Take A Shortcut Back: Mitigating the Gradient Vanishing for Training Spiking Neural Networks

Yufei Guo[*], **Yuanpei Chen**[*], **Zecheng Hao, Weihang Peng, Zhou Jie, Yuhan Zhang, Xiaode Liu, Zhe Ma**[†]
Intelligent Science & Technology Academy of CASIC, China
School of Computer Science, Peking University, China
yfguo@pku.edu.cn, rop477@163.com, haozecheng@pku.edu.cn, mazhe_thu@163.com

Abstract

The Spiking Neural Network (SNN) is a biologically inspired neural network infrastructure that has recently garnered significant attention. It utilizes binary spike activations to transmit information, thereby replacing multiplications with additions and resulting in high energy efficiency. However, training an SNN directly poses a challenge due to the undefined gradient of the firing spike process. Although prior works have employed various surrogate gradient training methods that use an alternative function to replace the firing process during back-propagation, these approaches ignore an intrinsic problem: gradient vanishing. To address this issue, we propose a shortcut back-propagation method in the paper, which advocates for transmitting the gradient directly from the loss to the shallow layers. This enables us to present the gradient to the shallow layers directly, thereby significantly mitigating the gradient vanishing problem. Additionally, this method does not introduce any burden during the inference phase. To strike a balance between final accuracy and ease of training, we also propose an evolutionary training framework and implement it by inducing a balance coefficient that dynamically changes with the training epoch, which further improves the network's performance. Extensive experiments conducted over popular datasets using several popular network structures reveal that our method consistently outperforms state-of-the-art methods.

1 Introduction

The Spiking Neural Network (SNN) has become a popular neural network due to its efficiency and has been widely used in various fields such as object recognition Li et al. (2021a); Xiao et al. (2021), object detection Kim et al. (2019); Qu et al. (2023), and pose tracking Zou et al. (2023). The SNN operates by using binary spike signals to transmit information. When the membrane potential exceeds the threshold, the spiking neuron fires a spike represented by 1; otherwise, there is no spike represented by 0. This unique information processing paradigm is energy-efficient since it replaces multiplications of weights and activations with simple additions. Additionally, this information processing paradigm can be implemented in an efficient event-driven-based computation manner on neuromorphic hardware Ma et al. (2017); Akopyan et al. (2015); Davies et al. (2018); Pei et al. (2019); Guo et al. (2023a), where the computational unit activates only when a spike occurs. This feature saves energy since the computational unit remains silent when there is no spike. Studies have shown that an SNN can save orders of magnitude more energy than its Artificial Neural Network (ANN) counterpart Akopyan et al. (2015); Davies et al. (2018).

[*] Equal Contributions.
[†] Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

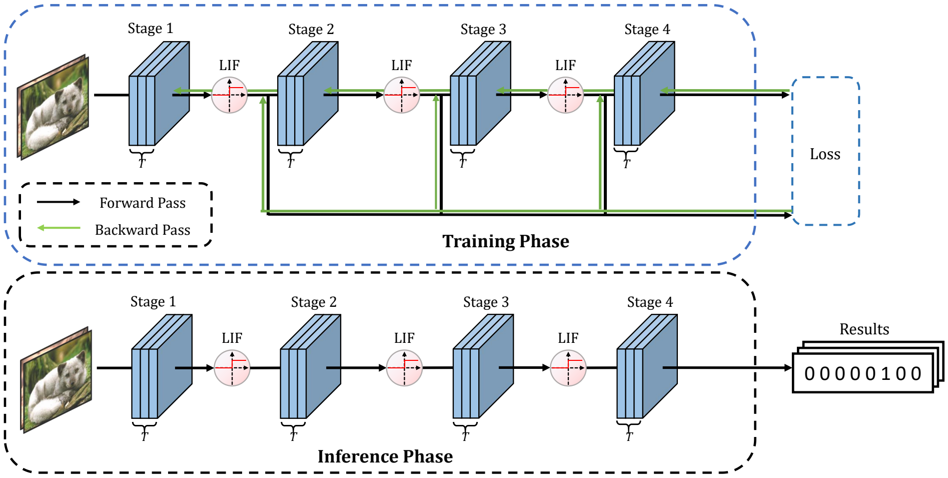


Figure 1: The overall workflow of the proposed method. We add multiple shortcut branches from the intermediate layers to the output thus the gradient from the output could be present to the shallow layers directly.

Although the SNN is energy-efficient, it is challenging to train it directly because the gradient of the firing spike process is not well-defined. This means that it is impossible to use gradient-based optimization methods to train an SNN directly. To overcome this problem, researchers have proposed various surrogate gradient training (SG) methods (Courbariaux et al., 2016; Esser et al., 2016; Bellec et al., 2018; Rathi & Roy, 2020; Wu et al., 2019; Neftci et al., 2019). These methods use an alternative function to replace the firing process during back-propagation. For example, in (Bohte, 2011), (Zenke & Ganguli, 2018), (Guo et al., 2022a), and (Cheng et al., 2020; Guo et al., 2024; Zhang et al., 2024), researchers used the truncated quadratic function, the `sigmoid` function, the `tanh`-like function, and the rectangular function as surrogates, respectively. However, SG methods have an intrinsic problem: gradient vanishing. All surrogate functions are bounded, and their gradients would be close to 0 in most intervals. As a result, the gradient of the SNN would quickly decrease from output to input, causing the weights of the shallow layers of the SNN to freeze during optimization. In subsection 4.1, we will theoretically and experimentally demonstrate the gradient vanishing problem.

