A Unified Optimization Framework of ANN-SNN Conversion: Towards Optimal Mapping from Activation Values to Firing Rates

Haiyan Jiang [1] **Srinivas Anumasa** [1] **Giulia De Masi** [2 3] **Huan Xiong** [1 4] **Bin Gu** [1]

Abstract

Spiking Neural Networks (SNNs) have gained significant attention for their energy-efficient and fast-inference capabilities, but training SNNs from scratch can be challenging due to the discrete nature of spikes. One alternative method is to convert an Artificial Neural Network (ANN) into an SNN, known as ANN-SNN conversion. Currently, existing ANN-SNN conversion methods often involve redesigning the ANN with a new activation function, rather than utilizing the traditional ReLU, and converting it to an SNN. However, these methods do not take into account the potential performance loss between the regular ANN with ReLU and the tailored ANN. In this work, we propose a unified optimization framework for ANN-SNN conversion that considers both performance loss and conversion error. To achieve this, we introduce the ***SlipReLU*** activation function, which is a weighted sum of the threshold-ReLU and the step function. Theoretical analysis demonstrates that conversion error can be zero on a range of shift values $\delta \in [-0.5, 0.5]$ rather than a fixed shift term 0.5. We evaluate our SlipReLU method on CIFAR datasets, which shows that SlipReLU outperforms current ANN-SNN conversion methods and supervised training methods in terms of accuracy and latency. To the best of our knowledge, this is the first ANN-SNN conversion method that enables SNN inference using only 1 time step. Code is available at https://github.com/HaiyanJiang/SNN_Conversion_unified.

[1]Department of Machine Learning, Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE [2]ARRC, Technology Innovation Institute, Abu Dhabi, UAE [3]BioRobotics Institute, Sant'Anna School of Advanced Studies, Pisa, Italy [4]IASM, Harbin Institute of Technology, China. Correspondence to: Huan Xiong <huan.xiong.math@gmail.com>, Bin Gu <bin.gu@mbzuai.ac.ae>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

Spiking neural networks (SNNs) are biologically-inspired neural networks based on biologically plausible spiking neuron models to process real-time signals (Hodgkin & Huxley, 1952; Izhikevich, 2003). Due to the significant advantages of low power consumption and fast inference on neuromorphic hardware (Roy et al., 2019), SNNs are 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). In this model, each neuron in the SNN 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, which results in low energy consumption when implemented on specialized neuromorphic hardware. Another significant property of SNNs is the pseudo-simultaneity of their inputs and outputs 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 main methods for obtaining an SNN: (1) training an SNN from scratch (Wu et al., 2018; Neftci et al., 2019; Zenke & Vogels, 2021), and (2) ANN-SNN conversion (Cao et al., 2015; Diehl et al., 2015; Deng & Gu, 2021), i.e., converting an ANN to an SNN. Training from scratch uses a gradient-based supervised optimization method, such as back-propagation, treating SNNs as specialized ANNs. Due to the non-differentiability of the binary activation function in SNNs, surrogate gradients are usually used (Neftci et al., 2019). However, this method can only train SNNs on small to moderate-size datasets (Li et al., 2021). On the other hand, ANN-SNN conversion is an effective method for obtaining deep SNNs with comparable performance to ANNs on large-scale datasets. There are two main types of ANN-SNN conversion mechanisms: (1) one-step conversion, which converts the pre-trained ANN to an 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 an SNN, for example Cao et al. (2015); Deng & Gu (2021); Bu et al. (2021).

In this work, we investigate a two-step method for ANN-SNN conversion. This method involves redesigning the ANN by replacing the regular ReLU activation function with a new activation function, training the tailored ANN, and subsequently converting it to an SNN. A tailored ANN that deviates too much from the regular ANN will degrade its performance, resulting in a performance loss that will be inherited by 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 research. To achieve high-accuracy and low-latency SNNs (e.g., 1 or 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 that considers both the ANN performance loss and the conversion error simultaneously.

(2) We propose using the SlipReLU activation function in the tailored ANN to minimize the layer-wise conversion error while maintaining the performance of the tailored ANN as close as possible to that of the regular ANN.

(3) The SlipReLU method covers a family of activation functions that map activation values in source ANNs to firing rates in target SNNs. Many state-of-the-art optimal ANN-SNN conversion methods can be viewed as special cases of our proposed SlipReLU method.

(4) Through two theorems, we demonstrate that the expected ANN-SNN conversion error can theoretically be zero within a range of shift values $\delta \in [-0.5, 0.5]$, rather than a fixed shift term 0.5. Experimental results further validate the effectiveness of the proposed SlipReLU method.

2. Preliminaries

In this study, we investigate a classification problem on an image dataset denoted as $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, where each image $\boldsymbol{x}$ is associated with a ground-truth class label $\boldsymbol{y}$. Our goal is to train a neural network $f : \boldsymbol{x} \to f(\boldsymbol{x})$, which can take the form of an ANN or an SNN, by optimizing the standard cross-entropy (CE) loss. The CE loss is defined as $L_{\mathrm{CE}}(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{C} \boldsymbol{y}_i \log(\boldsymbol{p}_i)$, where $\boldsymbol{y}_i$ is the ground-truth label and $\boldsymbol{p}_i$ is the network prediction $\boldsymbol{p}_i = f(\boldsymbol{x}_i)$. For consistency, we use the notation f to represent the same shared infrastructures of the source ANN and the target SNN. Moreover, we use $\mathcal{F}_{\mathrm{ANN}}$ and $\mathcal{F}_{\mathrm{SNN}}$ to denote the activation functions employed in the ANN and SNN models, respectively. For the notations, refer to Table S5.

ANN Neuron Model. In a traditional ANN, the entire input vector is fed into the network at once, and it undergoes layer-by-layer processing through continuous activation functions to generate a single output value. The forward pass of analog neurons in ANNs can be 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, $\mathbf{W}^{(\ell)}$ is the weight matrix, and $\mathcal{F}_{\mathrm{ANN}}(\cdot)$ is the activation function of the ANN.

SNN Neuron Model. In contrast to ANNs, SNNs employ binary activations (i.e. spikes) in each layer. To compensate for the limited representation capacity of the binary activation, the time dimension, or latency, is introduced in SNNs. Inputs for the forward pass in SNNs are presented as streams of events and the forward pass is repeated for T time-steps in order to produce the final result.

In this study, we consider the Integrate-and-Fire (IF) neuron model (Cao et al., 2015; Bu et al., 2021; Deng & Gu, 2021) for SNNs. The forward propagation of PSP (postsynaptic potential) through layers in the target SNN is equivalent to the forward computation of the analog neurons in the source ANN. We then derive the forward propagation of PSP. At time-step t, the IF neuron in ℓ-th layer receives its binary input $\mathbf{x}^{(\ell-1)}(t)$ from the previous layer, and temporarily updates its membrane potential according to the equation

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

where $\mathbf{v}^{(\ell)}(t)$ is the membrane potential at time step t, $\mathbf{u}^{(\ell)}(t)$ is the temporary intermediate variable used to determine the update from $\mathbf{v}^{(\ell)}(t-1)$ to $\mathbf{v}^{(\ell)}(t)$. If the temporary intermediate potential $u_i^{(\ell)}(t)$ exceeds the membrane threshold $V_{\mathrm{th}}^{(\ell)}$, it will produce a spike output $s_i^{(\ell)}(t) = 1$. Otherwise, it will 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 different layers. The membrane potential is updated by the reset-by-subtraction mechanism (Rueckauer et al., 2017; Han et al., 2020), that is, the temporary membrane potential $u_i^{(\ell)}(t)$ is subtracted by the threshold value $V_{\mathrm{th}}^{(\ell)}$ if the neuron fires $s_i^{(\ell)}(t) = 1$,

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

If the neuron in the current ℓ-th layer generates a spike, it will transmit an unweighted PSP $\mathbf{x}^{(\ell)}(t)$ as input to the succeeding layer, which is similar to Deng & Gu (2021),

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

As for the input to the first layer and the output of the last layer of the SNN, we do not employ any spiking mechanism as in Li et al. (2021). We directly encode the static image to temporal dynamic spikes as input to the first layer, which can prevent undesired information loss introduced by the Poisson encoding. For the last layer output, we only integrate the pre-synaptic input and do not fire any spikes.

3. Unified Optimization Framework of ANN-SNN Conversion

In this section, we propose a unified optimization framework for ANN-SNN conversion, together with the conversion error analysis. Our unified framework addresses the trade-off between the performance of the converted SNN and the deviation introduced by the tailored ANN with a new activation function and the regular ANN with ReLU activation.

The performance of the converted SNN is determined by both the source ANN performance and the conversion error. Previous methods for ANN-SNN conversion have focused solely on minimizing the conversion error without considering the performance of the tailored ANN (Cao et al., 2015; Diehl et al., 2015; Deng & Gu, 2021). Our approach, however, takes into account the performance of the tailored ANN, as well as the conversion error, in a two-step process. First, we design a new activation function for the source ANN to create a tailored ANN. Then, we train the tailored ANN and convert it to an SNN. By considering the performance loss between the tailored ANN and the regular ANN, our framework ensures that the new activation function does not deviate too far from the regular ReLU.

3.1. ANN-SNN Conversion in a Unified Framework

We define a unified optimization framework for the conversion of ANNs to SNNs.

Definition 1 (Unified Optimization Framework of ANN-SNN Conversion)**.** *The framework is formulated as an optimization problem with an implicit variable T,*

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

where $w \in [0,1]$. Specifically, when the ANN $\mathcal{F}_{\mathrm{ANN}}$ is designed with consideration of the deviation from the regular ReLU, the layer-wise conversion error $\mathbb{E}(|\mathrm{Err}^{(\ell)}|)$ becomes

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

The same neural network infrastructure is used for both the source ANN and the target SNN, as described in Sect. 2. The notation are as follows: $\mathcal{F}_{\mathrm{ReLU}}(\cdot)$ denotes the regular ANN with ReLU activation, $\mathcal{F}_{\mathrm{ANN}}(\cdot)$ is the tailored ANN with a new activation function, $\mathcal{F}_{\mathrm{SNN}}(\cdot)$ is the converted SNN, $\mathbf{z}$ is the input to the neural network, $\mathbf{W} = \{\mathbf{W}^{(\ell)}\}$ are the weight matrices 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. Additionally, T allows for flexibility in balancing the latency and the accuracy of the converted SNN for different applications.

Before proceeding, it is important to note that the unified framework defined in Definition 1 provides guidance for researchers to propose new solutions for optimal ANN-SNN conversion. Specifically, the following Remark 1 highlights several key points to consider.

Remark 1. *(A) the tailored ANN's activation function, $\mathcal{F}_{\mathrm{ANN}}$, should be designed to address the potential performance loss caused by the deviation from the ANN with regular ReLU activation. (B) When $\mathcal{F}_{\mathrm{ANN}}$ is designed by considering the deviation from the regular ReLU, the layer-wise error in Eq. (6) may arise from any mismatch of the following 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) An "optimal" ANN-SNN conversion is achieved when the conversion error, $\mathbb{E}_{\mathbf{z}}(|\mathrm{Err}^{(\ell)}|)$, reaches its minimum. For example, Deng & Gu (2021) has achieved an optimal minimum error of $\frac{(V_{\mathrm{th}}^{(\ell)})^2}{4T}$, while Bu et al. (2021) has theoretically achieved an optimal minimum error of* 0.

3.2. ANN-SNN Conversion Error Analysis

In the following, we will address the three potential errors that can occur during the conversion of an ANN to an SNN.

