Neural Networks 184 (2025) 107076

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

CS-QCFS: Bridging the performance gap in ultra-low latency spiking neural networks

Hongchao Yang [a,b], Suorong Yang [a,b], Lingming Zhang [a,b], Hui Dou [a,b], Furao Shen [a,c,*], Jian Zhao [d]

[a] National Key Laboratory for Novel Software Technology, Nanjing University, China
[b] School of Computer Science, Nanjing University, Nanjing 210023, China
[c] School of Artificial Intelligence, Nanjing University, Nanjing 210023, China
[d] School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China

ARTICLE INFO

Keywords:
Spiking neural networks
ANN-SNN conversion
Ultra-low latency

ABSTRACT

Spiking Neural Networks (SNNs) are at the forefront of computational neuroscience, emulating the nuanced dynamics of biological systems. In the realm of SNN training methods, the conversion from ANNs to SNNs has generated significant interest due to its potential for creating energy-efficient and biologically plausible models. However, existing conversion methods often require long time-steps to ensure that the converted SNNs achieve performance comparable to the original ANNs. In this paper, we thoroughly investigate the process of ANN-SNN conversion and identify two critical issues: the frequently overlooked heterogeneity across channels and the emergence of negative thresholds, both of which lead to the problem of long time-steps. To address these issues, we introduce an innovative activation function called Channel-wise Softplus Quantization Clip-Floor-Shift (CS-QCFS) activation function. This function effectively handles the disparities between channels and maintain positive thresholds. This innovation enables us to achieve high-performance SNNs, particularly in ultra-low time-steps. Our experimental results demonstrate that the proposed method achieves state-of-the-art performance on CIFAR datasets. For instance, we achieve a top-1 accuracy of 95.86% on CIFAR-10 and 74.83% on CIFAR-100 with only 1 time-step.

1. Introduction

In recent years, the realm of neural networks has experienced extraordinary evolution, extending its scope beyond traditional Artificial Neural Networks (ANNs). This evolution is marked by the emergence of novel architectures that draw inspiration from the biological workings of the human brain. One such promising architecture is the Spiking Neural Network (SNN) (Maass, 1997), which aims to capture the dynamism and efficiency of biological neurons by mimicking the spike-based communication found in the human nervous system. The spike-based communication, where each neuron generates spikes only when its membrane potential exceeds the firing threshold, enables SNNs to exhibit distinctive attributes such as high sparsity, multiplication-free, and biological plausibility (Hao, Bu, Ding, Huang, & Yu, 2023; Hao, Ding, Bu, Huang, & Yu, 2023). Consequently, SNNs demonstrate reduced power consumption and enhanced computational efficiency, especially when deployed on neuromorphic chips (Davies et al., 2018; Merolla et al., 2014), in comparison to ANNs.

Nonetheless, the development of SNNs faces various challenges. The most prominent challenge is the scarcity of universally efficient learning algorithms. This stems from the fact that traditional backpropagation methods used in ANNs are not directly applicable to SNNs due to their inherently non-differentiable nature. Consequently, researchers often resort to two prevalent strategies: training SNNs using specialized algorithms or converting pre-trained ANNs to SNNs. The former involves direct training of SNNs using specialized algorithms such as Spike-Timing Dependent Plasticity (STDP) (Caporale & Dan, 2008) and surrogate gradient methods (Neftci, Mostafa, & Zenke, 2019). STDP utilizes precise spike timing to adjust synaptic strength, which can effectively capture the temporal patterns in data. Surrogate gradient methods approximate gradients for non-differentiable spiking events, enabling gradient-based optimization in SNNs. The latter strategy entails converting pre-trained ANNs to SNNs (Diehl et al., 2015). The conversion approach involves training an ANN using conventional techniques, and then translating its architecture and learned weights into

* Corresponding author at: School of Artificial Intelligence, Nanjing University, Nanjing 210023, China.
E-mail addresses: yanghc@smail.nju.edu.cn (H. Yang), sryang@smail.nju.edu.cn (S. Yang), frshen@nju.edu.cn (F. Shen).

https://doi.org/10.1016/j.neunet.2024.107076
Received 22 December 2023; Received in revised form 3 August 2024; Accepted 18 December 2024
Available online 1 January 2025

an equivalent SNN. The ANN-SNN conversion presents notable advantages. It capitalizes on the extensive knowledge and optimized architectures from the ANN domain, circumventing the challenges of training SNNs directly. Furthermore, it resolves the non-differentiability issue associated with SNNs, providing a straightforward and efficient path to obtain high performance SNNs.

Following the noted advantages of the ANN-SNN conversion, numerous methods have been proposed to reduce the error in the conversion process, including weight normalization (Diehl et al., 2015; Kim, Park, Na, & Yoon, 2020), threshold rescaling (Sengupta, Ye, Wang, Liu, & Roy, 2019), soft-reset (Han, Srinivasan, & Roy, 2020; Rueckauer, Lungu, Hu, Pfeiffer, & Liu, 2017), threshold shift (Deng & Gu, 2021), as well as trainable threshold (Bu et al., 2023; Hao, Bu, et al., 2023; Hao, Ding, et al., 2023). However, previous research on ANN-SNN conversion often overlooks two critical aspects. Firstly, substantial variability exists between different channels within the same layer. Ignoring this variability can result in under-utilization of the representational power of some channels and over-penalization of others, potentially restricting the performance of SNNs. Secondly, maintaining positive thresholds in SNN neurons is indispensable. The threshold of an SNN neuron plays a pivotal role in dictating its spiking behavior. Negative thresholds can introduce behavioral inconsistency between SNN neurons and their corresponding ANN counterparts. Maintaining thresholds within a positive range is crucial for the network's stability and performance, as it fosters consistent neuronal behavior. Addressing these nuances is essential for a comprehensive understanding and optimization of SNN architectures, ultimately improving their performance in complex tasks.

To achieve higher-performance SNNs with ultra-low latency, which means achieving short simulation time-steps (e.g., 1 time-step), we explicitly address these two crucial issues. Firstly, we recognize the significant channel-wise variances within layers, training individual thresholds for each channel to fully utilize their representational power. Secondly, we incorporate the inherent requirement of positive firing thresholds in the ANN-SNN conversion into our model design. We accomplish this by introducing the Softplus function, which ensures that the thresholds consistently remain within a positive range, enhancing stability and performance. Our contributions can be summarized as follows:

- We delve into the process of ANN-SNN conversion, uncovering significant issues related to channel variability within layers and the necessity of maintaining positive thresholds in SNN neurons. Neglecting these issues can lead to imbalanced channel performance and behavioral inconsistencies between SNNs and ANNs, ultimately affecting overall network stability and performance.
- We propose the channel-wise Softplus quantization clip-floor-shift activation function, which not only addresses the potential differences between channels within a layer but also ensures positive thresholds. This approach aims to optimize channel performance while aligning the behavior of SNN neurons with their ANN counterparts, thereby significantly enhancing the performance of SNNs.
- We demonstrate the superior performance of our method on various deep network architectures on CIFAR-10 and CIFAR-100 datasets. Our method outperforms the existing state-of-the-art ANN-SNN conversion methods in the same time-steps. For instance, we achieve a top-1 accuracy of 95.86% on CIFAR-10 and 74.83% on CIFAR-100 with only 1 time-step.

The rest of the paper is organized as follows. Section 2 offers a brief review of the related work in the field of ANN-SNN conversion. In Section 3, we provide a detailed introduction to SNNs. Section 4 details the method we propose. In Section 5, we present our experimental results, demonstrating the effectiveness of our proposed method. The final chapter concludes the paper, summarizing our main findings and contributions.

2. Related work

Due to the intricate temporal dynamics and spike discontinuity, training SNNs directly often presents substantial challenges. This includes the necessity for specialized learning algorithms that can effectively handle the non-differentiable characteristics inherent in spike generation. As an alternative, ANN-SNN conversion has been recently proposed to convert a pre-trained ANN to SNN. This approach offers advantages due to the well-established and efficient training methodologies of ANNs (Wang, Zhang, Chen, & Qu, 2022). Cao, Chen, and Khosla (2015) pioneered a methodological advancement by initially training ANNs with ReLU activations and subsequently replacing these activation layers with spiking neurons. To mitigate the information loss during the conversion from ANNs to SNNs, Rueckauer et al. (2017) proposed a soft-reset mechanism that resets the membrane potential by subtraction. In an effort to bridge the gap between spiking fire rates and ReLU activation values, a family of parameter normalization methods was introduced. Diehl et al. (2015) proposed data-based normalization to improve the performance in deep SNNs. The initial normalization scales are derived from the maximum activation values of each layer in the ANNs. Rueckauer et al. (2017) employed the pth percentile of the total activity distribution as the normalization scale to discard extreme outliers. Typically, p is set to 99 or 99.9. Kim et al. (2020) proposed Spiking-YOLO with channel-wise data-based normalization.