To address this problem, we propose a shortcut back-propagation method in this paper, which involves transmitting the gradient from the loss to the shallow layers directly. To achieve this, we add multiple shortcut branches from intermediate layers to the output in the network. This allows information from the shallow layers to reach the output and final loss directly. Consequently, the gradient from the output can be present in the shallow layers, and their weights can be updated adequately, resulting in improved accuracy. Importantly, these shortcut branches can be removed without introducing any burden during the inference phase. Our proposed training framework is essentially a joint optimization problem on the weighted sum of the loss functions associated with these shortcut branches. However, if we give more weight to the main branch net, the earlier layer weights may not be updated sufficiently. Conversely, if we give more attention to the side shortcut branches, the accuracy cannot reach a high level since it directly relates to the main branch outputs rather than the side branch outputs. To balance this conflict, we introduce an evolutionary training framework. During early training, we pay more attention to the side branch net, allowing sufficient weight updates of the shallow layers. Towards the end of training, we increase the weight given to the main branch net, which further improves final accuracy. We accomplish this by inducing a dynamically changing balanced coefficient that adjusts with each training epoch. To illustrate our method's workflow, please refer to Figure 1. Our paper provides several key contributions, which can be summarized as follows:

- Firstly, we have identified that the gradient vanishing problem is a significant issue for SNNs. We have supported this conclusion with theoretical justifications and in-depth experimental analysis. To mitigate this problem, we have proposed the shortcut back-propagation approach, which is a simple yet effective method. Importantly, it will not introduce any additional burden during the inference phase.

- Secondly, we have proposed an evolutionary training framework that balances the weights of these branches with a gradual strategy. This approach ensures that earlier layer weights can be adequately updated while also improving overall accuracy.
- Lastly, we have evaluated the effectiveness and efficiency of our proposed methods on both static (CIFAR-10, CIFAR-100, ImageNet) and spiking (CIFAR10-DVS) datasets with widely used backbones. Our results demonstrate that the SNN trained with our proposed method is highly effective, achieving a top-1 accuracy of 77.79% on CIFAR-100 using ResNet19 with only 2 timesteps. This represents a significant improvement of about 3.3% compared with the current state-of-the-art SNN models even with more timesteps.

2 Related Work

2.1 Learning of Spiking Neural Networks

Unsupervised learning Diehl & Cook (2015); Hao et al. (2020), converting ANN to SNN (ANN2SNN) Sengupta et al. (2019); Hao et al. (2023a,b), and supervised learning Li et al. (2021b); Guo et al. (2022c) are three commonly used learning paradigms for SNNs. In unsupervised learning, the spike-timing-dependent plasticity (STDP) approach Lobov et al. (2020) is utilized to update the SNN model, which is considered more biologically plausible. However, due to the lack of explicit task guidance, this method is typically limited to small-scale networks and datasets. The ANN-SNN conversion method Han & Roy (2020); Li et al. (2021a); Bu et al. (2021); Ho & Chang (2021); Bu et al. (2022); Hao et al. (2023a); Lan et al. (2023) involves training an ANN first and then converting it into a homogeneous SNN by transferring the trained weights and replacing the ReLU neuron with a temporal spiking neuron. However, this method is not suitable for neuromorphic datasets as the ReLU neuron in the ANN cannot capture the rich temporal dynamics required for sequential information. Supervised learning Fang et al. (2021a); Wu et al. (2018), on the other hand, adopts an alternative function during back-propagation to replace the firing process, enabling direct training of the SNN as an ANN. This approach leverages the success of gradient-based optimization and can achieve good performance with only a few time steps, even on large-scale datasets. Moreover, supervised learning can handle temporal data effectively, making it an increasingly popular choice in SNN research. Our work also falls within this domain.