Firing Rates in SNNs and Activation Values in ANNs. One such error relates to the difference between firing rates in SNNs and activation values in ANNs. In SNNs, the activation value of an IF neuron is defined as the average post-synaptic potential (i.e. average PSP), denoted as $\bar{\mathbf{x}}^{(\ell)}$. The firing rate is represented by the average number of spikes over a given time period (latency) T, denoted as $\bar{\mathbf{s}}^{(\ell)}$. The firing rate and average PSP may be used interchangeably in SNNs, but in this paper, they are defined differently,

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

To minimize the layer-wise error during conversion, it is ideal for the converted SNN to have activation values that are similar to those of the source ANN for each layer. This can be represented mathematically as

$$\boldsymbol{a}^{(\ell)} \approx \bar{\mathbf{x}}^{(\ell)},$$

where $\boldsymbol{a}^{(\ell)}$ is the activation value of the ANN and $\bar{\mathbf{x}}^{(\ell)}$ is the activation value of the SNN. The term $\bar{\mathbf{x}}^{(\ell)}$ represents the average PSP released by the ℓ-th layer, which serves as the input to the succeeding layer. The threshold $V_{\text{th}}^{(\ell)}$ in SNN can differ across layers. Therefore, we treat it as 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 errors.

Activation Function in SNNs. In terms of the activation function in SNNs, it defines the relationship between activation values, $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$, of successive layers. We utilize the derivation presented in previous studies (Deng & Gu, 2021; Li et al., 2021), to deduce the SNN activation function, $\mathcal{F}_{\text{SNN}}$. By combining Eq. (2) and Eq. (4), and summing over the time-step from 1 to T, then we get

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

Due to the spike-in-spike-out property of the IF neurons in SNN, the output spikes at each time-step can be either 0 or 1. The accumulated spikes are represented by $\mathbf{m} = \sum_{t=1}^{T}\mathbf{s}^{(\ell)}(t) = \{m_i\}$, where each $m_i \in \{0, 1, 2, \cdots, T\}$ denotes the total number of spikes of neuron i. Further we assume the terminal membrane potential $\mathbf{v}^{(\ell)}(T)$ should be within the range $[\mathbf{0}, \mathbf{V}_{\text{th}}^{(\ell)})$. Therefore, with a shift value $\boldsymbol{\delta}$, we have

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

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

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

The $\text{clip}\,(x, a, b)$ function sets the lower bound a and upper bound b, while the floor function $\lfloor x \rfloor$ gives the greatest integer that is less than or equal to x. With $\bar{\mathbf{x}}^{(\ell)} = V_{\text{th}}^{(\ell)}\bar{\mathbf{s}}^{(\ell)} = \mathbf{m}V_{\text{th}}^{(\ell)}/T$, the SNN activation function gives the relationship between activation values $\bar{\mathbf{x}}^{(\ell-1)}$ and $\bar{\mathbf{x}}^{(\ell)}$ as follows,

$$\begin{aligned}\bar{\mathbf{x}}^{(\ell)} &= \mathcal{F}_{\text{SNN}}\left(\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)}\right) \\ &= V_{\text{th}}^{(\ell)}\text{clip}\left(\frac{1}{T}\left\lfloor \frac{T\mathbf{W}^{(\ell)}\bar{\mathbf{x}}^{(\ell-1)}}{V_{\text{th}}^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, 1\right) . \quad (7)\end{aligned}$$

The activation function $\mathcal{F}_{\text{SNN}}(\cdot)$ in SNNs is a step function defined within the interval $[0, V_{\text{th}}^{(\ell)}]$ with a step size of $\frac{V_{\text{th}}^{(\ell)}}{T}$ (i.e. the green curve in Fig. 1). Since the SNN output is discrete while the ANN output is continuous, there is an intrinsic difference between $\boldsymbol{a}^{(\ell)}$ and $\bar{\mathbf{x}}^{(\ell)}$, as illustrated in Fig. 1. A comprehensive analysis of the SNN activation function is provided in Appendix A.

4. Proposed SlipReLU

Through the above analysis, the performance of the converted SNN is usually determined by the source ANN performance and the conversion error. From Remark 1, the layer-wise conversion error in Eq. (6) can be affected by the difference between activation values from source ANNs and target SNNs, the difference between activation functions, and the latency T (i.e., time-step). The goal of ANN-SNN conversion is to minimize the conversion error with low latency T while maintaining the performance of the tailored ANN. Recently, many research works try to minimize the gap between activation functions. For example, Deng & Gu (2021) uses the shift-threshold-ReLU as the activation function in the source ANN, and Bu et al. (2021) employes the quantization clip-floor-shift (QCFS) activation in source ANN instead of regular ReLU activation.

4.1. The SlipReLU Activation Function

In this section, by following our unified optimization framework, we will exploit the two-step conversion mechanism. The process involves redesigning the ANN with a new activation function to get a tailored ANN, training the tailored ANN, and then converting it to an SNN by copying the weights from the tailored ANN to the target SNN.

However, a performance loss will occur if the new activation function of the tailored ANN deviates too much from the regular ReLU activation function. Therefore, it is crucial to minimize the conversion error while keeping the deviation from the regular ReLU activation function minimal. To achieve this, we propose the SlipReLU activation function, which is a weighted sum of the threshold-ReLU and the step function (i.e., SNN activataion function). This allows for a balance between the regular ReLU and the step function. We assume that both the ANN and the SNN receive the same input from the previous layer,

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

Proposed SlipReLU Activation Function. Following the unified optimization framework outlined in Sect. 3.1, our proposed SlipReLU activation function aims to minimize the mismatch between the tailored ANN and the target SNN, while also minimizing deviation from the regular ReLU,

$$\begin{aligned}\text{SlipReLU}(\mathbf{z}^{(\ell)}) &= c\theta^{(\ell)}\text{clip}\left(\frac{\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta}_1, 0, 1\right) \\ &+ (1-c)\theta^{(\ell)}\text{clip}\left(\frac{1}{N}\left\lfloor \frac{N\mathbf{z}^{(\ell)}}{\theta^{(\ell)}} + \boldsymbol{\delta} \right\rfloor, 0, 1\right) .\end{aligned} \quad (8)$$

The SlipReLU activation function is a weighted sum of the threshold-ReLU and the step function, with the slope $0 \leqslant c \leqslant 1$ balancing the weight between the two. This is illustrated in the red curves of (C1)-(C3) in Fig. 1.

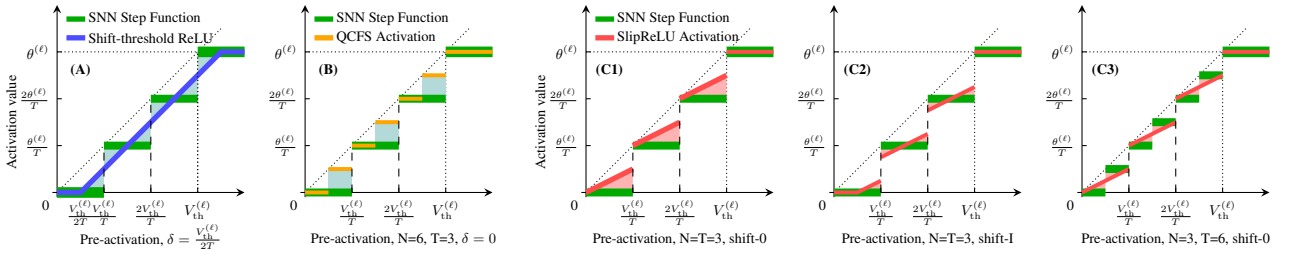


Figure 1. The activation functions of source ANNs and the activation function of target SNNs. The figure shows three different types of activation functions for source ANNs: (A) shift-threshold-ReLU (blue curve) from Deng & Gu (2021), (B) quantization clip-floor-shift (QCFS) activation (orange curve) from Bu et al. (2021), and (C1)-(C3) the proposed SlipReLU activation (red curve). The activation function of SNNs is the step function represented by a green curve. The error between the activation function of ANNs and the step function of SNNs is the sum of all the shaded area together, which is referred to as the ANN-SNN conversion error.

The SlipReLU activation function utilizes parameters N and $\theta^{(\ell)}$, whereas in contrast, the SNN activation function employs T and $V_{\text{th}}^{(\ell)}$ ($T \leftrightarrow N$, $V_{\text{th}}^{(\ell)} \leftrightarrow \theta^{(\ell)}$). Instead of using the inherent property of the SNN, the latency T, we use the quasi-time-step (quasi-latency) N in the ANN. Additionally, the threshold value $V_{\text{th}}^{(\ell)}$ in the SNN is replaced by the trainable value $\theta^{(\ell)}$ in the ANN, which can be learned and copied to the target SNN.

To gain some insights into the proposed SlipReLU, we set $\boldsymbol{\delta}_1 = \boldsymbol{\delta} = 0$ for simplicity. With some linear algebra, the SlipReLU can be reformulated as a piece-wise linear function with a constant slope c,

$$\text{SlipReLU}(\mathbf{z}^{(\ell)}) = c\mathbf{z}^{(\ell)} + (1-c)\frac{k\boldsymbol{\theta}^{(\ell)}}{N},\ k = 0, 1, \cdots, N-1.$$

Here $\frac{k\boldsymbol{\theta}^{(\ell)}}{N} \leqslant \mathbf{z}^{(\ell)} < \frac{(k+1)\boldsymbol{\theta}^{(\ell)}}{N}$, and c ($0 \leqslant c \leqslant 1$) is the constant slope of the piece-wise linear function. The constant slope c effectively balances the contributions of the threshold-ReLU and the step function, resulting in a function that resembles a slippery step function with a slope, hence the name "SlipReLU", as illustrated in (C1)-(C3) in Fig. 1. A detailed derivation can be found in Appendix B.

Special Cases of SlipReLU. The proposed SlipReLU activation function encompasses several special cases that fit within the unified optimization framework.

(1) When $c = 0, \boldsymbol{\delta} = [\frac{1}{2}]$, the proposed SlipRLU becomes the quantization-clip-floor-shift (QCFS) in Bu et al. (2021). This case only focuses on being close to the step function of the target SNN, but neglects the deviation from the regular ReLU. (2) When $c = 1, \boldsymbol{\delta}_1 = [-\frac{1}{2N}]$, the proposed SlipRLU becomes the shift-threshold ReLU in Deng & Gu (2021). While this case accounts for the deviation from the regular ReLU, it overlooks the proximity to the step function of the target SNN. In contrast, our proposed SlipReLU balances the trade-off between the regular ReLU and the step function of the target SNN. For further details, refer to Appendix B.

4.2. Theorems on the Conversion Error

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

Theorem 1. *Consider an ANN trained with SlipReLU activation function as defined in Eq. (8), and its conversion to an SNN with the same weights. Let* $V_{\text{th}}^{(\ell)} = \theta^{(\ell)}$, $\mathbf{v}^{(\ell)}(0) = V_{\text{th}}^{(\ell)}\boldsymbol{\delta}$, *and* $c = 0$. *Then, for any arbitrary values of* T *and* N, *the expected conversion error of the proposed unified method reaches* 0, *i.e.,*

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\text{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \mathbf{0}\,, \tag{9}$$

provided that the shift term δ *satisfies* $\delta \in [-\frac{1}{2}, \frac{1}{2}]$.

Theorem 1 indicates that when $c = 0$, the expectation of the conversion error reaches zero, even though $N \neq T$, as long as the shift term δ satisfies $\delta \in [-\frac{1}{2}, \frac{1}{2}]$. The proof can be found in Appendix C.

Theorem 2. *Consider an ANN trained with SlipReLU activation function as defined in Eq. (8), and its conversion to an SNN with the same weights. Let* $V_{\text{th}}^{(\ell)} = \theta^{(\ell)}$, $\mathbf{v}^{(\ell)}(0) = V_{\text{th}}^{(\ell)}\boldsymbol{\delta}$, *and* $\boldsymbol{\delta}_1 = [\frac{\delta-1/2}{T}]$. *Then, for arbitrary values of* 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.,*

$$\forall\, T, L \quad \mathbb{E}_{\mathbf{z}}\left(\left|\text{Err}^{(\ell)}\right|\right)\Big|_{\delta\in[-\frac{1}{2},\frac{1}{2}]} = \frac{c(V_{\text{th}}^{(\ell)})^2}{4T}\,, \tag{10}$$

as long as the shift term δ *satisfies* $\delta \in [-\frac{1}{2}, \frac{1}{2}]$.