Similar to the concept behind parameter normalization, many works focused on dynamically adjusting the firing threshold. Han et al. (2020) proposed a threshold balancing technique by scaling the SNN thresholds to improve the inference latency. Ding, Yu, Tian, and Huang (2021) proposed the Rate Norm Layer (RNL) to replace the ReLU activation in ANN, where RNL determines the threshold during training. Ho and Chang (2021) presented a trainable clipping layer technique that finds the optimal data-normalization factor in the training process. Deng and Gu (2021) conducted a theoretical analysis of the conversion error and effectively minimized the layer-wise error. This was achieved by employing the threshold ReLU in place of the conventional ReLU, along with the introduction of an additional bias. Li, Deng, Dong, and Gu (2022) further used the Minimization of Mean Squared Error (MMSE) method to obtain the adaptive threshold and calibrate the SNN parameters by quantized fine-tuning. Similar to Deng and Gu (2021) and Li et al. (2022), Bu et al. (2023) delved into the conversion error and introduced the quantization clip-floor-shift activation function for ANNs, which better fit the finiteness and discreteness of the spike firing rate. They also demonstrated that setting the initial membrane potential to half the threshold is optimal, as it results in the expectation of conversion error reaching zero. Due to the presence of the unevenness error described by Bu et al. (2023), there still exists a gap between ANN and SNN. Hao, Bu, et al. (2023) categorized the unevenness error into four cases based on the outputs of ANN and SNN, and proposed an optimization strategy focused on residual membrane potential to alleviate this error. In the subsequent work (Hao, Ding, et al., 2023), they introduced the offset spike concept to gauge the variance between the actual and desired SNN firing rates, proposing an optimization approach that adjusts the initial membrane potential to correct conversion errors.

Although these studies have demonstrated high performance with low latency, they do not adequately address the significant differences that exist between channels within the same layer. Additionally, they overlook the basic requirement in the ANN-to-SNN conversion, which is to maintain a positive firing threshold. In this paper, we give special consideration to these issues and incorporate insights from prior methods to attain higher-accuracy and ultra-low latency SNNs.

3. Preliminaries

3.1. Spiking neural networks

SNNs, characterized by their event-driven nature and low power consumption, represent the third generation of neural networks (Maass, 1997). Contrary to ANNs, SNNs consist of spiking neurons in which information is conveyed through spikes. The SNN neuron features a membrane potential that accumulates the input over time. Once this potential exceeds a predetermined threshold, generally a positive value, the neuron emits a spike and then resets its potential. Integrate-and-Fire (IF) neurons (Burkitt, 2006) are widely-used spiking neurons that accumulate the input z into a membrane potential V_{mem} as:

$$V^l_{mem}(t) = V^l_{mem}(t-1) + z^l(t) - V^l_{th}S^l(t), \tag{1}$$

$$z^l(t) = W^l S^{l-1}(t)V^{l-1}_{th}, \tag{2}$$

$$S^l(t) = \Theta(V^l_{mem}(t) - V^l_{th}). \tag{3}$$

Here, $V^l_{mem}(t)$, $z^l(t)$ and $S^l(t)$, respectively denote the membrane potential, the input, and the spike at time t of the lth layer. Each layer l is characterized by a specific threshold voltage V^l_{th}, which influences the spiking behavior of the neurons. The input $z^l(t)$ is derived from the spike $S^{l-1}(t)$ of the previous layer, as described in Eq. (2), where W^l represents the weight in the lth layer. Additionally, a spike $S^l(t)$ is generated when the integrated membrane potential V_{mem} exceeds the threshold voltage V_{th}, as described in Eq. (3), with $\Theta(\cdot)$ being the Heaviside step function (Weisstein, 2002). Since a neuron generates a spike solely when its membrane potential exceeds the threshold, the resultant spike pattern is inherently sparse. This sparsity implies that only a subset of neurons, specifically those that receive an input spike, are actively involved in processing and integrating this input, as demonstrated in Eq. (2). Compared to ANNs, which typically process in a dense, continuous manner, the event-based computation of SNNs significantly reduces the computational energy consumption.

3.2. ANN-SNN conversion

The fundamental concept of the ANN-SNN conversion is transferring the parameters from pre-trained ANN models to their SNN counterparts (Hao, Bu, et al., 2023). This process aims to approximate the continuous activations in the ANN with the ReLU function by the average spike fire rates (or average postsynaptic potential) observed in the SNN with IF function (Liu, Zhao, Chen, Wang, & Jiang, 2022).

By summing Eq. (1) from $t = 1$ to $t = T$ and substituting variable $z^l(t)$ from Eq. (2) into Eq. (1), we have:

$$V^l_{mem}(T) - V^l_{mem}(0) = W^l V^{l-1}_{th}\sum_{i=1}^{T} S^{l-1}(i) - V^l_{th}\sum_{i=1}^{T} S^l(i). \tag{4}$$

Then dividing T on both sides and denoting the average spike fire rate by $r^l(T) = \sum_{i=1}^{T} S^l(i)/T$, we get:

$$V^l_{th}r^l(T) = W^l V^{l-1}_{th} r^{l-1}(T) - \frac{V^l_{mem}(T) - V^l_{mem}(0)}{T}. \tag{5}$$

The relationship between the average spike fire rates and the average postsynaptic potential is $\phi^l(T) = r^l(T)V^l_{th}$, then we obtain:

$$\phi^l(T) = W^l\phi^{l-1}(T) - \frac{V^l_{mem}(T) - V^l_{mem}(0)}{T}, \tag{6}$$

where $\phi^l(T)$ represents the average postsynaptic potential, which is always non-negative in the context of our derivation. Notably, in the scenario where T approaches infinity, Eq. (6) shows a striking resemblance to the forward propagation in ANNs. Specifically, in ANNs, the forward propagation can be described as:

$$a^l = \mathrm{ReLU}(W^l a^{l-1}), \tag{7}$$

where a^l represents the activation of the layer l.

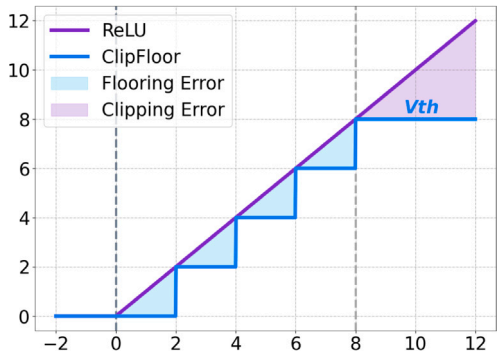


Fig. 1. Errors in ANN-SNN Conversion.

3.3. Conversion error analysis

To understand the differences between IF neurons and ReLU-based neurons, we theoretically analyze the conversion error between the outputs of ReLU-based neurons and IF neurons. In the IF neurons, the average postsynaptic potential is:

$$\phi^l(T) = r^l(T)V^l_{th} = V^l_{th}(\sum_{i=1}^{T} S^l(i))/T. \tag{8}$$

Since the average spike fire rate of IF neurons is in the range of $[0, 1]$, the corresponding average postsynaptic potential is also between 0 and V_{th}. At each time-step, the spike $S(i)$ can be either 0 or 1, thus the average postsynaptic potential $\phi(T)$ has a quantization resolution of V^l_{th}/T. Based on the aforementioned points, we can reformulate Eq. (8) as follows:

$$\begin{aligned}\phi^l(T) =& \frac{V^l_{th}}{T}(\sum_{i=1}^{T} S^l(i)) = \frac{V^l_{th}}{T}\lfloor\frac{T}{V^l_{th}}W^l\phi^{l-1}(T)\rfloor \\ =& \mathrm{clip}(\frac{V^l_{th}}{T}\lfloor\frac{T}{V^l_{th}}W^l\phi^{l-1}(T)\rfloor, 0, V^l_{th}),\end{aligned} \tag{9}$$

here the clip function establishes the upper limit as V^l_{th} and sets the lower limit to 0. The floor function, denoted as $\lfloor x\rfloor$, yields the largest integer not exceeding x. Under the condition where the input of the previous layer is the same for both SNN and ANN, i.e., $a^{l-1} = \phi^{l-1}(T)$, the conversion error between ANN and SNN of layer l is:

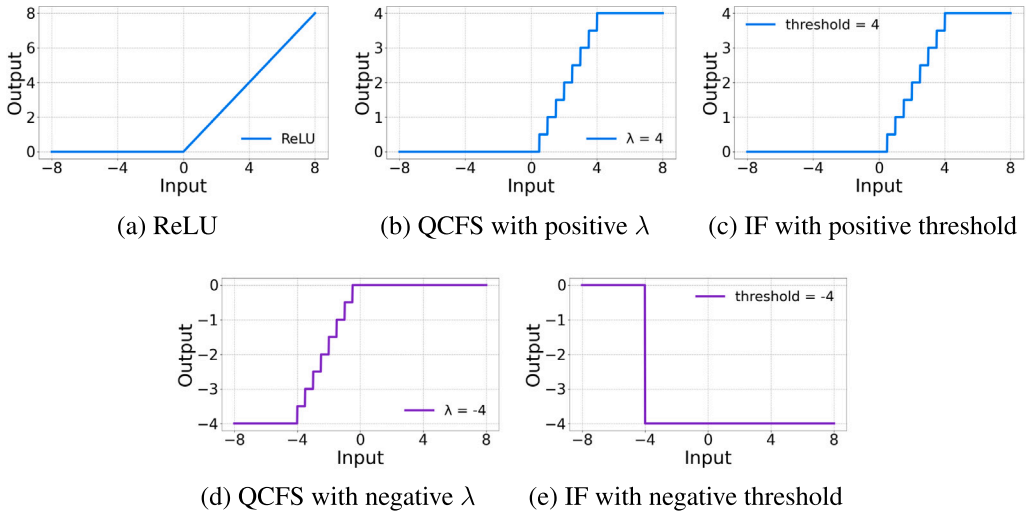
$$\begin{aligned}error^l =& \phi^l(T) - a^l \\ =& \mathrm{clip}(\frac{V^l_{th}}{T}\lfloor\frac{T}{V^l_{th}}W^l a^{l-1}\rfloor, 0, V^l_{th}) - \mathrm{ReLU}(W^l a^{l-1}).\end{aligned} \tag{10}$$

According to Eq. (10), the conversion error comes from two aspects, namely the flooring error and the clipping error (Bu et al., 2023). As shown in Fig. 1, the flooring error originates from the approximation introduced by the floor function, which leads to a mismatch between the continuous value in ANN and the quantized value in SNN. On the other hand, the clipping error is associated with the bounds set by the $Clip$ function, which restricts the output range of SNN neurons within 0 to V_{th}. In contrast, ANN neurons have an output range extending from 0 to infinity. Consequently, this disparity leads to a mismatch, as values in the ANN that exceed V_{th} lack equivalent representations in the SNN.