2.2 Relieving Training Difficulties for Supervised Learning of SNNs

As mentioned earlier, the surrogate gradient (SG) approach is commonly employed to address the non-differentiability of SNNs. Various SG functions have been utilized, including the truncated quadratic function (Bohte, 2011), the `sigmoid` function (Zenke & Ganguli, 2018), the `tanh`-like function (Guo et al., 2022a), and the rectangular function (Cheng et al., 2020). While the SG method is generally effective, it can also introduce certain issues. Firstly, there is a gradient mismatch between the true gradient and the surrogate gradient, resulting in slow convergence and reduced accuracy. To tackle this problem, IM-Loss (Guo et al., 2022a) proposed a dynamic manual SG method that adapts with each epoch, ensuring sufficient weight updates and accurate gradients simultaneously. In contrast to this manual design, the Differentiable Spike method (Li et al., 2021b) and the differentiable SG search method determine the optimal gradient estimation using finite difference and NAS techniques, respectively. Secondly, due to the firing function being bounded, all these SG functions are also bounded. As a result, the gradient approaches or reaches close to zero in most intervals, exacerbating the vanishing gradient problem in SNNs. To mitigate this issue, SEW-ResNet (Fang et al., 2021a) suggested using the ResNet with activation before addition form, while MS-ResNet (Hu et al., 2021) advocated for the ResNet with pre-activation form. Additionally, normalization techniques have been employed to address the vanishing/explosion gradient problems. For example, Threshold-dependent batch normalization (tdBN) (Zheng et al., 2021) normalized the data along both the channel and temporal dimensions. Other techniques such as Temporal Batch Normalization Through Time (BNTT) (Kim & Panda, 2021), postsynaptic potential normalization (PSP-BN) (Ikegawa et al., 2022), and temporal effective batch normalization (TEBN) (Duan et al., 2022) recognized the significant variation in spike distributions across different timesteps, and thus regulated spike flows by applying separate timestep batch normalization. MPBN (Guo et al., 2023b) introduced an additional batch normalization step after the membrane potential updating function to reestablish data flow. Similarly, regularization loss has been utilized to alleviate gradient explosion/vanishing problems. In RecDis-

SNN (Guo et al., 2022c), a membrane potential regularization loss was proposed to control spike flow within an appropriate range. In Spiking PointNet (Ren et al., 2023), a trained-less but learning-more paradigm was proposed. This method can be seen as using a small network in the training to mitigate the training difficulty problem.

However, all these methods still need to present the gradient from the output layer to the first layer step by step, thus the gradient vanishing problem cannot be solved completely. In this paper, we propose a shortcut back-propagation method. Different from the above methods, we present the gradient from the output layer to these shallow layers directly, thus the gradient vanishing problem can be solved totally.

3 Preliminary

The spiking neuron serves as the fundamental and specialized computing unit in SNNs, drawing inspiration from the human brain. In our paper, we employ the widely used spiking Leaky-Integrate-and-Fire (LIF) neuron model. This model accurately captures the behavior of biological neurons by considering the interaction between the membrane potential and input current. To show the spiking neuron in detail, we introduce the notation first. Throughout the paper, we denote the vectors in bold italic letters. For instance, we use the $\boldsymbol{x}$ and $\boldsymbol{o}$ to represent the input and target output variables. We denote the matrices or tensors by bold capital letters (e.g., $\mathbf{W}$ is for weights). We denote the constants by small upright or downright letters. For example, $\boldsymbol{u}_i^{(t)}$ means the i-th membrane potential at time step t. Then, the LIF neuron can be described as follows:

$$\boldsymbol{u}^{(t+1),\text{pre}} = \tau \boldsymbol{u}^{(t)} + \boldsymbol{c}^{(t+1)}, \text{where } \boldsymbol{c}^{(t+1)} = \mathbf{W}\boldsymbol{x}^{(t+1)}, \tag{1}$$

where τ is a constant within $(0,1)$, which controls the leakage of membrane potential. When τ is 1, the neuron will degenerate to the Integrate-and-Fire (IF) neuron model. In the paper, we set τ as 0.5. $\boldsymbol{u}^{(t+1),\text{pre}}$ is the pre-synaptic input at time step $t+1$, which is charged by the input current $\boldsymbol{c}^{(t+1)}$. Note that we omit the layer index for simplicity. The input current is computed by the dot-product between the weights, $\mathbf{W}$ of the current layer and the spike output, $\boldsymbol{x}^{(t+1)}$ from the previous layer. Once the membrane potential, $\boldsymbol{u}^{(t+1),\text{pre}}$ exceeds the firing threshold V_{th}, a spike will be fired from the LIF neuron, given by

$$\begin{aligned}\boldsymbol{o}^{(t+1)} &= \begin{cases}1 & \text{if } \boldsymbol{u}^{(t+1),\text{pre}} > V_{\text{th}} \\ 0 & \text{otherwise}\end{cases}, \\ \boldsymbol{u}^{(t+1)} &= \boldsymbol{u}^{(t+1),\text{pre}} \cdot (1 - \boldsymbol{o}^{(t+1)}).\end{aligned} \tag{2}$$

After firing, the spike output $\boldsymbol{o}^{(t+1)}$ will propagate to the next layer and become the input $\boldsymbol{x}^{(t+1)}$ of the next layer. In the paper, we set V_{th} as 1.