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

4.3. Algorithm for Training ANN with SlipReLU

Training ANNs with SlipReLU activation function through backpropagation can be a challenging task. Despite the SlipReLU having a constant slope as its derivative, as shown in Eq. (8), small slope values $c \in [0, 1]$ can result in the gradient vanishing problem. To overcome this, we draw inspiration from the works of Bu et al. (2021); Bengio et al. (2013), and utilize the surrogate gradient as the derivative of the floor function, with $\frac{d\lfloor x \rfloor}{x} = 1$. The overall rule for derivation is as follows, $\frac{d\mathcal{F}_{\text{ANN}}(\mathbf{z}^{(\ell)})}{dz_i^{(\ell)}} = 1$ if $\mathrm{z}_i^{(\ell)} \in D_1 \cup D_2$ and 0 otherwise, 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 the Stochastic Gradient Descent algorithm, and convert it to an SNN. Please refer to Appendix D for our proposed ANN-SNN conversion algorithm.

4.4. Criterion to Choose Best Hyper-parameters (N, c)

In accordance with the criterion of the unified optimization framework of ANN-SNN conversion outlined in Eq. (5), we determine the optimal hyper-parameters (N, c) by minimizing the following criterion measure,

$$\text{Crit.} = \frac{1}{2}\left|Acc_{\text{ReLU}} - Acc_{\text{ANN}}\right| + \frac{1}{2}\left|Acc_{\text{ANN}} - Acc_{\text{SNN}}\right|.$$

5. Related Work

The study of ANN-SNN conversion was first proposed by Cao et al. (2015), which focused on converting ANNs with the ReLU activation function to SNNs. Subsequently, Diehl et al. (2015) proposed data-based and model-based weight-normalization methods to convert a three-layer CNN to an SNN. However, due to the error analyzed in Sect. 3.1, the converted SNN typically requires hundreds of time-steps to achieve accurate results. To address this issue, the "reset-by-subtraction" mechanism (Rueckauer et al., 2017), also known as "soft-reset" mechanism (Han et al., 2020), was proposed an alternative to the "reset-to-zero" method. Recently, many methods and algorithms have been proposed to eliminate conversion errors, such as the weight-normalization technique proposed by Sengupta et al. (2019) which takes into account the actual SNN operations during the conversion process. 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 fine-tuning to correct errors layer-by-layer. Our work is similar to that of Deng & Gu (2021); Bu et al. (2021), which also focus on optimal conversion. Deng & Gu (2021) minimized the layer-wise error by a shift-threshold ReLU which only considers the deviation from the standard ReLU in the unified optimization framework in Sect. 3.1. Bu et al. (2021) proposed using a quantization clip-floor-shift activation function to train ANNs, which only minimizes the conversion error, but neglects the performance loss of the tailored ANN with new activation function. They both achieved "optimal" results with some fixed shift term. In contrast, our proposed unified framework offers more flexibility for different application scenarios when converting ANNs to SNNs by using techniques that eliminate the conversion error while preserving the ANN performance with less deviation from the standard ANN with regular ReLU. Our SlipReLU is able to balance the trade-off between the ANN performance and the conversion error simultaneously.

6. Experiments

In this section, we compare our SlipReLU method with existing state-of-the-art approaches for image classification tasks on CIFAR-10 (LeCun et al., 1998) and CIFAR-100 (Krizhevsky & Hinton, 2009) datasets. Similar to previous works, we use the VGG-16, ResNet-18, and ResNet-20 network structures as the source ANNs. We compare our method with the state-of-the-art ANN-SNN conversion methods and supervised training methods, including Hybrid-Conversion (HC) (Rathi et al., 2020), RNL (Ding et al., 2021), ReLU-Threshold-Shift (RTS) (Deng & Gu, 2021), RMP (Han et al., 2020), TSC (Han & Roy, 2020), SNN Conversion with Advanced Pipeline (SNNC-AP) (Li et al., 2021), and the ANN-SNN conversion with Quantization Clip-Floor-Shift activation function (QCFS) (Bu et al., 2021). Refer to Appendix E for the network structures and training setups. We use SlipReLU with shift settings $\delta_1 = 0, \delta = 0.5$, and refer to Appendix F for ablation studies of SlipReLU with/without shifts.

6.1. Comparison with SOTA Conversion Methods

Table 1 shows the performance comparison of the proposed SlipReLU with the state-of-the-art ANN-SNN conversion methods on CIFAR-10. Notably, **the proposed SlipReLU method is the only existing work that enables SNN inference using only one time-step.** Specially, when latency $T = 1$, the SlipReLU method is able to achieve an accuracy of 93.11% for ResNet-18 with settings $(N, c) = (1, 0.4)$, and an accuracy of 88.17% for VGG-16 with the SlipReLU activation function. For ultra-low latency inference at $T = 2$, the proposed SlipReLU method has the best performance of 93.97% compared to existing state-of-the-art ANN-SNN conversion methods for ResNet-18, with a significant margin compared to the next best baseline QCFS of 75.44%. The accuracy for VGG-16 is 89.57% with SlipReLU activation, which is slightly worse than QCFS. For ResNet-20, an accuracy of 82.25% is achieved with 2 time-steps, which is also the best. In conclusion, the proposed SlipReLU method provides the best SNN accuracy

Table 1. Comparison between the proposed SlipReLU method and previous works on CIFAR10.

Architecture	Method	ANN Acc.	T=1	T=2	T=4	T=8	T=16	T=32	T=64	T⩾256
VGG-16	RMP (Han et al., 2020)	93.63	-	-	-	-	-	60.30	90.35	93.39
	RTS (Deng & Gu, 2021)	92.09	-	-	-	-	92.29	92.29	92.22	92.26
	RNL (Ding et al., 2021)	92.82	-	-	-	-	57.90	85.40	91.15	92.95
	SNNC-AP (Li et al., 2021)	95.72	-	-	-	-	-	93.71	95.14	95.79
	QCFS (Bu et al., 2021)	95.52	-	91.18	93.96	94.95	95.40	95.54	95.55	95.59
	ReLU	95.92	10.00	10.00	11.51	70.97	88.39	93.05	94.76	95.19
	SlipReLU (N=2, c=0.2)	**93.02**	**88.17**	**89.57**	**91.08**	**92.26**	**92.96**	**93.19**	**93.25**	**93.25**
	SlipReLU (N=4, c=0.9)	*95.60*	*11.37*	*75.18*	*88.80*	*93.54*	*95.20*	*95.66*	*95.65*	*95.66*
ResNet-20	RMP (Han et al., 2020)	91.47	-	-	-	-	-	-	-	91.36
	QCFS (Bu et al., 2021)	91.77	-	73.20	83.75	89.55	91.62	92.24	92.35	92.41
	ReLU	93.71	11.58	12.54	16.05	36.47	70.84	83.47	85.93	86.46
	SlipReLU (N=1, c=0.2)	**82.07**	**80.99**	**82.25**	**83.52**	**84.46**	**84.70**	**84.85**	**84.89**	**84.69**
	SlipReLU (N=4, c=0.6)	*92.96*	*45.87*	*57.82*	*73.17*	*86.66*	*92.13*	*93.23*	*93.36*	*93.29*
ResNet-18	RTS (Deng & Gu, 2021)	92.32	-	-	-	-	92.41	93.30	93.55	93.58
	SNNC-AP (Li et al., 2021)	95.46	-	-	-	-	-	94.78	95.30	95.45
	QCFS (Bu et al., 2021)	96.04	-	75.44	90.43	94.82	95.92	96.08	96.06	96.06
	ReLU	96.71	11.00	25.07	55.21	73.80	88.44	94.50	96.00	96.50
	SlipReLU (N=1, c=0.4)	**94.61**	**93.11**	**93.97**	**94.59**	**94.92**	**95.18**	**95.07**	**94.81**	**94.67**
	SlipReLU (N=4, c=0.3)	*96.15*	*86.52*	*90.78*	*93.84*	*95.48*	*96.10*	*96.12*	*96.22*	*96.15*

for ultra-low latency inference, particularly for $T = 1$.

As there is an intrinsic trade-off between latency and accuracy in SNN models, the SlipReLU method, which has high accuracy at low-latency, may suffer from a performance degradation when used for inference at larger latency. For example, as shown in Table 1 and Table 2, a converted SNN that has the highest accuracy at $T = 1$ may perform worse at $T = 16$ than another converted SNN. This illustrates the phenomenon that one converted SNN model cannot perform better than others for both low-latency and long-latency inference. Therefore, we present two converted SNN models, for **low-latency (T⩽8)** and *long-latency (T⩾16)* inference respectively, corresponding to **results in bold** and *results in italics* in tables. As we focus on ultra-low-latency inference of SNNs in this paper, *results in italics* can be considered as additional information showcasing the performance of the proposed SlipReLU method for long-latency inference.

We further evaluate the performance of SlipReLU method on the large-scale CIFAR-100 dataset and present the results in Table 2. Notably, when $T = 1$, our SlipReLU method is able to achieve an accuracy of 71.51% for ResNet-18 and an accuracy of 64.21% for VGG-16, while all the other methods fail to provide inference accuracy for this latency. Additionally, when $T = 2$, our SlipReLU method achieves an accuracy of 73.91% for VGG16, which is 3.12% higher than the next best QCFS method. These results demonstrate that our SlipReLU method outperforms the previous conversion methods in both accuracy and ultra-low latency.

We also investigate the conversion of ReLU-ANN to SNN and results are shown in Table 1 and Table 2. From the results, we can see that the ReLU is better in terms of the ANN accuracy, but for the SNN accuracy, ReLU completely fails to deliver satisfactory results for low latency (e.g., $T = 1, 2, 4$). Compared with ReLU, our proposed SlipReLU model shows a slight performance drop in ANN accuracy, but significantly outperforms ReLU in terms of SNN accuracy.

6.2. Comparison with Supervised Training Methods

Table 3 reports the results of the proposed SlipReLU method against the state-of-the-art supervised training methods on CIFAR10 dataset. These state-of-the-art supervised training methods include Hybrid-Conversion (HC) (Rathi et al., 2020), TSSL (Zhang & Li, 2020), tdBN (Zheng et al., 2021), TET (Deng et al., 2021), NAS (Kim et al., 2022) and NA (Yang et al., 2021). Our approach for ResNet-18 achieves an accuracy of 93.11% with time-step $T = 1$. The CIFARNet achieves an accuracy of 95.31% with time-step $T = 4$, which is higher than any other supervised trained models. The TET and tdBN methods can achieve comparable accuracy, but they use a more complex ResNet-19, whereas our SlipReLU uses ResNet-18. Notably, our ultra-low latency performance is comparable to other state-of-the-art supervised training methods.

6.3. Effect of the Slope c and the Quasi-Latency N

In our SlipReLU method, the slope parameter c balances the weight of the threshold ReLU and the step function, which ultimately affects the accuracy of the converted SNN. To better understand the effect of c on the SNN performance and determine the optimal value, we have conducted experiments on the CIFAR-10 and CIFAR-100 using VGG-16, ResNet-18 and ResNet-20 networks with quasi-latency $N = 2$ and $N = 32$. The results in Fig. 2 illustrate the impact of the slope c on the converted SNN accuracy for different quasi-latency N at different time-step/latency T.

Table 2. Comparison between the proposed SlipReLU method and previous works on CIFAR-100.