3.4. Eliminating conversion errors

The clipping and quantization errors can be eliminated by modifying the activation function of source ANNs. Specifically, the quantization clip-floor-shift (QCFS) activation function is proposed to replace the ReLU activation function in source ANNs, which is defined as:

$$a^l = \mathrm{QCFS}(z^l) = \lambda^l \mathrm{clip}(\frac{1}{L}\lfloor\frac{z^l L}{\lambda^l} + \varphi\rfloor, 0, 1), \tag{11}$$



(a) ReLU (b) QCFS with positive λ (c) IF with positive threshold

(d) QCFS with negative λ (e) IF with negative threshold

Fig. 2. The behavior of QCFS, ReLU, and IF.

where z^l represents the input of the lth layer, L denotes the quantization step, λ^l is the trainable threshold specific to layer l, and φ controls the shift of the activation function. The QCFS activation function is an SNN-compatible activation function from ANN, capable of eliminating the clipping error and flooring error present in the ANN-SNN conversion, and it can also adapt λ^l during training, which is mapped to the threshold V_{th}^l in SNN.

4. Method

In this section, we initially highlight the crucial importance of positive thresholds in SNNs and explain how we integrate the Softplus transformation with the QCFS activation function to ensure positive thresholds. Following this, we analyze the limitations of layer-wise threshold in the QCFS activation function and introduce our proposed Channel-wise QCFS (C-QCFS) activation function, which addresses these limitations by training individualized thresholds for each channel. Finally, we integrate the Softplus transformation with the Channel-wise QCFS to form the Channel-wise Softplus QCFS (CS-QCFS) activation function, and present our comprehensive algorithm for training the CS-QCFS activation function.

4.1. Ensuring positivity of the threshold

4.1.1. The necessity for positive thresholds in SNNs

In SNNs, a neuron's threshold holds significant physiological meaning. It essentially determines the trigger point for the neuron to emit a spike. We analyze the behavior of IF neurons in SNNs, as well as the functional dynamics of neurons in ANNs utilizing ReLU and QCFS activation functions. The distinct response patterns of these neurons are systematically illustrated in Fig. 2.

When converting ANNs to SNNs, it is imperative to set positive thresholds to ensure that SNNs functionally align with ANNs, particularly when ANNs employ ReLU or QCFS as their activation function. This alignment, as illustrated in Fig. 2(a)–2(c), not only ensures that SNN neurons mirror the activity patterns of ANN neurons, but also aligns with the fundamental principle of ANN-SNN conversion. This principle involves approximating ANN neuron outputs with the firing rates of SNN neurons.

In contrast, negative thresholds can lead to behavioral discrepancies between SNNs and ANNs. In the context of ANNs with ReLU activation function, as depicted in Fig. 2(a), a neuron is activated only when its input is greater than zero. This means that for positive inputs, the neuron contributes to the network's output, while for negative inputs, it remains inactive. In this setup, if the corresponding neuron in SNNs has a negative threshold, it could erroneously fire spikes in response to some negative inputs, a deviation clearly illustrated in Fig. 2(e). This discrepancy highlights SNN neurons can activate under conditions where their ANN counterparts would not. Similarly, for ANNs using the QCFS activation function, the situation is equally critical. As shown in Fig. 2(d), if the trainable threshold λ is negative, a neuron's output could be negative for negative inputs and zero for positive inputs. In an SNN that mirrors this ANN, negative thresholds would lead to neurons potentially firing for some negative inputs and definitely firing for positive inputs. This behavior, again depicted in Fig. 2(e), deviates from the expected behavior of the SNN model.

To further understand the implications of negative thresholds, we can formulate the error resulting from using negative thresholds mathematically. We firstly define the error in the activation function output as:

$$error^l = \mathrm{QCFS}(z^l) - \phi^l(T). \tag{12}$$

Here, the error refers to the difference between the output of ANN neuron and SNN neuron. When threshold λ^l is positive, the error is zero:

$$error^l_{\lambda^l>0} = 0. \tag{13}$$

However, when threshold λ^l is negative, the error can be described as follows:

$$error^l_{\lambda^l<0} = \begin{cases} -\lambda^l & z^l < \lambda^l, \\ \frac{\lambda^l}{L}\lfloor \frac{z^l L}{\lambda^l} \rfloor & \lambda^l \le z^l < 0, \\ \lambda^l & z^l \ge 0. \end{cases} \tag{14}$$

The above equations illustrate how the presence of negative thresholds introduces a bias in the activation function output, leading to deviations from the expected behavior.

This insight underscores the vital importance of positive thresholds in SNNs for maintaining computational and functional consistency with ANNs. However, the training process for λ in the QCFS function does not inherently guarantee this positivity, as it is unbounded and subject to gradient-based optimization techniques. Consequently, depending on the data distribution, network architecture, and learning dynamics, λ can potentially assume negative values during the iterative training process.

4.1.2. The Softplus transformation

To address this limitation and ensure non-negative thresholds, we propose replacing λ with a transformation using the Softplus function. The Softplus function is defined as:

$$\mathrm{Softplus}(x) = \log(e^x + 1), \tag{15}$$

which ensures that its output is always positive for all real-valued inputs. Mathematically, the behavior of the Softplus function varies across different regions of its domain. For large positive values of λ, it exhibits a linear behavior, effectively resembling an identity function. This makes it suitable for large-input scenarios where a linear response is desired. On the other hand, for small or negative values of λ, Softplus gently approaches zero, avoiding negative outputs and providing a soft threshold that is crucial for stable neural computations.

Another important aspect of the Softplus function is its derivative, the logistic function, which remains positive and varies between 0 and 1. This stable gradient is advantageous for gradient-based learning algorithms, preventing drastic changes during the learning process (Zheng, Yang, Liu, Liang, & Li, 2015). Furthermore, Softplus can be viewed as a smooth approximation to the ReLU function. While ReLU undergoes a sharp transition from no output (zero) to linear growth, Softplus offers a smooth and continuous transition. This characteristic combines the advantages of ReLU, such as simplicity and effectiveness in promoting sparse activations, with the additional benefit of having a well-defined gradient.

Overall, the Softplus function stands out for its smooth transition, positive outputs, and stable gradient properties, rendering it exceptionally suitable for SNNs. Its ability to handle various input ranges while maintaining computational stability is invaluable for effective and physiologically plausible neural network learning.

The QCFS activation function, incorporating the Softplus transformation, is then:

$$\begin{aligned} a_i^l &= \text{S-QCFS}(z^l) \\ &= \text{Softplus}(\lambda^l) \times \text{clip}\left(\frac{1}{L}\lfloor\frac{z^l L}{\text{Softplus}(\lambda^l)} + \phi\rfloor, 0, 1\right). \end{aligned} \tag{16}$$

Here, the key modification involves applying the Softplus function to the trainable threshold λ^l. This transformation is pivotal in ensuring that the effective threshold is always positive. Such a feature guarantees that neurons in SNNs exhibit response behaviors akin to their counterparts in ANNs, maintaining essential operational consistency between the two types of neural networks.

In summary, the integration of the Softplus function into the QCFS framework adeptly maintains the flexibility and adaptability of its trainable thresholds, while simultaneously ensuring that these thresholds consistently stay positive. Consequently, this approach not only preserves the advanced capabilities of QCFS function but also aligns the operational dynamics of SNNs more closely with those of ANNs.

4.1.3. Why Softplus

Although the exponential function ($\text{Exp}(x) = e^x$) can also ensure the positivity of λ, the Softplus function is often preferred in SNNs due to its better alignment with neural computation requirements. Softplus provides a smooth and gradual increase, making it well-suited for producing controlled responses in the threshold. This is in contrast to the exponential function's rapid escalation, which can be overly sensitive and less stable, especially for large positive inputs. Softplus also maintains computational stability with its stable gradient, crucial for effective backpropagation and avoiding issues like gradient exploding. This stability is further enhanced as Softplus typically operates in the same order of magnitude as λ, ensuring that parameters and activation values stay within similar scales.

In summary, Softplus is preferred over the exponential function in SNNs for its gradual non-linearity, computational stability, and ability to maintain a similar scale as λ. These characteristics make it particularly suitable for nuanced threshold adjustment in neural networks, ensuring positive thresholds while facilitating efficient and effective learning dynamics.