There is a notorious problem in SNN training the firing function is undifferentiable. To demonstrate this problem, we formulate the gradient by the chain rule, given as

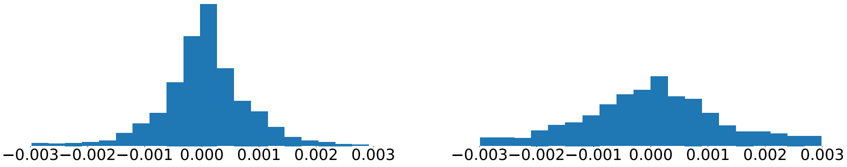
$$\frac{\partial L}{\partial \mathbf{W}} = \sum_t \left(\frac{\partial L}{\partial \boldsymbol{o}^{(t)}} \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{u}^{(t),\text{pre}}} + \frac{\partial L}{\partial \boldsymbol{u}^{(t+1),\text{pre}}} \frac{\partial \boldsymbol{u}^{(t+1),\text{pre}}}{\partial \boldsymbol{u}^{(t),\text{pre}}}\right) \frac{\partial \boldsymbol{u}^{(t),\text{pre}}}{\partial \mathbf{W}}. \tag{3}$$

Since the firing function Equation 2 is similar to the sign function. The $\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{u}^{(t),\text{pre}}}$ is 0 almost everywhere except for the threshold. Therefore, the updates for weights would either be 0 or infinity if we use the actual gradient of the firing function.

4 Methodology

4.1 The Gradient Vanishing Problem for SNNs

As aforementioned, the non-differentiability of SNNs poses challenges when training them directly. To address this issue, researchers have proposed the use of surrogate gradients. In this approach, the firing function remains unchanged during the forward pass, but a surrogate function is employed during the backward pass. The surrogate gradient is then computed based on this surrogate function.



(a)

(b)

Figure 2: The gradient distributions of the first layer for Spiking ResNet34 on CIFAR-100. (a) and (b) show the distributions for the vanilla model and the one with the shortcut back-propagation method.

There are three commonly used surrogate gradients:

$$\begin{cases} \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{u}^{(t),\text{pre}}} &= \gamma \max\left(0, 1 - \left|\frac{\boldsymbol{u}^{(t),\text{pre}}}{V_{\text{th}}} - 1\right|\right), \\ \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{u}^{(t),\text{pre}}} &= \frac{1}{a}\text{sign}\left(\left|\boldsymbol{u}^{(t),\text{pre}} - V_{\text{th}}\right| < \frac{a}{2}\right), \\ \frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{u}^{(t),\text{pre}}} &= k(1 - \tanh(\boldsymbol{u}^{(t),\text{pre}} - V_{\text{th}}))^2. \end{cases} \tag{4}$$

Each of these surrogate gradients includes a hyperparameter that controls the sharpness and width of the gradient. However, it is evident that these gradients, or their approximations, often become close to zero over a significant portion of their intervals. Consequently, this poses a considerable challenge in terms of severe gradient vanishing. While residual blocks have proven effective in mitigating gradient vanishing problems in traditional neural networks, their performance is not optimal when applied to SNNs. To demonstrate this, we express the skip connection using the following formulation:

$$\boldsymbol{o} = g(f(\boldsymbol{x}) + \boldsymbol{x}), \tag{5}$$

where $f(\cdot)$ is convolutional layers and $g(\cdot)$ is the activation function. The standard ResNet network is composed of multiple skip connection blocks cascaded together. In ANN, ReLU is used for $g(\cdot)$, since ReLU is unbounded for the positive part, the gradient can be passed to the input of the block directly. However, in the case of LIF neurons in SNNs, the gradient will be reduced through the surrogate gradient. Thus, the standard skip connections still suffer the gradient vanishing problem in SNNs. To visually illustrate this problem, we show the gradient distributions of the first layer for Spiking ResNet34 with 4 timesteps on the CIFAR-100 in the Figure 2(a). It can be seen that these gradients are close to 0 in most intervals, meaning the gradient vanishing problem is very significant for these shallow layers.

4.2 The Shortcut Back-propagation Method

Both theoretical analysis and experiments reveal the severity of the gradient vanishing problem in shallow layers of SNNs. In this paper, we propose a shortcut back-propagation method to address this issue. Specifically, the network is divided into several blocks, and we add multiple shortcut branches directly from these blocks to the output, as shown in Figure 1. These blocks are then followed by a global average pooling layer and a fully connected layer, resulting in the final output:

$$\boldsymbol{o}_{\text{final}} = \sum_{l} b_l(\boldsymbol{x}), \tag{6}$$

where the $b_l(\boldsymbol{x})$ represents the output of the l-th block. While the original final output can be expressed as

$$\boldsymbol{o}_{\text{final}} = b_n(\boldsymbol{x}), \ \text{ where } b_l(\boldsymbol{x}) = f_l(b_{l-1}(\boldsymbol{x})). \tag{7}$$