Architecture	Method	ANN Acc.	T=1	T=2	T=4	T=8	T=16	T=32	T=64	T⩾512
VGG-16	TSC (Han & Roy, 2020)	71.22	-	-	-	-	-	-	-	70.97
	RMP (Han et al., 2020)	71.22	-	-	-	-	-	-	-	70.93
	RTS (Deng & Gu, 2021)	70.62	-	-	-	-	65.94	69.8	70.35	70.55
	SNNC-AP (Li et al., 2021)	77.89	-	-	-	-	-	73.55	76.64	77.87
	QCFS (Bu et al., 2021)	76.28	-	63.79	69.62	73.97	76.24	77.01	77.10	77.08
	ReLU	73.39	1.00	1.53	15.55	28.56	46.03	62.42	70.05	72.16
	SlipReLU (N=1, c=0.4)	**68.46**	**64.21**	**66.30**	**67.97**	**69.31**	**70.09**	**70.19**	**70.05**	**69.79**
	SlipReLU (N=2, c=0.1)	*70.03*	*54.68*	*58.66*	*62.56*	*66.31*	*69.35*	*70.65*	*71.23*	*71.52*
ResNet-20	TSC (Han & Roy, 2020)	68.72	-	-	-	-	-	-	-	68.18
	RMP (Han et al., 2020)	68.72	-	-	-	-	-	27.64	46.91	67.82
	QCFS (Bu et al., 2021)	69.94	-	19.96	34.14	55.37	67.33	69.82	70.49	70.50
	ReLU	70.18	1.28	1.16	1.76	2.91	4.03	6.17	8.95	11.66
	SlipReLU (N=1, c=0.2)	**50.79**	**48.12**	**51.35**	**53.27**	**54.17**	**53.91**	**53.11**	**51.75**	**50.89**
	SlipReLU (N=4, c=0.4)	*68.40*	*16.52*	*23.79*	*37.94*	*57.20*	*66.61*	*68.76*	*69.04*	*69.09*
ResNet-18	RTS (Deng & Gu, 2021)	67.08	-	-	-	-	63.73	68.40	69.27	69.82
	SNNC-AP (Li et al., 2021)	77.16	-	-	-	-	-	76.32	77.29	77.25
	QCFS (Bu et al., 2021)	78.80	-	70.79	75.67	78.48	79.48	79.62	79.54	79.61
	ReLU	77.16	1.00	1.64	4.99	11.40	34.08	60.44	71.90	75.63
	SlipReLU (N=1, c=0.3)	**74.01**	**71.51**	**73.91**	**74.89**	**75.40**	**75.41**	**75.30**	**74.98**	**74.90**
	SlipReLU (N=2, c=0.5)	*77.08*	*51.01*	*60.03*	*68.72*	*74.59*	*77.29*	*78.04*	*77.97*	*77.99*

Table 3. Comparison with state-of-the-art supervised training methods on CIFAR-10 dataset.

Model	Method	Arch.	SNN	T
HC (Rathi et al., 2020)	Hybrid	VGG-16	91.13	100
TSSL (Zhang & Li, 2020)	Backprop	CIFARNet	91.41	5
tdBN (Zheng et al., 2021)	Backprop	ResNet-19	92.34	2
TET (Deng et al., 2021)	Backprop	ResNet-19	94.16	2
NAS (Kim et al., 2022)	Search	SNASNet	93.73	5
NA (Yang et al., 2021)	Backprop	AlexNet	91.76	5
SlipReLU (Ours)	ANN-SNN	AlexNet	94.94	5
		VGG-16	91.08	4
		VGG-16	88.17	1
		ResNet-18	93.11	1
		ResNet-18	93.97	2
		CIFARNet	95.31	4

It can be observed from Fig. 2 that for small values of quasi-latency N, the slope c has a significant effect on SNN accuracy for ultra-low and low-latency inference. Particularly, different slope values c can result in varying SNN accuracy levels when the time-step T is small. However, for larger values of N, the effect of c on the SNN accuracy is less pronounced, with all curves appearing similar, regardless of the value of T. This flexibility allows our SlipReLU method to be applied to different scenarios, with small values of N being preferred for ultra-low/low-latency inference and larger values of N being used when the inference time is not a concern. Further details can be found in Appendix G.

As the proposed SlipReLU method has two hyper-parameters, the slope c and the quasi-latency N, we use the criterion measure in Sect. 4.4 to select the optimal values. For instance, based on the criterion measure in Sect. 4.4, the optimal hyper-parameter combination for VGG-16 on CIFAR-10 is $(N, c) = (2, 0.2)$. We then convert this optimal single ANN to an SNN and obtain the SNN accuracy at different time-steps T. Refer to Appendix H for more detailed results of selecting the optimal hyper-parameters.

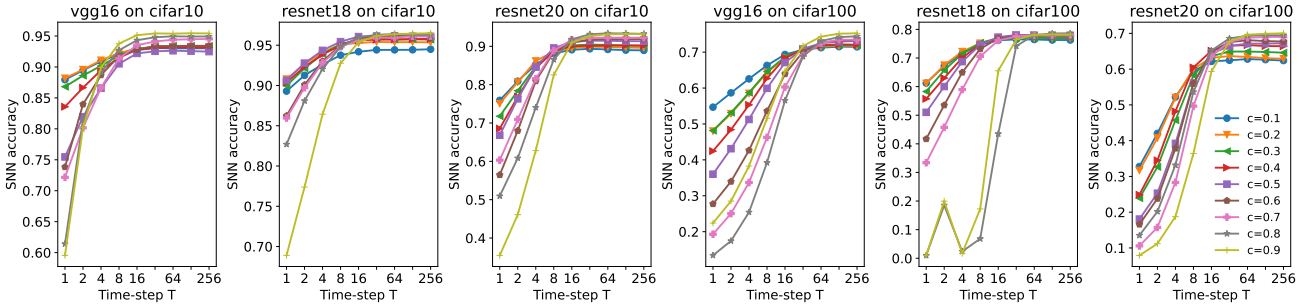
We also test our proposed method on the large-scale ImageNet dataset, and results are reported in Table 4 with one specified setting (N, c). As it is more challenging and expensive to conduct experiments on large-scale dataset such as ImageNet, we did not fine-tune the model with the best chosen hyper-parameters for ImageNet. We use the specified setting $(N, c) = (8, 0.2)$ for ResNet-34, the proposed SlipReLU method is only better than other conversion method when $T \geqslant 64$ with this specific setting. For VGG-16 model, we employ the specified setting $(N, c) = (8, 0.1)$, and we can achieve an accuracy of 51.54% when the time-step is 16. Our proposed method does not always perform well with specified (N, c) settings, which shows the importance of the selection of slope c and quasi-latency N.

6.4. Future Work

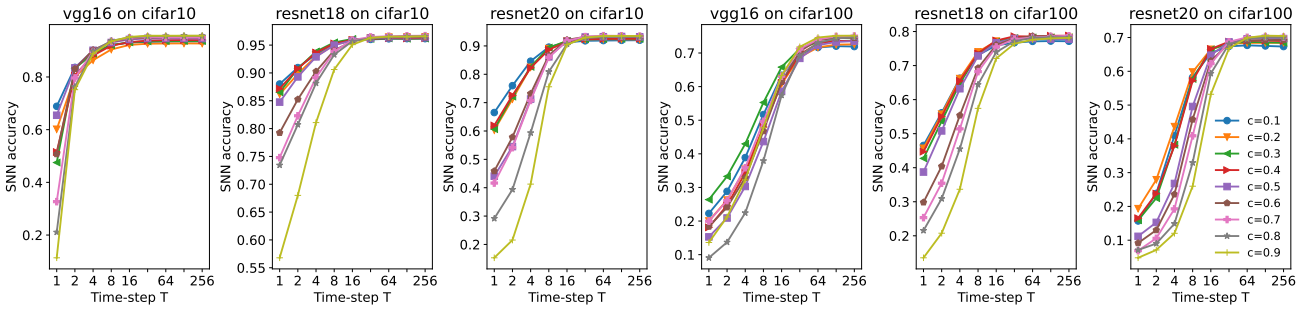
For future work, we propose to explore the learning of the slope c and the quasi-latency N during ANN training, rather than treating them as hyper-tuning parameters. By doing so, we aim to identify the optimal combination of (N, c) without the need for repetitive training, thereby enhancing the efficiency of the proposed method.

7. Discussion and Conclusion

In this work, we propose a unified framework for converting ANNs to SNNs that addresses a limitation of existing methods. Specifically, we take into account the performance loss that occurs when replacing the regular ReLU activation



(a) Influence of different slopes with the quasi-latency $N = 2$ on CIFAR-10 and CIFAR-100



(b) Influence of different slopes with the quasi-latency $N = 4$ on CIFAR-10 and CIFAR-100

Figure 2. Effect of different slopes c with different quasi-latency N on CIFAR-10 and CIFAR-100.

Table 4. Comparison between our proposed SlipReLU method and other conversion methods on ImageNet dataset.

Architecture	Method	ANN	T=16	T=32	T=64	T=128
ResNet-34	SNNC-AP (Li et al., 2021)	**75.66**	-	64.54	71.12	73.45
	QCFS (Bu et al., 2021)	74.32	**59.35**	**69.37**	72.35	73.15
	SlipReLU (N=8, c=0.2)	75.08	43.76	66.61	**72.71**	**74.01**
VGG-16	RTS (Deng & Gu, 2021)	-	39.42	69.11	70.21	70.45
	SNNC-AP (Li et al., 2021)	**75.36**	-	63.64	70.69	73.32
	QCFS (Bu et al., 2021)	74.29	50.97	**68.47**	**72.85**	**73.97**
	SlipReLU (N=8, c=0.1)	71.99	**51.54**	67.48	71.25	72.02

function in an ANN with a new activation function. This performance loss is then inherited by the resulting SNN. To address this issue, we formulate the ANN-SNN conversion as a unified optimization problem that considers both the performance loss and the conversion error. To this end, we introduce the SlipReLU activation function, which is a weighted combination of the threshold-ReLU and the step function, and improves the performance of either function alone. This allows for more accurate conversion of ANNs to SNNs.

The SlipReLU method covers a family of activation functions that map from activation values in source ANNs to firing rates in target SNNs. Most existing 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 $\frac{1}{2}$, allowing for converted SNNs with high accuracy and ultra-low latency. We have evaluated our proposed SlipReLU method on the CIFAR-10/100 datasets, and the results show that our proposed SlipReLU method outperforms the state-of-the-art ANN-SNN conversion methods and supervised training methods in terms of accuracy and latency. To the best of our knowledge, this is the first ANN-SNN conversion method that enables SNN inference using only one time-step with an accuracy of 93.11% on CIFAR-10 and 71.51% on CIFAR-100.

Acknowledgements

This work is part of the research project ("Energy-based probing for Spiking Neural Networks", Contract No. TII/ARRC/2073/2021) in collaboration between Technology Innovation Institute (TII, Abu Dhabi) and Mohamed bin Zayed University of Artificial Intelligence (MBZUAI, Abu Dhabi).

References

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

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

Bu, T., Fang, W., Ding, J., Dai, P., Yu, Z., and Huang, T. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2021.

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

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 113–123, 2019.

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

Deng, S., Li, Y., Zhang, S., and Gu, S. Temporal efficient training of spiking neural network via gradient reweighting. In *International Conference on Learning Representations*, 2021.

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. 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.

Ding, J., Yu, Z., Tian, Y., and Huang, T. 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.

Han, B. and Roy, K. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, pp. 388–404. Springer, 2020.

Han, B., Srinivasan, G., and Roy, K. 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.

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

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

Kim, Y., Li, Y., Park, H., Venkatesha, Y., and Panda, P. Neural architecture search for spiking neural networks. In *Computer Vision – ECCV 2022*, 2022.

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

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

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ANN: towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021.

Liu, Y.-H. and Wang, X.-J. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of computational neuroscience*, 10(1):25–45, 2001.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Skq89Scxx.

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

Neftci, E. O., Mostafa, H., and Zenke, F. 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.

Rathi, N., Srinivasan, G., Panda, P., and Roy, K. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv preprint arXiv:2005.01807*, 2020.

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

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252, 2015.

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12: 331, 2018.