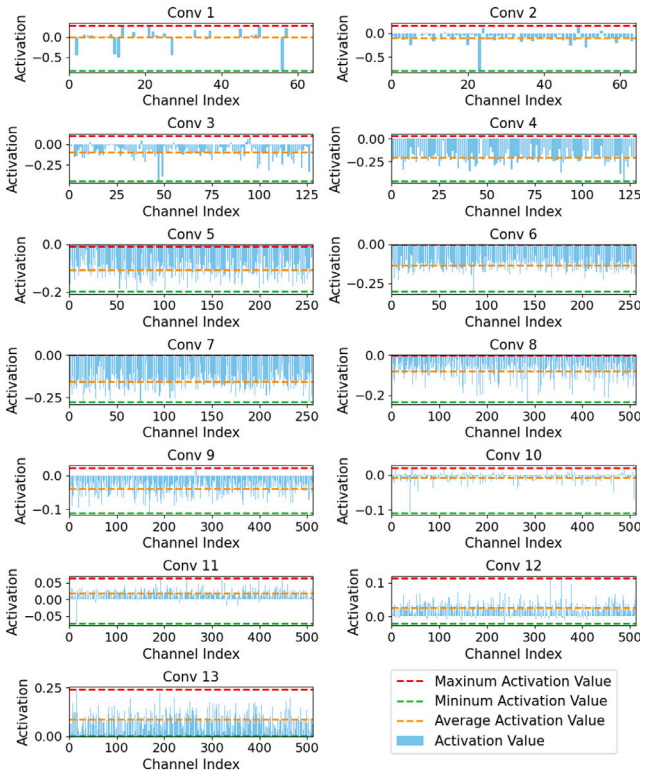


Fig. 3. The average activation values for each channel of VGG-16.

4.2. Channel-wise QCFS

4.2.1. Analysis of layer-wise limitation

The QCFS activation function, while effective in eliminating clipping and quantization errors, presents a significant limitation when applied in a layer-wise manner. In traditional deep learning architectures, different channels within a layer often capture distinct features or patterns (Zhang, Lipton, Li, & Smola, 2023). This phenomenon has been demonstrated through various visualization methods, showing that different channels capture distinct features and patterns (Olah et al., 2018; Qin, Yu, Liu, & Chen, 2018; Zeiler & Fergus, 2014). By assigning a single λ for the entire layer, it fails to account for the diverse activation distributions that exist across different channels. This homogeneity can lead to under-utilization of the representational power of some channels while over-penalizing others. Consequently, the layer-wise threshold might restrict the efficiency and potential performance of the SNN.

To confirm the existence of diverse activation distributions across different channels, we train a VGG-16 model on the CIFAR-10 dataset and calculate the average activation values for each channel across different layers of the model. The detailed results are presented in Fig. 3. Within a particular convolutional layer, there is a notable disparity in the activation values across different channels. Taking the Conv1 layer as an instance, channels like 14, 21, and 45 demonstrate normalized maximum activation values around 0.2. In contrast, channels 2, 12, 13, 27, and 56 show activation values approximating −0.5. A similar trend is observable in other convolutional layers.

4.2.2. Channel-wise threshold

To tackle the above limitation, our method proposes a shift from layer-wise QCFS function to channel-wise QCFS function. In our approach, each channel i within layer l is equipped with its distinct trainable threshold λ_i^l. The C-QCFS function is defined as follows:

$$a_i^l = \text{C-QCFS}(z^l) = \lambda_i^l \text{clip}\left(\frac{1}{L}\lfloor\frac{z^l L}{\lambda_i^l} + \phi\rfloor, 0, 1\right), \tag{17}$$

where λ_i^l is the trainable threshold for the channel i in layer l. For each channel i in layer l, the activation value a_i^l is computed using its respective λ_i^l, offering the potential for a more adaptive response based on the input distribution.

The introduction of channel-wise thresholds significantly enhances SNNs by improving their representational capacity. With each channel being able to independently adapt its threshold based on the processed input, it enables more nuanced feature extraction and improved learning dynamics. Overall, the channel-wise QCFS activation function represents a significant advancement in SNN design and functionality. By equipping each channel within a layer with its own unique and trainable threshold, this approach paves the way for more advanced, adaptive, and efficient neural computations. It brings SNNs closer to the intricate nature of biological neural processing, thus expanding the potential of SNNs in diverse applications.

4.3. Channel-wise Softplus QCFS

To address the limitations of layer-wise thresholds in QCFS and to ensure positive thresholds, we propose the Channel-wise Softplus QCFS activation function. This new activation function combines the benefits of channel-wise threshold adaptation with the positive threshold enforcement of the Softplus transformation, which is defined as follows:

$$\begin{aligned} a_i^l &= \text{CS-QCFS}(z^l) \\ &= \text{Softplus}(\lambda_i^l) \times \text{clip}\left(\frac{1}{L}\lfloor \frac{z^l L}{\text{Softplus}(\lambda_i^l)} + \phi \rfloor, 0, 1\right). \end{aligned} \tag{18}$$

All modifications to QCFS function, including channel-wise thresholds and the integration of Softplus function, are applied during the training phase of ANNs. This strategy has no impact on the subsequent ANN-SNN conversion and the operational dynamics of SNNs, thereby preserving the inherent low-energy consumption attribute of SNNs. Overall, the CS-QCFS activation function represents a significant advancement in SNN design and functionality.

4.3.1. Derivation rule for CS-QCFS

In developing the derivation rule for the CS-QCFS function, we build upon the principles established in the original QCFS function while incorporating the channel-wise thresholds and the Softplus transformation. The approach employs the straight-through estimator (Bengio, Léonard, & Courville, 2013) for the derivative of the floor function, which is defined as $\frac{d\lfloor x \rfloor}{dx} = 1$. The detailed derivation rules for CS-QCFS function are presented in Eqs. (19) and (20):

$$\frac{\partial f_{ij}(z^l)}{\partial z_{ij}^l} = \begin{cases} 1 & -\frac{\lambda_i^l}{2L} < z_{ij}^l < \lambda_i^l - \frac{\lambda_i^l}{2L}, \\ 0 & \text{otherwise}. \end{cases} \tag{19}$$

$$\frac{\partial f_{ij}(z^l)}{\partial \lambda_i^l} = \begin{cases} \frac{\text{Sigmoid}(\lambda_i^l) f(z^l) - z^l e^{\lambda_i^l}}{\text{Softplus}(\lambda_i^l)} & -\frac{\lambda_i^l}{2L} \le z_{ij}^l < \lambda_i^l - \frac{\lambda_i^l}{2L}, \\ 0 & z_{ij}^l < -\frac{\lambda_i^l}{2L}, \\ 1 & z_{ij}^l \ge \lambda_i^l - \frac{\lambda_i^l}{2L}. \end{cases} \tag{20}$$

In these equations, f denotes the CS-QCFS function, z_{ij}^l represents the jth element of the ith channel in z^l. Eq. (19) provides a simple yet effective way of computing the partial derivative of the neuron's output with respect to its input, facilitating gradient propagation through the network. Eq. (20) represents the partial derivative of the neuron's output with respect to the channel-wise threshold λ_i^l. With these derivation rules, the ANN with CS-QCFS activation function can be effectively trained using stochastic gradient descent algorithms. This training approach allows the network to learn and adjust the channel-specific thresholds λ_i^l during the learning process, enhancing the representational capacity and adaptability of the SNN. This methodology offers a significant step forward in developing more nuanced and efficient SNNs that are better aligned with the principles of biological neural processing and the computational paradigms of ANNs.

4.3.2. Gradual training for CS-QCFS

Directly training the channel-wise thresholds without prior knowledge could lead to poor convergence or get trapped in undesirable local minima. The primary reason for this is the sensitivity to initial values in an expanded parameter space. When training individual thresholds for each channel, the parameter space expands, which may potentially destabilize the early stages of training. Additionally, employing a uniform starting threshold across all channels of different layers fails to address the distinct characteristics between layers from the outset. Without the context provided by prior layer-wise training, this uniform approach is less effective for guiding the optimal evolution of channel-specific thresholds.

Algorithm 1 ANN-SNN Conversion of CS-QCFS.

Input: ANN Model $M_{\text{ANN}}(x; W)$ with initial weight W; Dataset D
Parameter: Quantization step L; Initial dynamic thresholds λ_0; Learning rate ϵ; Finetune learning rate ϵ_{ft}
Output: $M_{\text{SNN}}(x; \hat{W})$

for $l = 1$ to M_{ANN}.layers **do**
  if is ReLU Activation **then**
    Replace ReLU(x) by QCFS($x; L, \lambda_0$) (Eq. (11))
  end if
end for
Train layer-wise threshold λ^l in QCFS using SGD with lr=ϵ
for $l = 1$ to M_{ANN}.layers **do**
  if is QCFS **then**
    Replace QCFS($x; L, \lambda_0$) by CS-QCFS($x; L, \lambda^l$) (Eq. (18))
  end if
end for
Finetune channel-wise threshold λ_i^l in CS-QCFS using SGD with lr=ϵ_{ft}
for $l = 1$ to M_{ANN}.layers **do**
  $M_{\text{SNN}}.\hat{W}^l \longleftarrow M_{\text{ANN}}.W^l$ ▷ Transfer the weights from ANN to SNN
  for $c = 1$ to $M_{\text{ANN}}.l_i$.channels **do**
    $M_{\text{SNN}}.V_i^l \longleftarrow M_{\text{ANN}}.\lambda_i^l$ ▷ Set SNN's threshold
    $M_{\text{SNN}}.v_i^l(0) \longleftarrow M_{\text{SNN}}.V_i^l/2$ ▷ Initialize SNN's voltage
  end for
end for
return M_{SNN}

To alleviate this, a stepwise training strategy is adopted: Initially, we train the model with the layer-wise QCFS activation function. This provides a general understanding of an approximate threshold suitable for the entire layer. Subsequently, using the derived layer-wise threshold λ^l as the starting point for channel-wise threshold λ_i^l in the CS-QCFS activation function, the network then undergoes fine-tuning. This approach ensures that each channel begins its fine-tuning phase with a threshold value that is contextually informed by prior layer-wise training. The ANN-SNN conversion is shown in Algorithm 1.

