{z}^{(\ell)} + \mathbf{v}^{(\ell)}(0)}{V_{\mathrm{th}}^{(\ell)}} \right\rfloor \right| \right) \Big|_{\delta \in [-\frac{1}{2}, \frac{1}{2}]} \\ &= 0 \,. \end{aligned}$$

Then

$$\begin{aligned} &\mathbb{E}_{\mathbf{z}} \left(\left| \mathrm{Err}^{(\ell)} \right| \right) = \mathbb{E}_{\mathbf{z}} \left(\left| \mathcal{F}_{\mathrm{ANN}}(\mathbf{z}^{(\ell)}) - \mathcal{F}_{\mathrm{SNN}}(\mathbf{z}^{(\ell)}) \right| \right) = \mathbb{E}_{\mathbf{z}} \left(|c \cdot \mathrm{Err}_1 + (1-c) \cdot \mathrm{Err}_2| \right) \\ &\leqslant c \cdot \mathbb{E}_{\mathbf{z}} \left(|\mathrm{Err}_1| \right) + (1-c) \cdot \mathbb{E}_{\mathbf{z}} \left(|\mathrm{Err}_2| \right) \\ &= c \cdot \frac{(V_{\mathrm{th}}^{(\ell)})^2}{4T} + (1-c) \cdot 0 \\ &= \frac{c(V_{\mathrm{th}}^{(\ell)})^2}{4T} \,. \end{aligned}$$

This concludes the Theorem 2.

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}} \left(\left| \mathrm{Err}^{(\ell)} \right| \right) \Big|_{\delta \in [-\frac{1}{2}, \frac{1}{2}]} = \frac{c(V_{\mathrm{th}}^{(\ell)})^2}{4T} \,.$$

□

D PSEUDO-CODE FOR THE UNIFIED ANN-SNN CONVERSION ALGORITHM

Here is the pseudo-code for our proposed unified ANN-SNN conversion algorithm.

Algorithm 1: Algorithm for ANN-SNN conversion.

Input: ANN model structure $f_{\text{ANN}}(\mathbf{x};\mathbf{W})$ with initial weights $\mathbf{W}=\{\mathbf{W}^{(\ell)}\}$; quasi-latency N; Any shift value from the interval $\delta\in[-\frac{1}{2},\frac{1}{2}]$;Initial dynamic threshold $\boldsymbol{\theta}=\{\boldsymbol{\theta}^{(\ell)}\}$; Learning rate ϵ.
Output: SNN model $f_{\text{SNN}}(\mathbf{x};\mathbf{W})$
Data: Dataset D

```
for ℓ = 1 to f_ANN.layers do
    if is ReLU activation then
        Replace ReLU(x) by SlipReLU(x; N, θ^(ℓ))
    if is MaxPooling layer then
        Replace MaxPooling layer by AvgPooling layer
for e = 1 to epochs do
    for length of Dataset D do
        Sample minibach (x^0, y) from D for ℓ = 1 to f_ANN.layers do
            x^(ℓ) = SlipReLU(W^(ℓ) x^(ℓ−1); N, θ^(ℓ))
    Loss = CrossEntropy(x^(ℓ), y)
    for ℓ = 1 to f_ANN.layers do
        W^(ℓ) ← W^(ℓ) − ε ∂Loss/∂W^(ℓ)
        θ^(ℓ) ← θ^(ℓ) − ε ∂Loss/∂θ^(ℓ)
for ℓ = 1 to f_ANN.layers do
    f_SNN.W^(ℓ) ← f_ANN.W^(ℓ)
    f_SNN.V_th^(ℓ) ← f_ANN.θ^(ℓ)
    f_SNN.v^(ℓ)(0) ← f_SNN.V_th^(ℓ) × δ
Return f_SNN
```

E EXPERIMENTS DETAILS

E.1 NETWORK STRUCTURE AND TRAINING SETUPS

There are three steps in our proposed ANN-SNN conversion,
Step 1: Tailor the ANN;
Step 2: Train the tailored ANN;
Step 3: Convert the trained ANN to an SNN.

In the first step, we first replace max-pooling with average-pooling and then replace the ReLU activation with the proposed SlipReLU activation function. After training the tailored ANN, we copy all weights from the trained-tailored ANN to the converted SNN, and set the threshold $V_{\text{th}}^{(\ell)}$ in each layer of the converted SNN equal to the maximum activation value $\theta^{(\ell)}$ of the source ANN in the same layer. Besides, we set the initial membrane potential $\mathbf{v}^{(\ell)}(0)$ in converted SNN as $V_{\text{th}}^{(\ell)}\delta$ to match the optimal shift δ of the SlipReLU in the tailored source ANN, where the the optimal shift δ can be any value in the interval $\delta\in[-\frac{1}{2},\frac{1}{2}]$. Despite the common data normalization, we use some data pre-processing techniques. For CIFAR-10 datasets, we resize the images into 32×32. We use the Stochastic Gradient Descent (SGD) optimizer (Bottou, 2012) with a momentum parameter of 0.9. The initial learning rate is set to $\epsilon=0.1$ for CIFAR-10. We use constant input when evaluating the converted SNNs.

E.2 INTRODUCTION OF DATASET

CIFAR-10: The CIFAR-10 dataset (Krizhevsky & Hinton, 2009) consists of $60,000$ 32×32 color images in 10 classes of objects such as airplanes, cars, and birds, with $6,000$ images per class. There are $50,000$ samples in the training set and $10,000$ samples in the test set.

F FUTURE STUDY

Remark 2. *Our unified conversion framework exploits both the one-step conversion mechanism and the two-step conversion mechanism. The one-step conversion method uses a pre-trained source ANN, such as Li et al. (2021), however, the two-step conversion method needs to redesign the activation function of the ANN to get a tailored source ANN, train it and convert it to SNN, such as Deng & Gu (2021); Bu et al. (2021).*

Remark 3. *Usually, implicit variables of an optimization problem are variables which do not need to be optimized but are used to model feasibility conditions (Mehlitz & Benko, 2021), and they are often interpreted as explicit ones (Mehlitz & Benko, 2021), by using union of image sets associated with given set-valued mappings to make the implicit variables as explicit variables, which can be an interesting future work but not what we are interested in this paper.*

As mentioned in Sect. 3.1, the multi-step output feature of SNN implies that higher-latency output depend on the outputs of all previous time-steps, which can be explored through multi-task learning. Therefore, it is reasonable to use multi-task learning for ANN-SNN conversion where the different time-steps can be seen as different but related tasks.