Yang, Y., Zhang, W., and Li, P. Backpropagated neighborhood aggregation for accurate training of spiking neural networks. In *International Conference on Machine Learning*, pp. 11852–11862. PMLR, 2021.

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

Zhang, W. and Li, P. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems*, 33: 12022–12033, 2020.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11062–11070, 2021.

Notations in the Paper

Throughout the paper and this Appendix, we use the following notations in Table S5. 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 $\boldsymbol{\delta} = [\delta]$.

Table S5. 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

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 either 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 total 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. (8) without and 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

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

B.1. Derivation of SlipReLU Activation Function

We start with the SlipReLU activation function in Eq. (8) without shift, then proceed with SlipReLU activation function in Eq. (8) with different shifts.

Derivation of SlipReLU activation function in Eq. (8) without shifts. 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,

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

That is the SlipReLU activation function in Eq. (8),

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

B.2. SlipReLU Activation Function with Different Shift Modes

Derivation of SlipReLU in Eq. (8) with shifts. As mentioned in Sect. 4, the SlipReLU activation function in Eq. (8) 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 shifting in the SlipReLU activation function. The SlipReLU extension with in Eq. (8) can be formulated as follows,

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

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.

1. **Mode 0:** We set $\delta_1 = \delta = 0$, then

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

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

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

B.3. Special Cases of the SlipReLU Activation Function

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 3. *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 4. *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

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

Then

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

□

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 $\mathbf{z}$.

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

Then according to Lemma Theorem 3, 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 (\text{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 (\text{C.5})
\end{aligned}$$

Then

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

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

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

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

Here

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

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

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

Then

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

And the minimum L_2-norm of the first error becomes

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

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

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

From Lemma Theorem 3 and Theorem 1, we have

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

Then

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

This concludes the Theorem 2.

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

□

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; Shift value δ 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 $f_{\text{SNN}}(\mathbf{x};\mathbf{W})$
Dataset D
for $\ell=1$ to $f_{\text{ANN}}.layers$ **do**
  if is ReLU activation **then**
    Replace $\text{ReLU}(\mathbf{x})$ by $\text{SlipReLU}(\mathbf{x};N,\theta^{(\ell)})$
  else if is MaxPooling layer **then**
    Replace MaxPooling layer by AvgPooling layer
  end if
end for
for $e=1$ to epochs **do**
  for length of Dataset D **do**
    Sample minibatch $\{(\mathbf{x}^{(0)},\mathbf{y})\}$ from D
    for $\ell=1$ to $f_{\text{ANN}}.layers$ **do**
      $\mathbf{x}^{(\ell)}=\text{SlipReLU}(\mathbf{W}^{(\ell)}\mathbf{x}^{(\ell-1)};N,\theta^{(\ell)})$
    end for
  end for
  $Loss=CrossEntropy(\mathbf{x}^{(\ell)},\mathbf{y})$
  for $\ell=1$ to $f_{\text{ANN}}.layers$ **do**
    $\mathbf{W}^{(\ell)}\leftarrow\mathbf{W}^{(\ell)}-\epsilon\frac{\partial Loss}{\partial\mathbf{W}^{(\ell)}}$
    $\theta^{(\ell)}\leftarrow\theta^{(\ell)}-\epsilon\frac{\partial Loss}{\partial\theta^{(\ell)}}$
  end for
end for
for $\ell=1$ to $f_{\text{ANN}}.layers$ **do**
  $f_{\text{SNN}}.\mathbf{W}^{(\ell)}\leftarrow f_{\text{ANN}}.\mathbf{W}^{(\ell)}$
  $f_{\text{SNN}}.V_{\text{th}}^{(\ell)}\leftarrow f_{\text{ANN}}.\theta^{(\ell)}$
  $f_{\text{SNN}}.\mathbf{v}^{(\ell)}(0)\leftarrow f_{\text{SNN}}.V_{\text{th}}^{(\ell)}\times\delta$
end for
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. The tailored ANN is also called the source ANN. In the second step, we train the tailored ANN. After training the tailored ANN, we copy all weights from the trained-tailored source ANN to the converted SNN, and set the threshold $V_{\text{th}}^{(\ell)}$ in each layer of the converted SNN equal to the threshold 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 activation in the tailored source ANN, where the optimal shift δ can be any value in the interval $\delta\in[-\frac{1}{2},\frac{1}{2}]$.

Common data normalization and some data pre-processing techniques are used in the experiments. For example, we resize the images in the CIFAR-10/CIFAR-100 datasets into 32×32. Besides, random cropping images, Cutout (DeVries & Taylor, 2017) and AutoAugment (Cubuk et al., 2019) are used for all datasets. The Stochastic Gradient Descent (SGD) optimizer (Bottou, 2012) is used in the experiments with a momentum parameter of 0.9. We use a cosine decay scheduler (Loshchilov & Hutter, 2017) to adjust the learning rate with a weight decay 5×10^{-4} for CIFAR-10/CIFAR-100 datasets. All models are trained for 300 epochs. We set the initial learning rate to $\epsilon=0.1$ for CIFAR-10 and CIFAR-100.

When considering small quasi-latency $N = 1$ and $N = 2$ for CIFAR-10/CIFAR-100, we first try to train the model with learning rate $\epsilon = 0.1$, for models that can not be trained properly with learning rate $\epsilon = 0.1$, we set the initial learning rate to 0.05. We set $\delta_1 = 0, \delta = \frac{1}{2}$ for the SlipReLU activation for all the models and all the datasets. For the ablation study, we train all the networks on CIFAR-10/CIFAR-100 dataset with quasi-latencies $N = 1, 2, 4, 8, 16, 32, 64$ and slopes $c = 0.1, \cdots, 0.9$.

As for the input to the first layer and the output of the last layer of the SNN, we do not employ any spiking mechanism as in Li et al. (2021). We directly encode the static image to temporal dynamic spikes as input to the first layer, which can prevent the undesired information loss introduced by the Poisson encoding. For the last layer output, we only integrate the pre-synaptic input and do not fire any spikes. We use constant input when evaluating the converted SNNs.

E.2. Introduction of Datasets

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.

CIFAR-100: The CIFAR-100 dataset (Krizhevsky & Hinton, 2009) consists of $60,000$ 32×32 color images in 100 classes with $6,000$ images per class. There are $50,000$ samples in the training set and $10,000$ samples in the test set.

ImageNet: In the ImageNet dataset (Russakovsky et al., 2015), the training set is composed of $1,281,167$ images with 1000 object classes, the validation set is composed of $50,000$ images with 1000 object classes, and the testing set is composed of $100,000$ images with 1000 object classes. We use the ImageNet dataset validation set as the test set. The images vary in dimensions and resolution. Many applications resize/crop all the images to 256×256 pixels.

F. Comparison of SlipReLU and SlipReLU-Shift Activation

Here we further conduct ablation studies on SlipReLU and SlipReLU-shift, by comparing the performance of SNNs converted from ANNs with SlipReLU activation and ANN with SlipReLU-shift activation. In Sect. 4, we prove that for arbitrary T and N, the expectation of the conversion error reaches 0 with the SlipReLU-shift activation function when $c = 0$. We also prove that 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 $c(V_{\text{th}}^{(\ell)})^2/(4T)$. To verify these, we set $N = 1, 2, 4, 8, 16, 32$ and train ANNs with SlipReLU activation and SlipReLU-shift activation, respectively.

Fig. S3 shows how the accuracy of converted SNNs changes with respect to the time-step T under different quasi-latency N settings. The accuracy of the converted SNN from ANN with SlipReLU activation (in the first and third columns) first increases or stays flat for time-step $T \leqslant 4$, and then decreases rapidly with the increase of time-steps, because we cannot guarantee that the conversion error is zero when $c \neq 0$. The best performance is still lower than the SlipReLU-shift activation. The non-shifted SlipReLU activation shows no advantage for ultra-low latency inference when $T \leqslant 4$. In contrast, the accuracy of the converted SNN from ANN with SlipReLU-shift activation (in the second and fourth columns) increases with the increase of time-step T. It converges to the same accuracy when the time step is larger than 16. The SlipReLU-shift activation shows advantages for ultra-low latency inference when $T \leqslant 4$.

G. Effect of the Slope c and the Quasi-Latency N

In our SlipReLU method, the slope c balances the weight of the threshold ReLU and the step function, which affects the accuracy of the converted SNN. To analyze the effect of c and better determine the optimal value, we train VGG-16/ResNet-20 networks with quasi-latency $N = 1, 2, 4, 8, 16, 32$, and then converted the trained networks to SNNs. The experimental results on CIFAR-10/100 dataset are shown in Fig. S4, where each of the colored curves shows the effect of the slope c on the SNN accuracy over different time-step/latency T, under different quasi-latency settings. Table S6, Table S7 and Table S8 are the detailed data used to plot the curves.

H. Selecting the Best Hyper-parameters including the Slope c and the Quasi-Latency N

As there are two hyper-parameters in the proposed SlipReLU method, i.e., the slope c and the quasi-latency N, it is better to choose hyper-parameters based on some criterion rather than a rule-of-thumb. Therefore, we use the criterion measure

in Sect. 4.4 to select the best hyper-parameters, and we rewrite it here,

$$\text{Criterion} = \frac{1}{2}\left|Acc_{\text{ReLU}} - Acc_{\text{ANN}}\right| + \frac{1}{2}\left|Acc_{\text{ANN}} - Acc_{\text{SNN}}\right|.$$

Here we take VGG-16 on CIFAR-10 as an example, and the best hyper-parameters are $(N, c) = (2, 0.2)$. Then we convert this best single model to SNN and obtain SNN accuracy under different time-steps T. The detailed results of how to choose the best hyper-parameter over quasi-latency N and slope c with VGG-16 on CIFAR-10 are shown in Table S9 and Table S10.

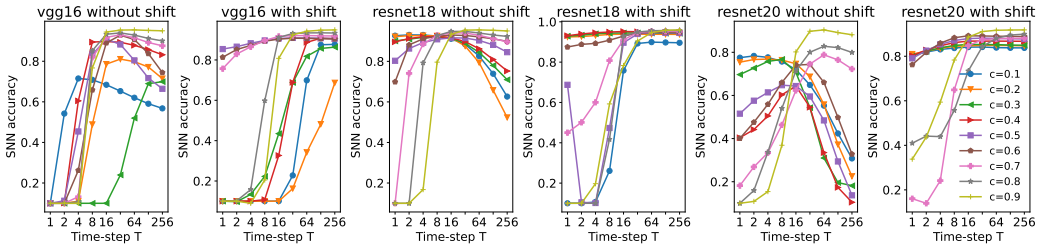
I. Future Study

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).*

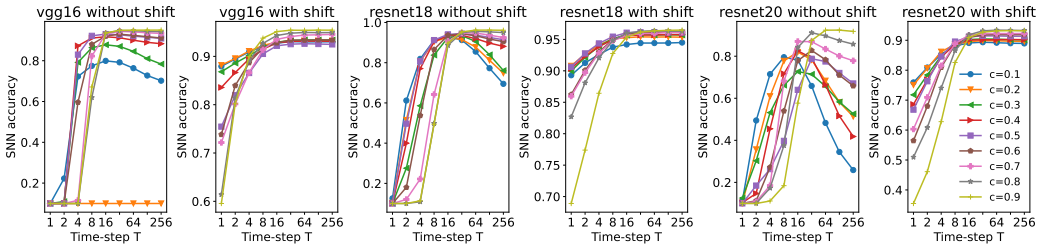
Remark 3. *Usually, implicit variables of an optimization problem are variables which do not need to be optimized but are used to model feasibility conditions (Mehlitz & Benko, 2021), and they are often interpreted as explicit ones (Mehlitz & Benko, 2021), by using the union of image sets associated 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 spike-output nature of SNN implies that higher-latency output depends 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.