By integrating prior context from the layer-wise training, this coarse-to-fine optimization approach ensures a smoother and more informed optimization landscape for each channel, leading to faster and potentially more robust convergence. The integrated approach is anticipated to yield a more robust and efficient conversion from ANNs to SNNs.

5. Experiments

In this section, we validate the efficacy and scalability of our method on image classification datasets, specifically CIFAR-10 and CIFAR-100 (Krizhevsky, Hinton, et al., 2009). The network architectures we employed include ResNet-18, ResNet-20, ResNet-50, ResNet-101 (He, Zhang, Ren, & Sun, 2016), and VGG-16 (Simonyan & Zisserman, 2014). Additionally, we benchmark our approach against previous state-of-the-art ANN-SNN conversion methods, including QCFS (Bu et al., 2023), SRP (Hao, Bu, et al., 2023), and COS (Hao, Ding, et al., 2023).

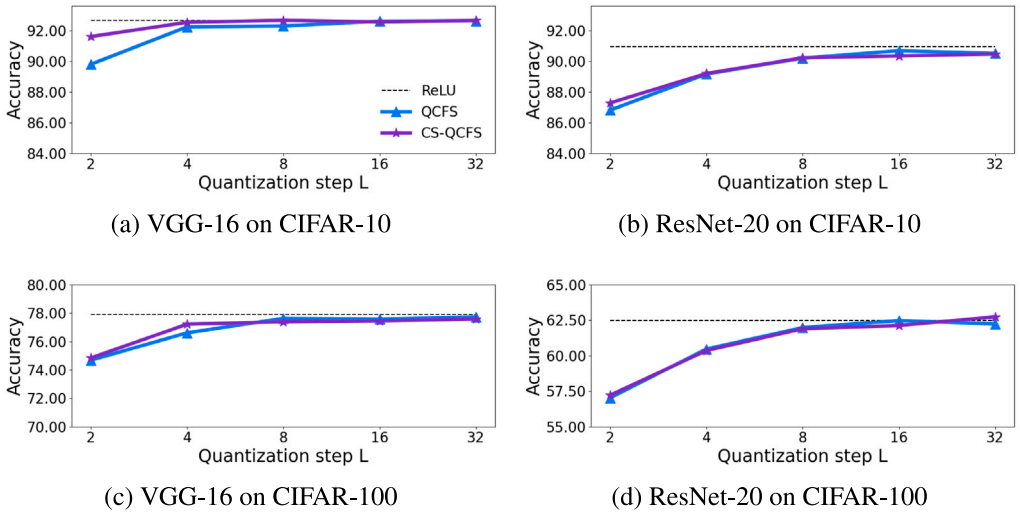


Fig. 4. Comparison of ANNs' accuracy.

5.1. Experimental settings

Training and Optimization: For training the ANN models, we employ the stochastic gradient descent optimizer (Bottou, 2012) with a momentum parameter of 0.9. The training is conducted in two phases. Initially, models are trained at a base learning rate of 0.1 for CIFAR-10 and 0.05 for CIFAR-100 for 300 epochs. When training ResNet-50 and ResNet-101, the base learning rate is set to 0.01 on both CIFAR-10 and CIFAR-100. Following this, we implement a fine-tuning phase where the learning rate is reduced to 1/10 of the initial rate and the training continues for an additional 100 epochs. The learning rate is modulated using the cosine annealing scheduler (Loshchilov & Hutter, 2016). Additionally, a weight decay of 5×10^{-4} is applied to both datasets to regularize the training.

Data Augmentation: To enhance model performance (Yang, Guo, Zhao, & Shen, 2024), we utilize random crop images, Cutout (DeVries & Taylor, 2017), and AutoAugment (Cubuk, Zoph, Mane, Vasudevan, & Le, 2019) across all datasets.

CS-QCFS parameters: For our experiments, the quantization step L is meticulously calibrated for each dataset. Specifically, on the CIFAR-10 dataset, we consistently set L to 4 across all network architectures, including ResNet-18, ResNet-20, ResNet-50, ResNet-101, and VGG-16. For the CIFAR-100 dataset, L is set to 4 for VGG-16, ResNet50, and ResNet101, and 8 for ResNet-20. Furthermore, We set the initial threshold to 4 for experiments on the CIFAR-10 dataset and for ResNet-50, ResNet-101, and VGG-16 on the CIFAR-100 dataset, while it is set to 8 for ResNet-20 on the CIFAR-100 dataset.

5.2. Accuracy of ANN with CS-QCFS

We first evaluate the performance of ANNs integrated with the original QCFS activation function, the proposed CS-QCFS activation function, and the standard ANNs with ReLU activation function. Fig. 4 presents the results of VGG-16 on CIFAR-10, ResNet-20 on CIFAR-10, VGG-16 on CIFAR-100, and ResNet-20 on CIFAR-100, respectively.

It can be observed that ANNs with our CS-QCFS activation function exhibit comparable performance to those using the original QCFS activation function. Remarkably, both the original QCFS activation function and our CS-QCFS activation function demonstrate similar levels of performance to the conventional ANNs with ReLU activation function when $L > 4$. This underscores that integrating the CS-QCFS activation function does not compromise the efficiency of ANNs. Therefore, CS-QCFS activation function is a more suitable choice for ANN-SNN conversion as it not only preserves the performance of ANNs but also, while retaining the original features of QCFS activation function, aligns the behavior of SNNs more closely with that of ANNs.

5.3. Comparison with the state-of-the-art

We compare our method with previous state-of-the-art ANN-SNN conversion methods, including QCFS, SRP, and COS, to validate the effectiveness of our method. In our approach, we also utilize offset spikes (Hao, Ding, et al., 2023) to further reduce the conversion errors.

In Table 1, we present the test accuracy of these methods on the CIFAR-10 dataset under various time-steps. For ResNet-18, our method achieves 95.86% accuracy at $T = 1$, surpassing all other methods even at $T = 32$. At $T = 32$, our accuracy improves to 96.51%. With VGG-16, our method consistently outperforms others, achieving 95.70% at $T = 4$, which is better than the other methods at $T = 32$. For ResNet-20, our method starts at 91.24% accuracy at $T = 1$, outperforming COS (89.88%) and SRP (86.37%). By $T = 32$, it reaches 92.41%, ahead of COS (92.16%) and SRP (91.72%). With ResNet-50, our method achieves 96.72% at $T = 1$ and improves to 96.90% at $T = 32$, consistently leading. For ResNet-101, our method starts strong with 96.66% at $T = 1$ and maintains this performance at $T = 32$, outperforming QCFS, SRP, and COS at every time-step. These results underline the efficiency and efficacy of our method, demonstrating its ability to achieve high accuracy in fewer time-steps and maintain consistent performance across various network architectures.

We expand our experiments to assess our method's generalization capabilities on the CIFAR-100 dataset, the results are shown in Table 2. For VGG-16, our method consistently outperforms others across all time-steps, achieving 76.87% accuracy at $T = 4$ and 77.64% at $T = 32$, a 0.6% improvement over the nearest competitor, QCFS. For ResNet-20, our method registers 62.47% accuracy at $T = 1$, significantly higher than SRP's 48.33% and COS's 59.43%, and maintains its superiority across subsequent time-steps. For ResNet-50, our method achieves 78.25% accuracy at $T = 1$, surpassing QCFS and SRP even at higher time-steps. At $T = 32$, it reaches 81.60%, the highest among all methods. For ResNet-101, our method starts with 78.28% accuracy at $T = 1$, outperforming QCFS and SRP. While COS achieves slightly better accuracy at $T = 32$, our method remains competitive overall. Therefore, the findings from the CIFAR-100 dataset experiments further validate our method's generalization capabilities and its consistent performance superiority, especially in ultra-low latency.