In the above equation, the n is the total number of the blocks and $f_l(\cdot)$ is the network of the l-th block. To demonstrate how our method alleviates the gradient vanishing problem, let's consider the gradient of the weight in the first layer as an example. For the original case, it can be expressed as

$$\frac{\partial L}{\partial \mathbf{W}_1} = \frac{\partial L}{\partial b_n(\boldsymbol{x})} \frac{\partial b_n(\boldsymbol{x})}{\partial b_{n-1}(\boldsymbol{x})} \cdots \frac{\partial b_{l+1}(\boldsymbol{x})}{\partial b_l(\boldsymbol{x})} \frac{\partial b_l(\boldsymbol{x})}{\partial \mathbf{W}_1}. \tag{8}$$

Algorithm 1 Training and inference procedure of SNN with our method.

```
Training
Input: An SNN to be trained; Initial balance coefficient $\lambda$; training dataset; total training iteration: $I_{\text{train}}$.
Output: The well-trained SNN.
for all $i = 1, 2, \ldots, I_{\text{train}}$ iteration do
    Get mini-batch training data, $\boldsymbol{x}_{\text{in}}(i)$ and class label, $\boldsymbol{y}(i)$;
    Feed the $\boldsymbol{x}_{\text{in}}(i)$ into the SNN and calculate every block output $b_l(\boldsymbol{x}_{\text{in}}(i))$ and the final main net output $b_n(\boldsymbol{x}_{\text{in}}(i))$;
    Update $\lambda$;
    Calculate the final output by Equation 10;
    Compute classification loss $L_{\text{CE}} = \mathcal{L}_{\text{CE}}(\boldsymbol{o}_{\text{final}}(i), \boldsymbol{y}(i))$;
    Calculate the gradient w.r.t. $\mathbf{W}$ by Equation 9;
    Update $\mathbf{W}$: $(\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial L}{\partial \mathbf{W}})$ where $\eta$ is learning rate.
end for
Inference
Input: The trained SNN; test dataset; total test iteration: $I_{\text{test}}$.
Output: The result.
for all $i = 1, 2, \ldots, I_{\text{test}}$ iteration do
    Get mini-batch test data, $\boldsymbol{x}_{\text{in}}(i)$ and class label, $\boldsymbol{y}(i)$ in test dataset;
    Feed the $\boldsymbol{x}_{\text{in}}(i)$ into the trained SNN;
    Calculate the final main net output $b_n(\boldsymbol{x}_{\text{in}}(i))$;
    Calculate the final output, $\boldsymbol{o}_{\text{final}}(i) = b_n(\boldsymbol{x}_{\text{in}}(i))$ ;
    Compare the final output $\boldsymbol{o}_{\text{final}}(i)$ and $\boldsymbol{y}(i)$ to compute the classification result.
end for
```

Since the $\frac{\partial b_{l+1}(\boldsymbol{x})}{\partial b_l(\boldsymbol{x})}$ is always reduced through the surrogate gradient, the $\frac{\partial L}{\partial \mathbf{W}_1}$ becomes very small, resulting in insufficient weight updates. However, for our method, it can be expressed as

$$\frac{\partial L}{\partial \mathbf{W}_1} = \sum_{l} \frac{\partial L}{\partial b_l(\boldsymbol{x})} \frac{\partial b_l(\boldsymbol{x})}{\partial \mathbf{W}_1}. \tag{9}$$

In our method, the gradient is directly fed into the first block and subsequently to $\mathbf{W}_1$. This completely solves the gradient vanishing problem. To further illustrate this advantage, we visualize the gradient distribution of the first layer for Spiking ResNet34 on CIFAR-100 in Figure 2(b). It can be observed that the distribution is relatively flat, indicating that the gradient vanishing problem has been effectively addressed in these shallow layers. Moreover, these shortcut branches can be removed during inference, incurring no additional cost.

4.3 The Evolutionary Training Framework

While the shortcut back-propagation method effectively addresses the gradient vanishing problem, it introduces a potential conflict. Each shortcut branch contributes to the final output, but if we focus too much on the shallow layer outputs, the overall accuracy may suffer. This is because the shortcut branches are removed during the inference phase, and the final accuracy is primarily influenced by the main branch. On the other hand, if we prioritize the main branch output, the final loss may not capture enough information from the shallow layers. Consequently, the weights of these shallow layers may not be updated adequately.

To address this issue, we propose an evolutionary training framework. During the early stages of training, we prioritize the former side branch net, allowing for sufficient weight updates in the shallow layers. As training progresses, we gradually shift our focus to the main branch net until all attention is on the main net at the end of training. To achieve this, we introduce a balance coefficient, denoted by $\lambda(i)$ and adopt a strategy of decreasing to adjust it as follows,

$$\boldsymbol{o}_{\text{final}} = b_n(\boldsymbol{x}) + \lambda(i) \sum_{l=1} b_l(\boldsymbol{x}), \text{ where } \lambda(i) = \lambda(1 - \frac{i}{I}). \tag{10}$$

Table 1: Ablation study for the shortcut back-propagation method.