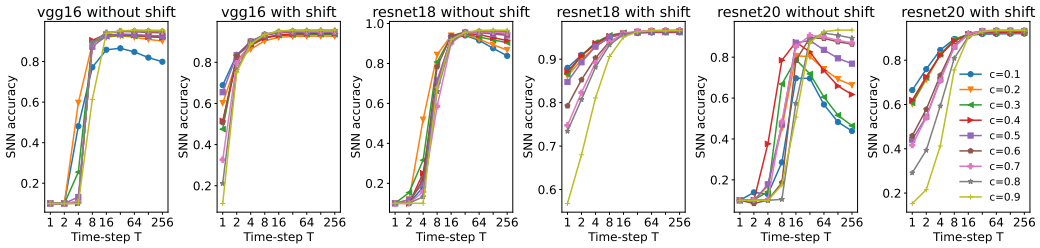
As mentioned in Sect. 6.4, another aspect of the future work is that we consider learning the slope c and the quasi-latency N during ANN training, rather than using them as hyper-tuning parameters, so that the best combination of (N, c) can be found without repeating the training, thus improving the efficiency of the proposed method.



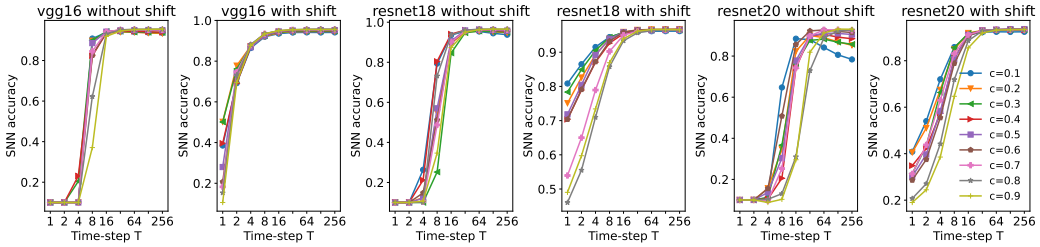
(a) Compare SlipReLU activation with/without shift when the quasi-latency $N = 1$



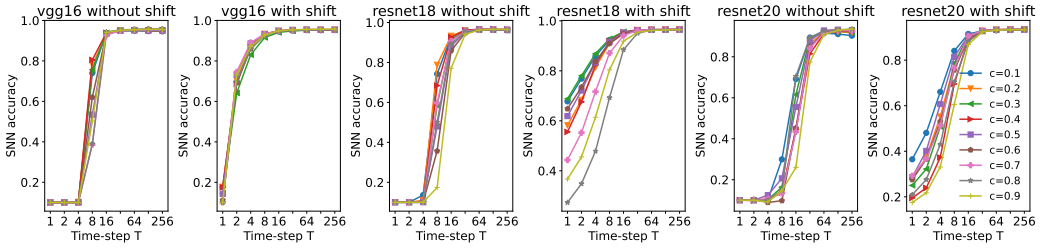
(b) Compare SlipReLU activation with/without shift when the quasi-latency $N = 2$



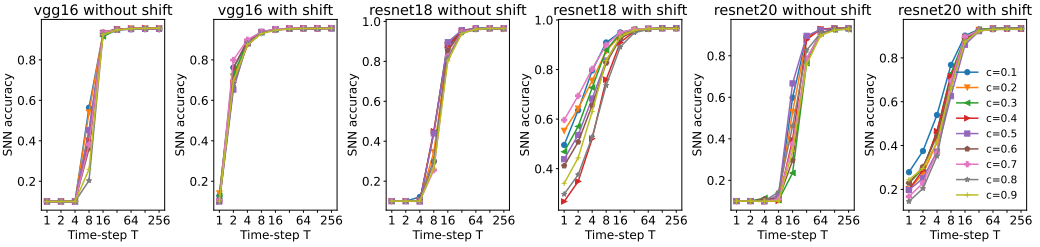
(c) Compare SlipReLU activation with/without shift when the quasi-latency $N = 4$



(d) Compare SlipReLU activation with/without shift when the quasi-latency $N = 8$

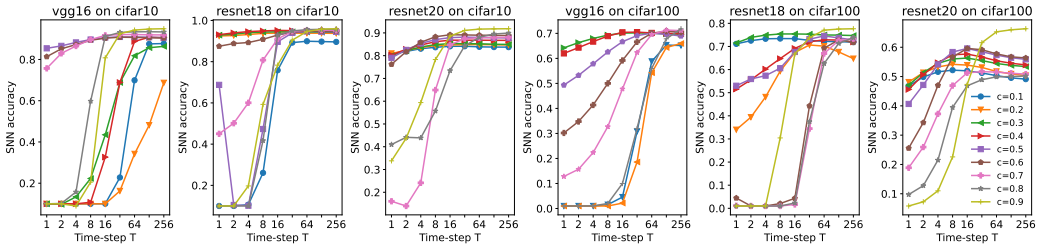


(e) Compare SlipReLU activation with/without shift when the quasi-latency $N = 16$

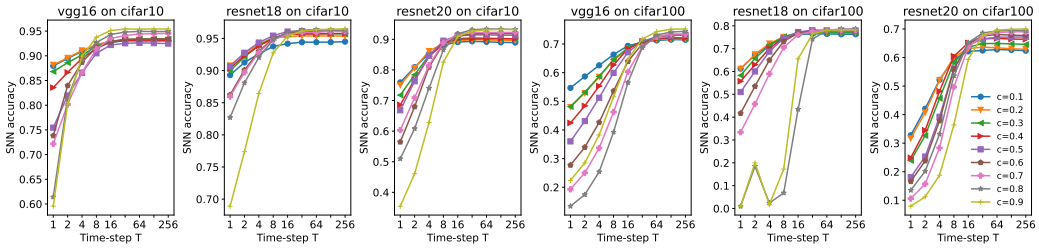


(f) Compare SlipReLU activation with/without shift when the quasi-latency $N = 32$

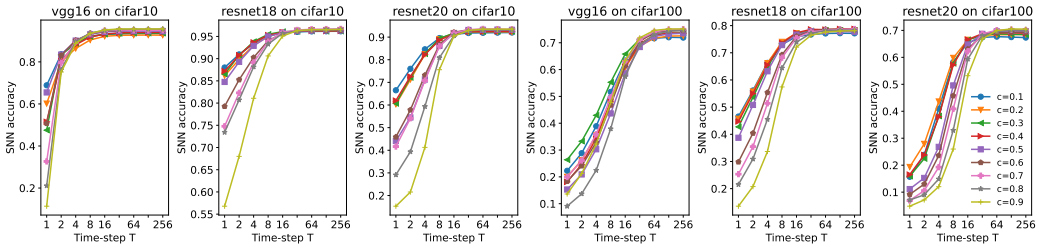
Figure S3. Ablation studies on SlipReLU activation with/without shift under different slopes c with different quasi-latency N.



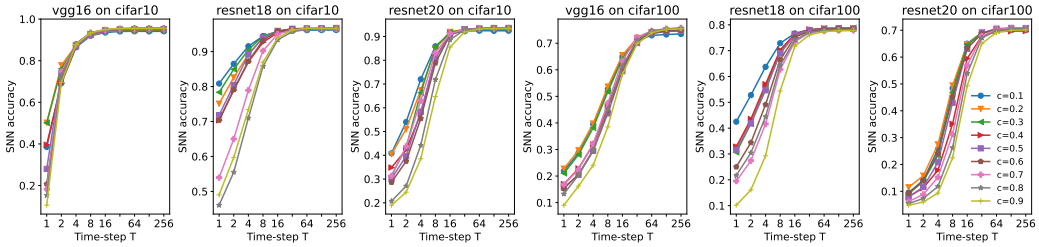
(a) Influence of different slopes with the quasi-latency $N = 1$ on CIFAR-10 and CIFAR-100



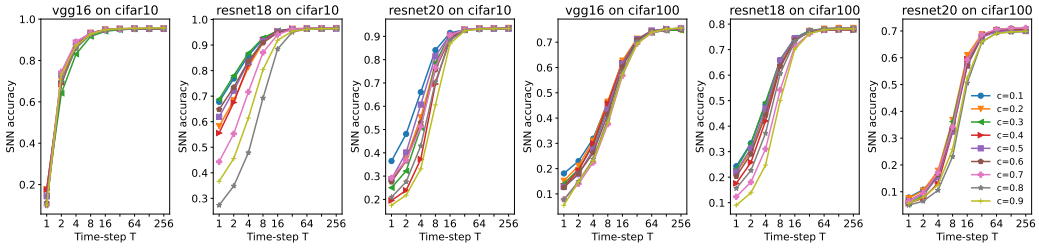
(b) Influence of different slopes with the quasi-latency $N = 2$ on CIFAR-10 and CIFAR-100



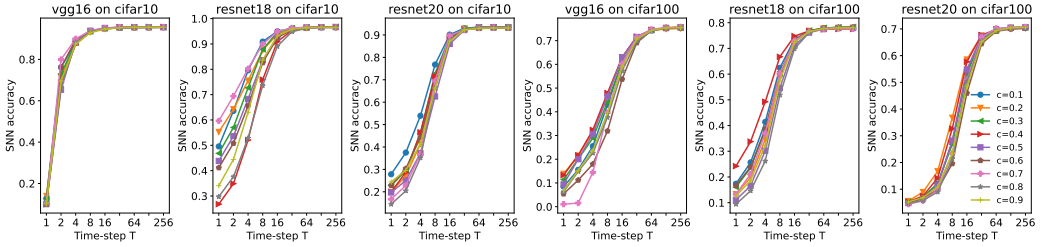
(c) Influence of different slopes with the quasi-latency $N = 4$ on CIFAR-10 and CIFAR-100



(d) Influence of different slopes with the quasi-latency $N = 8$ on CIFAR-10 and CIFAR-100



(e) Influence of different slopes with the quasi-latency $N = 16$ on CIFAR-10 and CIFAR-100



(f) Influence of different slopes with the quasi-latency $N = 32$ on CIFAR-10 and CIFAR-100

Figure S4. Effect of different slopes c with different quasi-latency N on CIFAR-10 and CIFAR-100.

Table S6. Influence of different slope c with the quasi-latency $N = 1$.