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

Architecture	Method	ANN*	T = 1	T = 2	T = 4	T = 8	T = 16	T = 32
ResNet-18	QCFS (Bu et al., 2023)	95.64%	–	91.75%	93.83%	95.04%	95.56%	95.67%
	SRP (Hao, Bu, et al., 2023)	95.67%	94.59%	95.06%	95.25%	95.60%	95.55%	95.55%
	COS (Hao, Ding, et al., 2023)	95.64%	95.25%	95.45%	95.46%	95.66%	95.68%	95.68%
	Ours	**96.48%**	**95.86%**	**96.24%**	**96.34%**	**96.41%**	**96.47%**	**96.51%**
VGG-16	QCFS (Bu et al., 2023)	95.52%	–	91.18%	93.96%	94.95%	95.40%	95.54%
	SRP (Hao, Bu, et al., 2023)	95.52%	93.80%	94.47%	95.32%	95.52%	95.44%	95.42%
	COS (Hao, Ding, et al., 2023)	95.51%	94.90%	95.36%	95.46%	95.51%	95.57%	95.61%
	Ours	**95.67%**	**94.93%**	**95.47%**	**95.70%**	**95.78%**	**95.93%**	**95.88%**
ResNet-20	QCFS (Bu et al., 2023)	91.77%	–	73.20%	83.75%	89.55%	91.62%	92.24%
	SRP (Hao, Bu, et al., 2023)	91.77%	86.37%	88.73%	90.51%	91.37%	91.64%	91.72%
	COS (Hao, Ding, et al., 2023)	91.77%	89.88%	91.26%	91.68%	91.86%	92.20%	92.16%
	Ours	**91.80%**	**91.24%**	**91.66%**	**91.74%**	**92.00%**	**92.29%**	**92.41%**
ResNet-50	QCFS (Bu et al., 2023)	96.81%	69.07%	84.29%	91.20%	94.30%	95.99%	96.58%
	SRP (Hao, Bu, et al., 2023)	96.81%	81.01%	92.24%	95.34%	95.91%	96.13%	96.22%
	COS (Hao, Ding, et al., 2023)	96.81%	95.99%	96.55%	96.58%	96.58%	96.65%	96.70%
	Ours	**96.86%**	**96.72%**	**96.62%**	**96.70%**	**96.76%**	**96.82%**	**96.90%**
ResNet-101	QCFS (Bu et al., 2023)	96.74%	32.17%	59.32%	84.02%	91.75%	94.33%	95.80%
	SRP (Hao, Bu, et al., 2023)	96.74%	50.54%	82.14%	91.48%	94.09%	94.89%	95.19%
	COS (Hao, Ding, et al., 2023)	96.74%	96.58%	96.59%	96.61%	96.55%	96.55%	96.60%
	Ours	**96.92%**	**96.66%**	**96.81%**	**96.83%**	**96.77%**	**96.67%**	**96.66%**

ANN* is the accuracy after replacing ReLU with QCFS/CS-QCFS that has less conversion error.

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

Architecture	Method	ANN	T = 1	T = 2	T = 4	T = 8	T = 16	T = 32
VGG-16	QCFS (Bu et al., 2023)	76.28%	–	63.79%	69.62%	73.96%	76.24%	77.01%
	SRP (Hao, Bu, et al., 2023)	76.28%	71.52%	74.31%	75.42%	76.25%	76.42%	76.45%
	COS (Hao, Ding, et al., 2023)	76.28%	74.24%	76.03%	76.26%	76.52%	76.77%	76.96%
	Ours	**76.84%**	**74.83%**	**76.56%**	**76.87%**	**77.26%**	**77.66%**	**77.64%**
ResNet-20	QCFS* (Bu et al., 2023)	68.30%	14.37%	20.91%	35.21%	55.47%	66.18%	68.52%
	SRP* (Hao, Bu, et al., 2023)	68.30%	48.33%	55.48%	60.28%	63.20%	64.81%	65.26%
	COS* (Hao, Ding, et al., 2023)	68.30%	59.43%	63.62%	64.70%	66.17%	67.86%	68.74%
	Ours	**68.54%**	**62.47%**	**64.91%**	**65.35%**	**66.61%**	**68.28%**	**68.94%**
ResNet-50	QCFS (Bu et al., 2023)	80.82%	38.55%	48.47%	58.75%	69.48%	76.57%	79.88%
	SRP (Hao, Bu, et al., 2023)	80.82%	55.98%	71.21%	73.81%	75.35%	76.25%	76.65%
	COS (Hao, Ding, et al., 2023)	80.82%	77.02%	80.27%	80.53%	80.71%	80.94%	81.10%
	Ours	**81.10%**	**78.25%**	**80.78%**	**80.80%**	**81.07%**	**81.38%**	**81.60%**
ResNet-101	QCFS (Bu et al., 2023)	80.53%	19.41%	32.06%	44.43%	58.72%	70.11%	76.81%
	SRP (Hao, Bu, et al., 2023)	80.53%	7.14%	53.54%	61.23%	65.80%	68.49%	69.82%
	COS (Hao, Ding, et al., 2023)	80.53%	77.89%	**80.13%**	80.23%	80.34%	**80.22%**	**80.29%**
	Ours	**80.66%**	**78.28%**	80.11%	**80.23%**	**80.44%**	80.17%	80.15%

The results marked with * are those we reproduce based on the code provided by the authors, with experimental settings identical to our method.

Table 3
Effect of channel-wise threshold and Softplus function.

Architecture		ANN	T = 1	T = 2	T = 4	T = 8	T = 16	T = 32
VGG-16	w/o channel	95.37%	94.68%	95.20%	95.33%	95.42%	95.50%	95.57%
	w/o Softplus	95.36%	94.91%	95.16%	95.31%	95.45%	95.41%	95.44%
	Full	95.67%	94.93%	95.47%	95.70%	95.78%	95.93%	95.88%
ResNet-20	w/o channel	91.48%	90.04%	91.06%	91.29%	91.68%	92.10%	92.13%
	w/o Softplus	91.35%	90.69%	91.05%	91.25%	91.49%	91.80%	92.01%
	Full	91.80%	91.24%	91.66%	91.74%	92.00%	92.29%	92.41%

5.4. Ablation study

In this section, we evaluate the individual impacts of the key components in our method: channel-wise threshold, the incorporation of the Softplus function, and the two-stage training.

5.4.1. Effect of the channel-wise threshold and Softplus

To test the potential of channel-wise threshold and the introduction of the Softplus function within the QCFS function, we contrast the standard QCFS function, channel-wise QCFS function (w/o Softplus), Softplus QCFS function (w/o channel) and channel-wise Softplus QCFS function (Full). The experiment results are shown in Table 3. Excluding the channel mechanism (w/o channel) results in a slight performance decrease across different time-steps as compared to the full model. This underscores the importance of channel-wise threshold, highlighting its role in allowing the network to capture and harness diverse channel-specific features more effectively. On the other hand, removing the Softplus function (w/o Softplus) leads to a subtle reduction in accuracy, implying that the Softplus function plays a crucial role in ensuring more stable and consistent activation thresholds.

In conclusion, the results robustly validate the hypothesis that both the channel-wise threshold and the Softplus function individually contribute to enhanced model performance.

Table 4
Effect of two-stage training.

Dataset	Architecture		ANN	T = 1	T = 2	T = 4	T = 8	T = 16	T = 32
CIFAR-10	VGG-16	origin	95.30%	94.49%	95.01%	95.26%	95.33%	95.47%	95.48%
		layer-wise	95.50%	94.67%	95.24%	95.47%	95.52%	95.62%	95.63%
		channel-wise	95.43%	95.01%	95.30%	95.35%	95.47%	95.51%	95.48%
	ResNet-20	origin	91.41%	89.82%	90.95%	91.27%	91.58%	92.00%	92.17%
		layer-wise	91.85%	90.03%	91.31%	91.65%	92.02%	92.36%	92.41%
		channel-wise	91.53%	90.87%	91.36%	91.47%	91.65%	92.02%	92.21%
CIFAR-100	VGG-16	origin	76.57%	74.67%	76.16%	76.52%	76.90%	77.11%	77.17%
		layer-wise	76.46%	74.83%	76.04%	76.43%	76.68%	77.01%	77.12%
		channel-wise	76.52%	75.53%	76.24%	76.46%	76.70%	77.02%	77.07%
	ResNet-20	origin	68.82%	59.88%	64.15%	65.38%	66.60%	68.56%	69.19%
		layer-wise	69.18%	59.89%	64.12%	65.47%	66.94%	68.86%	69.60%
		channel-wise	68.77%	62.46%	65.13%	65.75%	66.86%	68.44%	69.20%

5.4.2. Effect of two-stage training

The concept of two-stage training is also investigated in our ablation study, addressing a fundamental question: Does the performance improvement stem from the channel-wise strategy, or is it simply a result of employing the two-stage training process?

To answer this question, we perform a comparative analysis using three different training strategies. The first strategy is layer-level one-stage training, where a network with layer-wise QCFS activation function is trained straightforwardly in a single phase. The second strategy is layer-level two-stage training, which begins with a standard training phase for networks with layer-wise QCFS activation function, followed by a fine-tuning phase with a lower learning rate. The last strategy is channel-level two-stage training, which initially trains a network with layer-wise QCFS activation functions and then transitions to a fine-tuning phase where the network utilizes channel-wise QCFS activation functions.

As shown in Table 4, "origin", "layer-wise" and "channel-wise", respectively, denote the layer-level one-stage training, layer-wise two-stage training and channel-wise two-stage training. The CIFAR-10 results show that the two-stage training at the layer level does indeed improve the ANN accuracy, the corresponding improvement in SNN accuracy is only marginal. Notably, channel-wise two-stage training resulted in more significant improvements for low-latency scenarios. Specifically, for ResNet-20, although the SNN accuracy with layer-wise two-stage training improved from 89.82% to 90.03% for T = 1, the channel-wise two-stage training achieved a higher SNN accuracy of 90.87% for T = 1. This observation remains consistent when examining the CIFAR-100 results, reinforcing the same conclusion.

To sum up, the experiment results indicate that while two-stage training may offer some marginal benefits, the primary driver for enhanced performance in low-latency scenarios is the application of channel-wise refinement. This refinement enables more tailored adjustments, effectively capturing the intricate features within each channel, and ultimately leading to a more capable and adaptable neural network model.