Architecture	Method	Time-step	Accuracy
ResNet18	Vanilla Training	2	71.42%
	Shortcut Back-propagation	2	**73.68%**
	Evolutionary Training	2	**74.02%**
	Vanilla Training	4	72.22%
	Shortcut Back-propagation	4	**74.78%**
	Evolutionary Training	4	**74.83%**
ResNet34	Vanilla Training	2	69.82%
	Shortcut Back-propagation	2	**74.06%**
	Evolutionary Training	2	**74.17%**
	Vanilla Training	4	69.98%
	Shortcut Back-propagation	4	**75.67%**
	Evolutionary Training	4	**75.81%**

In the equation mentioned above, I represents the total number of training iterations, i denotes the current training iteration, and λ is a constant. In our work, we set λ to a value of 0.25. The training and inference of our SNN are detailed in Algo. 1.

5 Experiment

We conduct extensive experiments on CIFAR-10(100) Krizhevsky et al. (2010), ImageNet Deng et al. (2009), and CIFAR10-DVS Li et al. (2017) to demonstrate the superior performance of our method. The CIFAR-10(100) dataset comprises 50k training images and 10k test images, divided into 10(100) classes, each with 32×32 pixels. CIFAR10-DVS is a converted dataset derived from CIFAR-10. It consists of 10k images, with 1k images per class, in 10 classes. ImageNet is a significantly larger dataset, containing over 1,250k training images and 50k test images. For these static datasets (CIFAR-10, CIFAR-100, and ImageNet), we applied data normalization to ensure that they have 0 mean and 1 variance. Additionally, to prevent overfitting, we performed random horizontal flipping and cropping on all these datasets. For a fair comparison, AutoAugment Cubuk et al. (2018) was also used for data augmentation following these work Guo et al. (2022b); Li et al. (2021b) on CIFAR-10(100). Regarding the CIFAR10-DVS dataset, we partitioned it into 9k training images and 1k test images, as described in Wu et al. (2019). The training image frames were resized to 48×48 as in Zheng et al. (2021). Random horizontal flipping and random roll within a range of 5 pixels were also applied during training. For the test images, we simply resized them to 48×48 without any additional processing, following the approach of Li et al. Li et al. (2021b). We run the model with 8 V100.

5.1 Ablation Study

To validate the effectiveness of the proposed shortcut back-propagation method, we initially conducted several ablation experiments on the CIFAR-100 dataset using ResNet18 and ResNet34 with different timesteps. The results are detailed in Table 1. For ResNet18, the accuracy achieved through vanilla training is 71.42% and 72.22% under 2 and 4 timesteps, respectively, which aligns with existing works. Upon applying our shortcut back-propagation method, the accuracy improved to 73.68% and 74.78%, marking a notable 2.5% enhancement. Furthermore, with the evolutionary training method, the performance of ResNet18 saw an additional improvement, reaching 74.02% and 74.83%, respectively. Under vanilla training, ResNet34 achieved accuracies of 69.82% and 69.98% with 2 and 4 timesteps, respectively. These results are actually worse than those obtained with ResNet18. This suggests that the deeper model does not exhibit better performance due to the significant gradient vanishing problem in SNNs. However, by utilizing our shortcut back-propagation method, the accuracy significantly improves to 74.06% and 75.67%, representing a remarkable 5.0% enhancement. Notably, these results surpass the performance of ResNet18 as well. This clearly demonstrates the effectiveness of our proposed method. Furthermore, when incorporating the evolutionary training method, we observe further improvements in performance.

Table 2: Comparison with SoTA methods on CIFAR-10(100).

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR-10	TL Wu et al. (2021a)	Tandem Learning	CIFARNet	8	89.04%
	PTL Wu et al. (2021b)	Tandem Learning	VGG11	16	91.24%
	PLIF Fang et al. (2021b)	SNN training	PLIFNet	8	93.50%
	DSR Meng et al. (2022)	SNN training	ResNet18	20	95.40%
	KDSNN Xu et al. (2023)	SNN training	ResNet18	4	93.41%
	RecDis-SNN Guo et al. (2022c)	SNN training	ResNet19	2	93.64%
	Diet-SNN Rathi & Roy (2020)	SNN training	ResNet20	5	91.78%
				10	92.54%
	Dspike Li et al. (2021b)	SNN training	ResNet20	2	93.13%
				4	93.66%
	STBP-tdBN Zheng et al. (2021)	SNN training	ResNet19	2	92.34%
				4	92.92%
	TET Deng et al. (2022)	SNN training	ResNet19	2	94.16%
				4	94.44%
	Real Spike Guo et al. (2022d)	SNN training	ResNet19	2	95.31%
			ResNet20	4	91.89%
	Shortcut Back-propagation	SNN training	ResNet18	2	**93.89%**±0.11
				4	**94.30%**±0.09
			ResNet19	1	**94.47%**±0.09
				2	**95.19%**±0.10
	Evolutionary Training	SNN training	ResNet18	2	**93.92%**±0.08
				4	**94.46%**±0.11
			ResNet19	1	**94.81%**±0.13
				2	**95.36%**±0.10
CIFAR-100	T2FSNN Park et al. (2020)	ANN2SNN	VGG16	680	68.80%
	Real Spike Guo et al. (2022d)	SNN training	ResNet20	5	66.60%
	LTL Yang et al. (2022)	Tandem Learning	ResNet20	31	76.08%
	Diet-SNN Rathi & Roy (2020)	SNN training	ResNet20	5	64.07%
	RecDis-SNN Guo et al. (2022c)	SNN training	ResNet19	4	74.10%
	Dspike Li et al. (2021b)	SNN training	ResNet20	2	71.68%
				4	73.35%
	TET Deng et al. (2022)	SNN training	ResNet19	2	72.87%
				4	74.47%
	Shortcut Back-propagation	SNN training	ResNet18	2	**73.68%**±0.10
				4	**74.78%**±0.08
			ResNet19	1	**75.75%**±0.10
				2	**77.56%**±0.13
	Evolutionary Training	SNN training	ResNet18	2	**74.02%**±0.09
				4	**74.83%**±0.11
			ResNet19	1	**75.82%**±0.12
				2	**77.79%**±0.08