Slope c	ANN Acc.	T=1	T=2	T=4	T=8	T=16	T=32	T=64	T=128	T=256
VGG-16 on CIFAR-10										
c=0.1	90.93	10.00	10.00	10.00	10.00	10.05	22.78	69.91	87.55	87.72
c=0.2	88.68	10.00	10.00	10.00	10.00	10.00	16.25	34.22	48.16	68.69
c=0.3	86.73	10.00	9.74	13.37	22.29	43.48	68.81	81.81	85.71	86.35
c=0.4	91.90	10.00	10.00	10.00	10.77	32.66	68.87	88.74	90.77	90.55
c=0.5	90.86	85.40	86.59	88.27	89.67	90.67	90.93	90.91	90.81	90.59
c=0.6	90.97	81.35	85.18	87.21	89.32	90.32	90.79	90.70	90.52	90.42
c=0.7	92.15	75.68	82.96	86.52	89.64	91.68	92.05	92.10	91.94	91.90
c=0.8	93.51	10.00	10.19	15.76	59.78	91.17	93.11	93.40	93.45	93.35
c=0.9	94.93	10.00	10.00	8.95	20.52	80.80	92.47	94.24	94.76	94.90
ResNet-18 on CIFAR-10										
c=0.1	89.27	10.00	10.00	10.66	26.07	75.84	89.25	89.92	89.75	89.60
c=0.2	93.36	92.47	92.68	93.17	93.74	93.86	93.82	93.82	93.81	93.73
c=0.3	94.09	92.86	93.35	94.06	94.37	94.47	94.48	94.42	94.29	94.27
c=0.4	94.61	93.11	93.97	94.59	94.92	95.18	95.07	94.81	94.71	94.67
c=0.5	94.79	68.75	10.48	10.14	47.34	89.64	93.96	94.67	94.55	94.49
c=0.6	94.99	87.49	88.80	89.29	90.87	92.91	94.37	94.83	94.73	94.71
c=0.7	95.39	45.02	50.13	60.06	80.77	91.06	94.72	95.24	95.27	95.17
c=0.8	95.92	10.00	10.00	10.00	41.71	92.91	94.93	95.54	95.70	95.71
c=0.9	96.28	9.99	10.02	19.72	59.28	78.32	90.47	94.63	95.80	95.98
ResNet-20 on CIFAR-10										
c=0.1	81.53	80.65	81.87	83.06	83.68	84.11	84.14	83.88	83.78	83.75
c=0.2	82.07	80.99	82.25	83.52	84.46	84.70	84.85	84.89	84.80	84.69
c=0.3	83.46	80.03	82.17	83.81	84.84	85.32	85.26	85.22	85.01	84.95
c=0.4	84.97	80.30	82.80	84.69	86.12	86.81	86.79	86.79	86.75	86.71
c=0.5	86.49	79.06	82.53	85.36	87.06	87.93	88.13	88.02	87.87	87.77
c=0.6	88.48	76.21	81.74	85.86	88.30	89.19	89.11	88.99	88.93	88.85
c=0.7	89.70	16.04	13.97	24.14	64.91	84.75	87.43	87.86	88.06	88.08
c=0.8	91.07	40.97	44.14	43.90	55.70	73.42	84.29	88.08	89.52	89.93
c=0.9	92.98	33.81	43.71	59.40	78.30	88.60	91.09	91.66	91.78	91.83
VGG-16 on CIFAR-100										
c=0.2	65.04	1.00	1.00	1.01	1.53	4.57	30.82	58.80	65.33	65.13
c=0.3	66.05	1.00	1.00	1.00	1.00	2.15	18.58	54.26	64.34	65.96
c=0.4	68.46	64.21	66.30	67.97	69.31	70.09	70.19	70.05	69.79	69.62
c=0.5	69.30	61.99	64.31	66.71	68.91	70.42	70.50	70.18	70.03	69.85
c=0.6	69.49	49.34	53.22	57.83	62.58	66.67	69.11	70.07	69.63	68.96
c=0.7	70.97	30.19	34.77	41.37	50.01	59.17	66.61	70.07	70.85	70.45
c=0.8	72.13	12.81	15.68	22.37	32.70	47.78	62.35	69.54	71.31	71.11
c=0.9	74.76	1.00	1.00	1.03	2.02	9.97	32.00	55.97	67.82	71.85
ResNet-18 on CIFAR-100										
c=0.1	71.84	71.11	72.51	73.32	73.41	73.38	72.63	72.19	72.06	71.88
c=0.2	72.32	34.00	39.42	48.16	59.34	67.41	70.63	70.14	67.63	64.74
c=0.3	74.01	71.51	73.91	74.89	75.40	75.41	75.30	74.98	74.90	74.71
c=0.4	73.90	51.56	55.56	60.20	64.74	69.16	71.99	72.89	72.76	71.94
c=0.5	74.88	53.01	55.92	57.37	60.59	67.62	73.15	74.53	73.70	72.81
c=0.6	75.93	4.45	1.01	1.01	2.05	4.33	44.28	69.24	72.85	71.71
c=0.7	76.44	1.13	1.00	1.00	1.00	1.66	34.47	66.96	72.57	73.35
c=0.8	78.41	1.00	1.00	1.00	1.00	2.31	37.50	62.62	71.34	73.83
c=0.9	78.18	1.00	1.00	1.04	30.44	66.73	73.81	77.04	77.52	77.66
ResNet-20 on CIFAR-100										
c=0.1	48.62	46.80	49.85	51.61	52.19	51.95	51.23	50.31	49.56	49.12
c=0.2	50.79	48.12	51.35	53.27	54.17	53.91	53.11	51.75	50.89	50.35
c=0.3	52.84	47.08	51.34	54.51	56.00	56.31	55.46	54.46	53.82	53.42
c=0.4	55.18	45.58	50.63	54.72	57.44	57.67	56.69	55.38	54.54	53.97
c=0.5	57.51	40.65	47.14	54.15	58.37	59.59	58.47	57.33	56.41	55.88
c=0.6	59.98	25.56	34.28	47.01	56.64	59.60	59.16	57.74	56.73	56.35
c=0.7	64.71	18.87	25.93	37.26	46.92	51.28	51.68	51.52	51.12	50.84
c=0.8	66.96	9.73	12.76	21.48	39.48	46.84	48.91	49.90	50.01	50.18
c=0.9	69.36	5.82	7.25	11.01	22.58	47.32	61.57	65.26	65.96	66.32

Table S7. Influence of different slope c with the quasi-latency $N = 2$.

Slope c	ANN Acc.	T=1	T=2	T=4	T=8	T=16	T=32	T=64	T=128	T=256
VGG-16 on CIFAR-10										
c=0.1	92.73	87.94	89.45	90.91	92.06	92.72	93.06	93.00	93.01	93.02
c=0.2	93.02	88.17	89.57	91.08	92.26	92.96	93.19	93.25	93.24	93.25
c=0.3	93.11	86.81	88.60	90.19	91.83	92.96	93.25	93.41	93.40	93.40
c=0.4	93.10	83.57	86.65	89.47	91.48	92.81	93.08	93.18	93.18	93.12
c=0.5	92.84	75.46	81.96	86.56	90.51	92.19	92.53	92.62	92.56	92.48
c=0.6	93.44	73.83	83.96	88.69	91.73	92.79	93.39	93.36	93.42	93.41
c=0.7	94.31	72.17	80.16	86.63	91.38	93.52	94.11	94.41	94.47	94.54
c=0.8	94.93	61.41	81.37	89.14	92.70	94.32	94.80	94.95	94.91	94.93
c=0.9	95.47	59.55	80.19	89.92	93.77	95.13	95.42	95.39	95.44	95.42
ResNet-18 on CIFAR-10										
c=0.1	94.20	89.30	91.26	92.61	93.76	94.19	94.42	94.42	94.43	94.50
c=0.2	95.16	90.79	92.68	94.11	95.08	95.28	95.41	95.37	95.39	95.35
c=0.3	95.42	89.97	92.13	93.90	95.14	95.68	95.82	95.70	95.74	95.69
c=0.4	95.56	90.37	92.32	93.85	94.96	95.62	95.68	95.71	95.75	95.78
c=0.5	95.97	90.63	92.77	94.36	95.44	96.07	96.14	96.15	96.11	96.10
c=0.6	95.98	86.23	90.08	93.02	94.96	95.81	96.10	96.14	96.12	96.15
c=0.7	96.06	85.96	89.72	92.81	94.81	95.60	95.93	95.95	96.10	96.14
c=0.8	96.46	82.69	88.09	92.04	94.81	95.99	96.29	96.39	96.31	96.29
c=0.9	96.48	68.90	77.39	86.43	92.76	95.38	96.15	96.36	96.45	96.48
ResNet-20 on CIFAR-10										
c=0.1	87.91	75.92	80.90	85.29	88.14	89.10	89.35	89.19	89.01	88.95
c=0.2	88.66	75.06	80.65	86.17	88.65	89.51	89.90	89.83	89.69	89.61
c=0.3	89.53	71.77	78.42	84.76	88.82	90.24	90.45	90.37	90.20	90.15
c=0.4	89.73	68.57	76.85	84.33	88.71	90.05	90.14	90.20	90.25	90.20
c=0.5	90.72	66.82	76.31	84.58	89.57	91.13	91.46	91.45	91.35	91.32
c=0.6	91.48	56.46	67.99	80.94	88.52	90.99	91.77	91.89	91.84	91.88
c=0.7	92.17	60.32	70.93	81.52	88.88	91.72	92.26	92.26	92.25	92.27
c=0.8	92.91	50.95	60.84	74.04	86.55	91.83	93.14	93.40	93.35	93.26
c=0.9	93.11	35.47	46.10	62.81	82.57	90.93	92.71	93.14	93.21	93.18
VGG-16 on CIFAR-100										
c=0.1	70.03	54.68	58.66	62.56	66.31	69.35	70.65	71.23	71.52	71.47
c=0.2	70.73	48.02	52.87	58.53	64.34	68.35	70.66	71.52	71.79	71.76
c=0.3	71.16	48.14	53.15	58.71	64.57	68.66	70.93	71.83	72.00	71.98
c=0.4	71.43	42.45	48.41	55.32	62.68	68.34	70.84	71.88	72.17	72.07
c=0.5	72.66	36.01	43.11	51.25	59.92	67.10	70.95	72.48	72.91	73.15
c=0.6	72.73	27.72	33.97	42.64	53.60	63.91	70.07	72.61	73.26	73.35
c=0.7	73.47	19.30	25.04	33.69	46.26	60.29	69.07	72.51	73.46	73.46
c=0.8	74.12	13.40	17.44	25.41	39.23	56.54	68.80	73.11	74.18	74.41
c=0.9	75.18	22.41	28.52	38.27	51.58	64.70	71.73	74.32	75.14	75.25
ResNet-18 on CIFAR-100										
c=0.1	75.38	61.23	67.17	71.52	74.64	76.20	76.50	76.46	76.29	76.30
c=0.2	76.15	61.14	67.49	72.24	75.16	76.66	77.04	76.96	76.95	76.94
c=0.3	76.60	58.29	65.68	71.51	75.28	76.96	77.07	76.99	77.00	76.99
c=0.4	77.32	55.72	62.98	70.09	74.72	77.02	77.99	77.98	77.82	77.79
c=0.5	77.08	51.01	60.03	68.72	74.59	77.29	78.04	77.97	77.99	77.91
c=0.6	77.42	41.69	53.51	64.96	73.17	76.90	77.57	77.68	77.85	77.80
c=0.7	77.93	33.40	45.77	58.96	70.54	76.15	77.53	78.02	78.02	78.08
c=0.8	78.22	1.00	18.51	2.38	6.82	43.49	74.16	78.05	78.59	78.64
c=0.9	78.22	1.00	19.97	1.71	17.25	65.52	76.47	78.10	78.26	78.23
ResNet-20 on CIFAR-100										
c=0.1	59.83	32.76	42.03	52.20	59.45	62.15	62.47	62.81	62.65	62.40
c=0.2	61.36	31.66	40.76	52.05	60.08	63.17	63.63	63.42	63.07	62.90
c=0.3	62.96	23.91	32.65	45.70	58.25	63.62	64.90	64.92	64.80	64.59
c=0.4	64.32	24.88	34.52	48.19	60.47	65.14	66.55	66.72	66.46	66.41
c=0.5	65.85	18.07	25.30	39.24	56.10	64.30	66.50	67.12	67.15	67.21
c=0.6	66.75	16.58	23.81	37.84	56.00	64.97	67.53	68.09	67.85	67.71
c=0.7	68.49	10.58	15.74	28.33	49.67	63.67	67.64	68.77	68.98	69.03
c=0.8	69.03	13.51	20.17	33.17	53.63	65.21	68.59	69.32	69.51	69.45
c=0.9	69.70	7.92	11.19	18.76	36.40	59.36	67.75	69.62	69.89	70.02

Table S8. Influence of different slope c with the quasi-latency $N = 4$.