5.5. Effect of quantization steps L

Similar to QCFS function, in our proposed CS-QCFS method, the parameter L represents the hyperparameter known as the quantization steps. Its role in the quantization process directly influences the accuracy and performance of the converted SNNs. To thoroughly investigate the influence of varying quantization step values L and identify the optimal setting, we conduct a series of experiments. This involves integrating our CS-QCFS activation function into VGG-16 and ResNet-20 networks with a diverse range of L values, specifically 2, 4, 8, 16, and 32. Following the training, these networks are then converted into SNNs, allowing us to evaluate how different quantization levels affect the networks' performance within the spiking domain, providing valuable insights into the optimal quantization strategy for SNNs. To achieve an optimal balance between low latency and high accuracy in SNNs, our evaluation is focused on two key aspects. Firstly, we examine the accuracy of the SNNs at ultra-low latency, specifically within 4 time-steps. Secondly, we assess the peak accuracy that the SNNs could achieve. The experimental results depicted in Fig. 5 highlight the performance of converted SNNs on the CIFAR-10/100 dataset.

The experimental results indicate a clear pattern where the accuracy of SNNs at ultra-low latency decreases as the quantization step value L increases. Conversely, an overly small L can compromise the model's capacity, leading to diminished accuracy. Notably, with $L = 2$, a pronounced divergence in the best accuracy of the SNN from the original ANN is observed. However, when L is increased beyond 4, the SNN's top accuracy more closely aligns with the source ANN. From these observations, it becomes apparent that the optimal selection of L depends largely on the trade-off between achieving low latency and maximizing accuracy. Based on our analysis, we recommend setting L to either 4 or 8, as these values allow for a high-performing converted SNN that excels in both low and high time-steps.

In summary, our ablation study elucidates the merits of channel-wise adjustments, the Softplus introduction, and the strategic nuances of two-stage training. Collectively, these modifications confront and rectify inherent limitations in the traditional QCFS methodology, paving the way for more potent and efficient SNN implementations.

5.6. Necessity analysis

To ensure that our modifications to the QCFS function are not just empirically effective but also fundamentally indispensable, we carry out a necessity analysis. This section is bifurcated into an evaluation of the indispensability of channel-wise adjustment and the introduction of Softplus.

5.6.1. Necessity of channel-wise threshold

The primary objective behind introducing channel-wise thresholds is to address potential variations across different channels. To determine the necessity of channel-wise threshold, we focus on quantifying the disparity in thresholds among various channels within the network's layers. The experiments integrate C-QCFS and CS-QCFS as activation functions within VGG-16 and ResNet-20 architectures and then conduct evaluations on the CIFAR-10 and CIFAR-100 datasets. The results of the experiments are summarized in Table 5. The $MaxDiff$ metric is defined as the maximum value of $Diff$ across all layers in the model, where $Diff$ represents the proportional difference between the highest and lowest threshold values within a single layer, calculated as $Diff = (MaxThre - MinThre)/MinThre$. Here, $MaxThre$ is the maximum threshold value, and $MinThre$ is the minimum threshold value within a layer's channels. Similarly, the $AvgDiff$ metric calculates the average of these $Diff$ values across all layers.

Large values of $MaxDiff$ and $AvgDiff$ are observed across all models and datasets for both C-QCFS and CS-QCFS. For instance, with C-QCFS on the CIFAR-10 dataset, the VGG-16 model shows a $MaxDiff$ of 339.7% and an $AvgDiff$ of 153.4%. Even with the integration

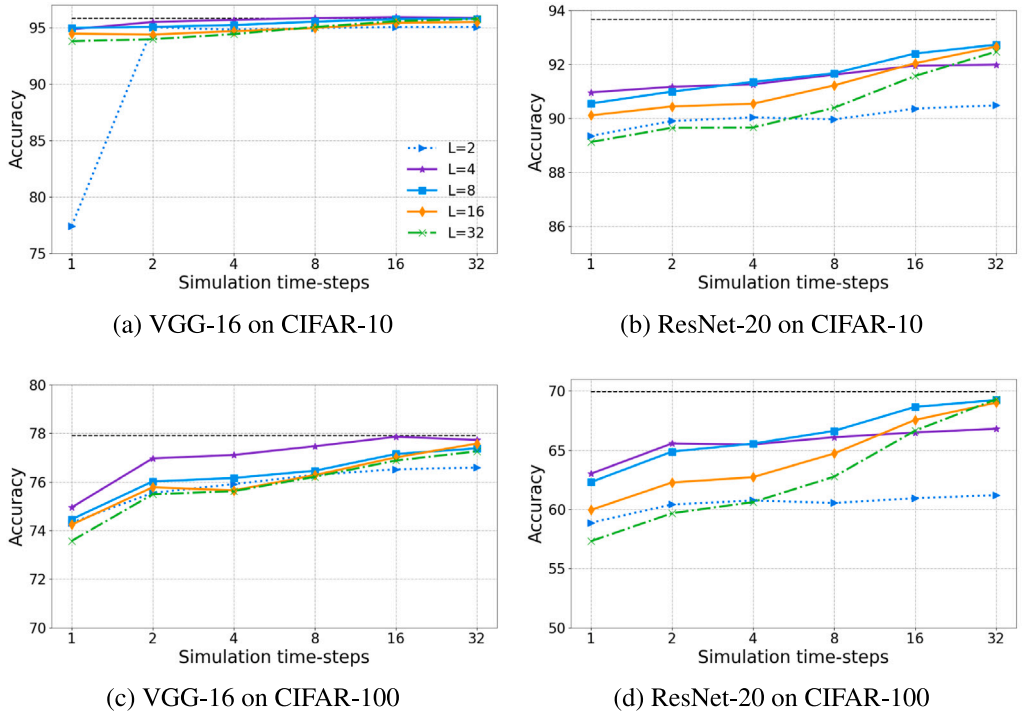


Fig. 5. Influence of different quantization steps.

Table 5
Necessity experiment of channel-wise threshold.

Activation	Dataset	Architecture	$MaxDiff$	$AvgDiff$
C-QCFS	CIFAR-10	VGG-16	339.7%	153.4%
		ResNet-20	155.3%	68.2%
	CIFAR-100	VGG-16	246.2%	99.5
		ResNet-20	98.4%	45.5%
CS-QCFS	CIFAR-10	VGG-16	55.3%	10.5%
		ResNet-20	38.2%	16.1%
	CIFAR-100	VGG-16	65.8%	14.0%
		ResNet-20	44.4%	16.9%

of Softplus function in CS-QCFS, notable differences remain, as seen with a $MaxDiff$ of 65.8% and an $AvgDiff$ of 14.0% for the VGG-16 model on the CIFAR-100 dataset. The Softplus function's gradual, non-linear response to input variances contributes to more balanced threshold adaptations across channels. Its inherent characteristic of ensuring positive thresholds moderates the range of threshold values, thus reducing extreme variations. The persistence of substantial $MaxDiff$ and $AvgDiff$ in both C-QCFS and CS-QCFS underscores the inherent variability across channels. This observation highlights the fact that uniform thresholds could potentially limit the network's learning capabilities and representational power. Consequently, it reinforces the necessity of adopting channel-wise thresholds, a strategy that can fully exploit network's potential by accommodating the varied feature processing demands of different channels.

5.6.2. Necessity of the Softplus function

The integration of the Softplus function aims to resolve the potential issue of negative thresholds that could arise in QCFS function. To discern its necessity, we analyze the thresholds derived from the networks without the Softplus adaptation.

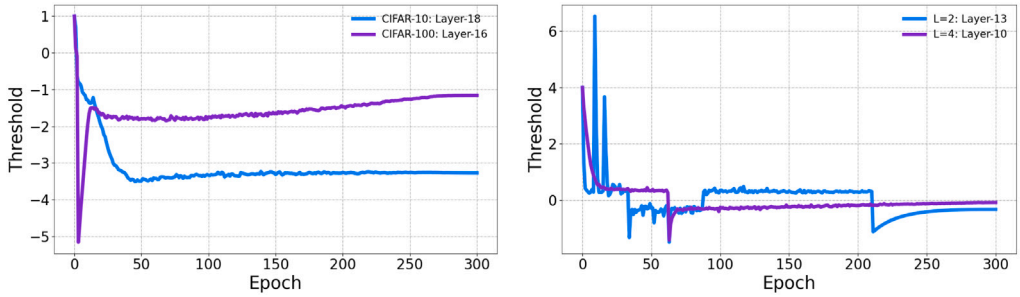
In our experiments, training the ResNet-20 model on both CIFAR-10 and CIFAR-100 datasets reveals the occurrence of negative thresholds, especially when starting with lower initial threshold. This phenomenon is illustrated in Fig. 6(a). Moreover, following the initial threshold suggested by QCFS, we encounter negative thresholds under various settings of the hyperparameter L in the training of VGG-16 models on the CIFAR-10 dataset, as shown in Fig. 6(b). This finding is crucial as it demonstrates that without any rectifying measures, the training process can indeed yield values for threshold that are not just suboptimal but fundamentally incompatible with the principle of ANN-SNN conversion.

The consistent emergence of negative thresholds, regardless of the datasets or model architectures, confirms the essentiality of the Softplus function. By reparameterizing threshold λ as $Softplus(\lambda)$, it consistently guarantees the positivity of thresholds, thereby aligning with the core idea of ANN-SNN conversion. This correction is not just a preventive enhancement but a vital fix to a fundamental flaw that could otherwise undermine the reliability and applicability of SNNs in practical scenarios.

To encapsulate, our necessity analysis substantiates that both channel-wise adjustment and the Softplus function are not mere augmentations but are crucial to the effective and consistent operation of SNNs.