5.2 Comparison with SoTA Methods

In this section, we conducted a comparative experiment for the shortcut back-propagation method and the evolutionary training framework, taking into consideration several state-of-the-art works. To ensure a fair comparison, we present the top-1 accuracy results based on 3 independent trials. We first evaluated our method on CIFAR-10 and CIFAR-100 datasets. The AdamW optimizer was employed with a learning rate of 0.01 which is cosine decay to 0 and a weight decay of 0.02. Throughout the training process, all models were trained using a batch size of 128 for a total of 300 epochs. The results are summarized in Table 2. For the CIFAR-10 dataset, we chose SpikeNorm Sengupta et al. (2019), Hybrid-Train Rathi et al. (2020), TSSL-BP Zhang & Li (2020), TL Wu et al. (2021a), PTL Wu et al. (2021b), PLIF Fang et al. (2021b), DSR Meng et al. (2022), KDSNN Xu et al. (2023), Diet-SNN Rathi & Roy (2020), Dspike Li et al. (2021b), STBP-tdBN Zheng et al. (2021), TET Deng et al. (2022), RecDis-SNN Guo et al. (2022c), and Real Spike Guo et al. (2022d) as our comparison. Previous works utilizing ResNet18, ResNet19, and ResNet20 as backbones achieved the highest accuracies of 95.40%, 95.31%, and 93.66% with 20, 2, and 4 timesteps respectively. While our method based on ResNet18 and ResNet19 could reach 93.92% and 95.36% with 4 and 2 timesteps, respectively. Note that, ResNet18 is smaller than ResNet20. On the CIFAR-100 dataset, our method can also achieve better accuracy than other prior state-of-the-art works with fewer timesteps. For instance, our ResNet19 model with only 2 timesteps outperforms the current best method, TET and

Table 3: Comparison with SoTA methods on ImageNet.

Method	Type	Architecture	Timestep	Accuracy
TET Deng et al. (2022)	SNN training	ResNet34	6	64.79%
RecDis-SNN Guo et al. (2022c)	SNN training	ResNet34	6	67.33%
GLIF Yao et al. (2022)	SNN training	ResNet34	4	67.52%
DSR Meng et al. (2022)	SNN training	ResNet18	50	67.74%
MS-ResNet Hu et al. (2023)	SNN training	ResNet18	6	63.10%
MPBN Guo et al. (2023b)	SNN training	ResNet18	4	63.14%
		ResNet34	4	64.71%
Real Spike Guo et al. (2022d)	SNN training	ResNet18	4	63.68%
		ResNet34	4	67.69%
SEW ResNet Fang et al. (2021a)	SNN training	ResNet18	4	63.18%
		ResNet34	4	67.04%
Shortcut Back-propagation	SNN training	ResNet18	4	**64.47%**±0.21
		ResNet34	4	**67.90%**±0.17
Evolutionary Training	SNN training	ResNet18	4	**65.12%**±0.18
		ResNet34	4	**68.14%**±0.15

Table 4: Comparison with SoTA methods on CIFAR10-DVS.

Method	Type	Architecture	Timestep	Accuracy
STBP-tdBN Zheng et al. (2021)	SNN training	ResNet19	10	67.80%
RecDis-SNN Guo et al. (2022c)	SNN training	ResNet19	10	72.42%
Real Spike Guo et al. (2022d)	SNN training	ResNet19	10	72.85%
Dspike Li et al. (2021b)	SNN training	ResNet18	10	75.40%
Shortcut Back-propagation	SNN training	ResNet18	10	**82.00%**±0.10
Evolutionary Training	SNN training	ResNet18	10	**83.30%**±0.10

RecDis-SNN even with 4 timesteps by about 3.3%. These experimental results clearly demonstrate the efficiency and effectiveness of our method.