Slope c	ANN Acc.	T=1	T=2	T=4	T=8	T=16	T=32	T=64	T=128	T=256
VGG-16 on CIFAR-10										
c=0.1	93.24	68.87	83.64	89.31	92.14	93.03	93.32	93.41	93.47	93.47
c=0.2	92.68	60.15	80.03	86.37	90.43	92.17	92.62	92.71	92.75	92.71
c=0.3	93.55	47.52	83.30	88.85	92.07	93.18	93.54	93.65	93.68	93.67
c=0.4	93.94	51.52	80.67	88.13	91.89	93.20	93.81	93.98	94.03	94.01
c=0.5	94.54	65.47	83.46	90.13	93.09	94.25	94.61	94.62	94.63	94.58
c=0.6	94.95	50.73	83.07	90.25	93.37	94.61	94.91	95.06	95.05	95.02
c=0.7	95.02	32.67	79.63	89.69	93.55	94.84	95.14	95.04	95.02	95.05
c=0.8	95.52	21.08	76.27	89.69	93.73	94.98	95.47	95.53	95.61	95.60
c=0.9	95.60	11.37	75.18	88.80	93.54	95.20	95.66	95.65	95.66	95.67
ResNet-18 on CIFAR-10										
c=0.1	96.01	88.01	90.96	93.34	95.12	95.86	96.02	96.13	96.16	96.16
c=0.2	96.31	86.16	89.82	93.00	95.02	95.90	96.27	96.43	96.44	96.45
c=0.3	96.15	86.52	90.78	93.84	95.48	96.10	96.12	96.22	96.15	96.19
c=0.4	96.27	87.18	90.76	93.66	95.29	95.90	96.13	96.25	96.21	96.25
c=0.5	96.38	84.76	89.29	92.89	94.99	95.82	96.27	96.33	96.36	96.36
c=0.6	96.29	79.25	85.25	90.26	94.05	95.68	96.30	96.39	96.42	96.41
c=0.7	96.68	74.78	82.30	89.16	93.86	95.89	96.46	96.60	96.66	96.69
c=0.8	96.53	73.43	80.72	88.15	93.28	95.70	96.23	96.45	96.55	96.58
c=0.9	96.67	56.79	68.00	81.08	90.61	95.08	96.31	96.53	96.52	96.59
ResNet-20 on CIFAR-10										
c=0.1	91.42	66.51	75.99	84.62	89.58	91.24	91.80	91.89	91.97	92.01
c=0.2	91.82	60.30	71.25	82.44	89.05	91.79	92.27	92.36	92.35	92.28
c=0.3	91.81	60.66	72.13	82.62	89.40	91.74	92.36	92.46	92.53	92.51
c=0.4	92.07	61.96	72.57	82.27	88.68	91.45	92.38	92.46	92.55	92.55
c=0.5	92.91	44.08	54.50	71.27	86.16	91.66	93.14	93.24	93.32	93.21
c=0.6	92.96	45.87	57.82	73.17	86.66	92.13	93.23	93.36	93.29	93.19
c=0.7	93.30	41.65	54.13	70.79	85.97	91.79	93.28	93.61	93.55	93.46
c=0.8	93.33	29.14	39.35	59.35	80.90	90.65	92.76	93.14	93.26	93.24
c=0.9	93.37	15.29	21.55	41.27	75.60	90.45	92.95	93.49	93.52	93.52
VGG-16 on CIFAR-100										
c=0.1	71.78	22.27	28.83	38.88	51.68	63.38	69.68	71.64	72.04	71.92
c=0.2	72.16	20.01	26.24	35.41	47.79	60.82	68.78	71.67	72.47	72.59
c=0.3	73.40	26.37	33.29	42.97	55.27	65.80	71.30	73.26	73.55	73.69
c=0.4	73.18	18.13	24.70	34.32	47.98	61.11	69.59	72.83	73.65	73.51
c=0.5	73.25	15.29	20.91	30.29	43.65	58.57	68.41	72.33	73.28	73.53
c=0.6	74.26	18.37	24.09	32.89	46.76	61.41	70.50	73.83	74.42	74.50
c=0.7	74.94	19.95	26.09	35.86	49.68	63.12	71.18	74.05	74.91	75.03
c=0.8	74.50	9.07	13.70	22.44	37.92	57.41	69.20	73.02	74.33	74.69
c=0.9	75.25	13.62	21.23	32.24	47.76	63.32	71.84	74.61	75.13	75.15
ResNet-18 on CIFAR-100										
c=0.1	76.71	46.54	56.14	66.28	73.07	75.93	76.72	77.11	77.20	77.15
c=0.2	77.82	45.71	55.72	66.18	74.02	77.19	77.96	78.15	78.21	78.25
c=0.3	77.85	42.74	53.62	64.77	73.37	76.95	78.06	78.26	78.26	78.26
c=0.4	78.28	44.72	55.01	65.50	73.58	77.36	78.61	78.54	78.76	78.78
c=0.5	77.69	38.73	50.81	63.20	72.84	76.69	77.95	77.94	77.77	77.74
c=0.6	78.30	29.83	40.41	55.37	69.27	76.07	77.94	78.66	78.61	78.59
c=0.7	78.56	25.33	35.45	51.41	68.22	75.46	77.79	78.24	78.55	78.75
c=0.8	77.96	21.54	30.90	45.51	64.42	73.94	77.14	78.16	78.34	78.33
c=0.9	78.00	13.54	20.76	33.67	57.43	72.09	76.43	77.60	77.98	78.14
ResNet-20 on CIFAR-100										
c=0.1	66.37	15.69	23.85	40.99	58.23	65.42	67.39	67.68	67.51	67.33
c=0.2	66.91	19.33	27.89	43.62	59.79	66.51	68.16	68.40	68.42	68.45
c=0.3	67.39	15.92	22.44	38.25	57.74	66.47	68.29	68.70	68.59	68.45
c=0.4	68.40	16.52	23.79	37.94	57.20	66.61	68.76	69.04	69.09	68.96
c=0.5	68.86	11.14	15.27	26.79	49.58	64.77	68.70	69.63	69.75	69.69
c=0.6	68.83	9.15	12.99	23.57	45.74	63.08	68.25	69.08	69.24	69.32
c=0.7	69.45	6.90	10.56	19.27	40.90	62.26	68.71	70.06	70.29	70.13
c=0.8	69.59	7.09	9.00	14.88	32.93	59.35	68.05	69.61	70.08	69.94
c=0.9	70.18	4.79	7.11	12.04	25.98	53.20	66.77	69.95	70.54	70.46

Table S9. Choosing the best hyper-parameter over quasi-latency N and slope c with VGG-16 on CIFAR-10.

Quasi-latency N	Slope c	Acc_{ReLU}	Acc_{ANN}	Acc_{SNN}	$\lvert Acc_{\mathrm{ReLU}} - Acc_{\mathrm{ANN}} \rvert$	$\lvert Acc_{\mathrm{ANN}} - Acc_{\mathrm{SNN}} \rvert$	Criterion
	0.10	95.92	90.93	10.00	4.99	80.93	85.92
	0.20	95.92	88.68	10.00	7.24	78.68	85.92
	0.30	95.92	86.73	10.00	9.19	76.73	85.92
	0.40	95.92	91.90	10.00	4.02	81.90	85.92
$N = 1$	0.50	95.92	90.86	85.40	5.06	5.46	10.52
	0.60	95.92	90.97	81.35	4.95	9.62	14.57
	0.70	95.92	92.15	75.68	3.77	16.47	20.24
	0.80	95.92	93.51	10.00	2.41	83.51	85.92
	0.90	95.92	94.93	10.00	0.99	84.93	85.92
	0.10	95.92	92.73	87.94	3.19	4.79	7.98
	0.20	95.92	93.02	88.17	2.90	4.85	**7.75**
	0.30	95.92	93.11	86.81	2.81	6.30	9.11
	0.40	95.92	93.10	83.57	2.82	9.53	12.35
$\boldsymbol{N = 2}$	0.50	95.92	92.84	75.46	3.08	17.38	20.46
	0.60	95.92	93.44	73.83	2.48	19.61	22.09
	0.70	95.92	94.31	72.17	1.61	22.14	23.75
	0.80	95.92	94.93	61.41	0.99	33.52	34.51
	0.90	95.92	95.47	59.55	0.45	35.92	36.37
	0.10	95.92	93.24	68.87	2.68	24.37	27.05
	0.20	95.92	92.68	60.15	3.24	32.53	35.77
	0.30	95.92	93.55	47.52	2.37	46.03	48.40
	0.40	95.92	93.94	51.52	1.98	42.42	44.40
$N = 4$	0.50	95.92	94.54	65.47	1.38	29.07	30.45
	0.60	95.92	94.95	50.73	0.97	44.22	45.19
	0.70	95.92	95.02	32.67	0.90	62.35	63.25
	0.80	95.92	95.52	21.08	0.40	74.44	74.84
	0.90	95.92	95.60	11.37	0.32	84.23	84.55
	0.10	95.92	94.21	38.54	1.71	55.67	57.38
	0.20	95.92	94.65	50.20	1.27	44.45	45.72
	0.30	95.92	94.84	50.11	1.08	44.73	45.81
	0.40	95.92	95.01	39.63	0.91	55.38	56.29
$N = 8$	0.50	95.92	95.25	27.98	0.67	67.27	67.94
	0.60	95.92	95.41	20.70	0.51	74.71	75.22
	0.70	95.92	95.58	18.26	0.34	77.32	77.66
	0.80	95.92	95.64	15.30	0.28	80.34	80.62
	0.90	95.92	95.60	10.56	0.32	85.04	85.36

Table S10. Choosing the best hyper-parameter over quasi-latency N and slope c with VGG-16 on CIFAR-10.

Quasi-latency N	Slope c	Acc_{ReLU}	Acc_{ANN}	Acc_{SNN}	$\lvert Acc_{\mathrm{ReLU}} - Acc_{\mathrm{ANN}} \rvert$	$\lvert Acc_{\mathrm{ANN}} - Acc_{\mathrm{SNN}} \rvert$	Criterion
$N = 16$	0.10	95.92	95.28	17.71	0.64	77.57	78.21
	0.20	95.92	95.44	17.80	0.48	77.64	78.12
	0.30	95.92	95.33	14.61	0.59	80.72	81.31
	0.40	95.92	95.56	17.49	0.36	78.07	78.43
	0.50	95.92	95.47	14.33	0.45	81.14	81.59
	0.60	95.92	95.37	11.17	0.55	84.20	84.75
	0.70	95.92	95.60	10.08	0.32	85.52	85.84
	0.80	95.92	95.44	10.03	0.48	85.41	85.89
	0.90	95.92	95.77	10.03	0.15	85.74	85.89
$N = 32$	0.10	95.92	95.49	12.78	0.43	82.71	83.14
	0.20	95.92	95.53	13.98	0.39	81.55	81.94
	0.30	95.92	95.54	12.72	0.38	82.82	83.20
	0.40	95.92	95.69	10.01	0.23	85.68	85.91
	0.50	95.92	95.50	10.00	0.42	85.50	85.92
	0.60	95.92	95.49	10.03	0.43	85.46	85.89
	0.70	95.92	95.64	10.51	0.28	85.13	85.41
	0.80	95.92	95.55	10.07	0.37	85.48	85.85
	0.90	95.92	95.60	10.18	0.32	85.42	85.74
$N = 64$	0.10	95.92	95.39	10.00	0.53	85.39	85.92
	0.20	95.92	95.55	10.05	0.37	85.50	85.87
	0.30	95.92	95.77	10.02	0.15	85.75	85.90
	0.40	95.92	95.71	10.07	0.21	85.64	85.85
	0.50	95.92	95.55	10.01	0.37	85.54	85.91
	0.60	95.92	95.53	10.08	0.39	85.45	85.84
	0.70	95.92	95.66	10.03	0.26	85.63	85.89
	0.80	95.92	95.64	10.10	0.28	85.54	85.82
	0.90	95.92	95.61	10.06	0.31	85.55	85.86
$N = 128$	0.10	95.92	95.58	10.04	0.34	85.54	85.88
	0.20	95.92	95.65	10.03	0.27	85.62	85.89
	0.30	95.92	95.51	10.09	0.41	85.42	85.83
	0.40	95.92	95.71	10.16	0.21	85.55	85.76
	0.50	95.92	95.53	10.04	0.39	85.49	85.88
	0.60	95.92	95.36	10.16	0.56	85.20	85.76
	0.70	95.92	95.76	10.05	0.16	85.71	85.87
	0.90	95.92	95.70	10.08	0.22	85.62	85.84