5.7. Training cost and energy consumption analysis

To evaluate the learning cost of CS-QCFS, we compare the average training time per epoch on the CIFAR-100 dataset using ResNet-50 between our method and previous methods. While ideally, we would compare CS-QCFS against QCFS, SRP, and COS, it is important to note that SRP and COS are both based on QCFS. Therefore, we focus on



(a) ResNet-20 on CIFAR-10 and CIFAR-100 (b) VGG-16 on CIFAR-10

Fig. 6. Negative threshold in the training.

Table 6
Comparison of the learning cost.

Activation	Average time (s)
QCFS	110
CS-QCFS	111.81(↑1.65%)

Table 7
Comparison of the energy consumption.

Model	ACs (M)	MACs (M)	Energy (mJ)
ANN	0	1310	6.026
SNN_{QCFS}	42.87	8.65	0.078
$SNN_{CS-QCFS}$	52.66	8.43	0.0862(↑10.51%)

comparing CS-QCFS with QCFS. The experiments are conducted on NVIDIA RTX2080TI GPUs. The results are summarized in Table 6. As shown in Table 6, the average training time per epoch for QCFS is 110 s, whereas for CS-QCFS, it is 111.81 s, indicating a slight increase of 1.65%. This minor increase demonstrates that the additional computational complexity introduced by CS-QCFS is minimal, making it a viable option considering its potential benefits.

We also evaluate the energy consumption of our method and QCFS on CIFAR-100 datasets. Here we use the same network structure of ResNet-50. Following the analysis in Yao et al. (2023), we use Accumulation (AC) and Multiply-and-Accumulate (MAC) operations to represent the required basic operation numbers to classify one image. We assume the data for various operations are 32-bit floating point implementation in 45 nm technology (Horowitz, 2014), in which E_{MAC} = 4.6 pJ and E_{AC} = 0.9 pJ. Note that we do not consider the memory access energy in our study because it depends on the hardware.

As shown in Table 7, the ANN model has no AC operations and requires 1310 million MAC operations, consuming 6.026 mJ. The SNN_{QCFS} model performs 42.87 million AC operations and 8.65 million MAC operations, resulting in an energy consumption of 0.078 mJ. In comparison, the $SNN_{CS-QCFS}$ model performs 52.66 million AC operations and 8.43 million MAC operations, with a total energy consumption of 0.0862 mJ, representing a 10.51% increase. Although CS-QCFS introduces a minor increase in energy consumption, it achieves higher accuracy.

6. Conclusions and future work

In this paper, we have made notable contributions to the field of ANN-SNN conversion by addressing two significant limitations within the QCFS function. Our comprehensive analysis has shed light on critical issues concerning channel variability and negative thresholds in SNNs. To overcome these challenges, we introduce channel-wise adaptive thresholds as an innovative solution. This approach accommodates the inherent variability among channels within layers by training individual thresholds for each channel. As a result, it enables more precise feature extraction and enhances the network's representational capacity. Additionally, we incorporate the Softplus transformation to ensure the positivity of thresholds. This adaptation not only aligns the behavior of SNNs more closely with that of ANNs but also improves the overall performance of SNNs. Our modifications to the QCFS function significantly enhance the efficacy of SNNs and set new performance benchmarks on CIFAR datasets. This advancement narrows the performance gap between SNNs and ANNs while preserving the biologically plausible modeling intrinsic to SNNs. Our proposed method holds promise for advancing the practical application of SNNs, particularly in the context of neuromorphic computing, where it can facilitate the development of more efficient and adaptive neural networks.

While CS-QCFS is applicable to all networks that employ ReLU as activation functions, future work will involve extending its application to more models (e.g., Transformer), and a wider variety of tasks such as object detection. This expansion will contribute to the broader understanding and implementation of SNNs in diverse computational scenarios.

CRediT authorship contribution statement

Hongchao Yang: Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation. **Suorong Yang:** Writing – review & editing, Methodology. **Lingming Zhang:** Writing – review & editing. **Hui Dou:** Writing – review & editing. **Furao Shen:** Writing – review & editing, Supervision, Project administration. **Jian Zhao:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the STI 2030-Major Projects of China under Grant 2021ZD0201300, the National Science Foundation of China under Grant 62276127, and the Fundamental Research Funds for the Central Universities under Grant 2024300394.

Data availability

Data will be made available on request.

References

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

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

Bu, Tong, Fang, Wei, Ding, Jianhao, Dai, PengLin, Yu, Zhaofei, & Huang, Tiejun (2023). Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. arXiv preprint arXiv:2303.04347.

Burkitt, Anthony N. (2006). A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biological Cybernetics*, *95*, 1–19.

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

Caporale, Natalia, & Dan, Yang (2008). Spike timing–dependent plasticity: a Hebbian learning rule. *Annual Review of Neuroscience*, *31*, 25–46.

Cubuk, Ekin D, Zoph, Barret, Mane, Dandelion, Vasudevan, Vijay, & Le, Quoc V (2019). Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 113–123).

Davies, Mike, Srinivasa, Narayan, Lin, Tsung-Han, Chinya, Gautham, Cao, Yongqiang, Choday, Sri Harsha, et al. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, *38*(1), 82–99.

Deng, Shikuang, & Gu, Shi (2021). Optimal conversion of conventional artificial neural networks to spiking neural networks. arXiv preprint arXiv:2103.00476.

DeVries, Terrance, & Taylor, Graham W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552.

Diehl, Peter U, Neil, Daniel, Binas, Jonathan, Cook, Matthew, Liu, Shih-Chii, & Pfeiffer, Michael (2015). Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 international joint conference on neural networks* (pp. 1–8). ieee.

Ding, Jianhao, Yu, Zhaofei, Tian, Yonghong, & Huang, Tiejun (2021). Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. arXiv preprint arXiv:2105.11654.

Han, Bing, Srinivasan, Gopalakrishnan, & Roy, Kaushik (2020). 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* (pp. 13558–13567).

Hao, Zecheng, Bu, Tong, Ding, Jianhao, Huang, Tiejun, & Yu, Zhaofei (2023). Reducing ANN-SNN conversion error through residual membrane potential. arXiv preprint arXiv:2302.02091.

Hao, Zecheng, Ding, Jianhao, Bu, Tong, Huang, Tiejun, & Yu, Zhaofei (2023). Bridging the gap between ANNs and SNNs by calibrating offset spikes. arXiv preprint arXiv:2302.10685.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, & Sun, Jian (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Ho, Nguyen-Dong, & Chang, Ik-Joon (2021). TCL: an ANN-to-SNN conversion with trainable clipping layers. In *2021 58th ACM/IEEE design automation conference* (pp. 793–798). IEEE.

Horowitz, Mark (2014). 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers* (pp. 10–14). IEEE.

Kim, Seijoon, Park, Seongsik, Na, Byunggook, & Yoon, Sungroh (2020). Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 34*, (07), (pp. 11270–11277).

Krizhevsky, Alex, Hinton, Geoffrey, et al. (2009). Learning multiple layers of features from tiny images.

Li, Yuhang, Deng, Shikuang, Dong, Xin, & Gu, Shi (2022). Converting artificial neural networks to spiking neural networks via parameter calibration. arXiv preprint arXiv:2205.10121.

Liu, Fangxin, Zhao, Wenbo, Chen, Yongbiao, Wang, Zongwu, & Jiang, Li (2022). Spikeconverter: An efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*: *vol. 36*, (2), (pp. 1692–1701).

Loshchilov, Ilya, & Hutter, Frank (2016). Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983.

Maass, Wolfgang (1997). Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, *10*(9), 1659–1671.

Merolla, Paul A, Arthur, John V, Alvarez-Icaza, Rodrigo, Cassidy, Andrew S, Sawada, Jun, Akopyan, Filipp, et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*(6197), 668–673.

Neftci, Emre O., Mostafa, Hesham, & Zenke, Friedemann (2019). Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, *36*(6), 51–63.

Olah, Chris, Satyanarayan, Arvind, Johnson, Ian, Carter, Shan, Schubert, Ludwig, Ye, Katherine, et al. (2018). The building blocks of interpretability. *Distill*, *3*(3), Article e10.

Qin, Zhuwei, Yu, Fuxun, Liu, Chenchen, & Chen, Xiang (2018). How convolutional neural network see the world-a survey of convolutional neural network visualization methods. arXiv preprint arXiv:1804.11191.

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

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

Simonyan, Karen, & Zisserman, Andrew (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Wang, Yuchen, Zhang, Malu, Chen, Yi, & Qu, Hong (2022). Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *International joint conference on artificial intelligence*.

Weisstein, Eric W. (2002). *Heaviside step function*. Wolfram Research, Inc., https://mathworld.wolfram.com/.

Yang, Suorong, Guo, Suhan, Zhao, Jian, & Shen, Furao (2024). Investigating the effectiveness of data augmentation from similarity and diversity: An empirical study. *Pattern Recognition*, *148*, Article 110204.

Yao, Man, Hu, Jiakui, Zhao, Guangshe, Wang, Yaoyuan, Zhang, Ziyang, Xu, Bo, et al. (2023). Inherent redundancy in spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 16924–16934).

Zeiler, Matthew D., & Fergus, Rob (2014). Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part i 13* (pp. 818–833). Springer.

Zhang, Aston, Lipton, Zachary C., Li, Mu, & Smola, Alexander J. (2023). *Dive into deep learning*. Cambridge University Press.

Zheng, Hao, Yang, Zhanlei, Liu, Wenju, Liang, Jizhong, & Li, Yanpeng (2015). Improving deep neural networks using softplus units. In *2015 international joint conference on neural networks* (pp. 1–4). IEEE.