We proceeded to conduct experiments on the ImageNet dataset, which is a more complex dataset than CIFAR. The learning rate is adjusted to $4e^{-3}$ here. The comparative results are presented in Table 3. Notably, there have been several state-of-the-art (SoTA) baselines proposed for this dataset recently, such as RecDis-SNN Guo et al. (2022c), GLIF Yao et al. (2022), DSR Meng et al. (2022), MPBN Guo et al. (2023b), MS-ResNet Hu et al. (2023), Real Spike Guo et al. (2022d), and SEW ResNet Fang et al. (2021a). It is important to note that Real Spike and SEW ResNet deviate from the typical ResNet backbone as they generate integer outputs in the intermediate layers, making them more energy-intensive compared to methods with standard backbones. In contrast, our approach adopts the standard ResNet18 and ResNet34 architectures, yet it still outperforms Real Spike and SEW ResNet. Specifically, our method achieves an accuracy of 65.12% and 68.14% using ResNet18 and ResNet34, respectively, surpassing Real Spike by 1.44% and 0.45%, respectively. This improvement is noteworthy and demonstrates the effectiveness of our method in handling large-scale datasets.

In addition to the aforementioned experiments, we conducted tests on the highly popular neuromorphic dataset, CIFAR10-DVS. We use the same hyper-parameter setting as CIFAR. Employing ResNet18 as the foundational architecture, which is notably smaller compared to ResNet19, our approach achieved remarkable accuracies of 82.00% and 83.30%, respectively. These results demonstrate a substantial improvement over previous methodologies.

6 Conclusion

In the paper, we proved that the Spiking Neural Network suffers severe gradient vanishing with theoretical justifications and in-depth experimental analysis. To mitigate the problem, we proposed a shortcut back-propagation method. This enables us to present the gradient to the shallow layers directly, thereby significantly mitigating the gradient vanishing problem. Additionally, this method does not introduce any burden during the inference phase. we also presented an evolutionary training framework by inducing a balance coefficient that dynamically changes with the training epoch, which could further improve the accuracy. We conducted various experiments to verify the effectiveness of our method.

Acknowledgment

This work is supported by grants from the National Natural Science Foundation of China under contracts No.12202412 and No.12202413.

References

Akopyan, F., Sawada, J., Cassidy, A., Alvarez-Icaza, R., Arthur, J., Merolla, P., Imam, N., Nakamura, Y., Datta, P., Nam, G.-J., et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., and Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

Bohte, S. M. Error-backpropagation in networks of fractionally predictive spiking neurons. In *International Conference on Artificial Neural Networks*, pp. 60–68. Springer, 2011.

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.

Bu, T., Ding, J., Yu, Z., and Huang, T. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 11–20, 2022.

Cheng, X., Hao, Y., Xu, J., and Xu, B. Lisnn: Improving spiking neural networks with lateral interactions for robust object recognition. In *IJCAI*, pp. 1519–1525, 2020.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., Dimou, G., Joshi, P., Imam, N., Jain, S., et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Deng, S., Li, Y., Zhang, S., and Gu, S. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

Diehl, P. U. and Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience*, 9:99, 2015.

Duan, C., Ding, J., Chen, S., Yu, Z., and Huang, T. Temporal effective batch normalization in spiking neural networks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=fLIgyyQiJqz`.

Esser, S. K., Merolla, P. A., Arthur, J. V., Cassidy, A. S., Appuswamy, R., Andreopoulos, A., Berg, D. J., McKinstry, J. L., Melano, T., Barch, D. R., et al. From the cover: Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America*, 113(41):11441, 2016.

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems*, 34:21056–21069, 2021a.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2661–2671, 2021b.

Guo, Y., Chen, Y., Zhang, L., Liu, X., Wang, Y., Huang, X., and Ma, Z. IM-loss: Information maximization loss for spiking neural networks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022a. URL `https://openreview.net/forum?id=Jw34v_84m2b`.

Guo, Y., Chen, Y., Zhang, L., Wang, Y., Liu, X., Tong, X., Ou, Y., Huang, X., and Ma, Z. Reducing information loss for spiking neural networks. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. (eds.), *Computer Vision – ECCV 2022*, pp. 36–52, Cham, 2022b. Springer Nature Switzerland. ISBN 978-3-031-20083-0.

Guo, Y., Tong, X., Chen, Y., Zhang, L., Liu, X., Ma, Z., and Huang, X. Recdis-snn: Rectifying membrane potential distribution for directly training spiking neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 326–335, June 2022c.

Guo, Y., Zhang, L., Chen, Y., Tong, X., Liu, X., Wang, Y., Huang, X., and Ma, Z. Real spike: Learning real-valued spikes for spiking neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XII*, pp. 52–68. Springer, 2022d.

Guo, Y., Huang, X., and Ma, Z. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023a.

Guo, Y., Zhang, Y., Chen, Y., Peng, W., Liu, X., Zhang, L., Huang, X., and Ma, Z. Membrane potential batch normalization for spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19420–19430, October 2023b.

Guo, Y., Chen, Y., Liu, X., Peng, W., Zhang, Y., Huang, X., and Ma, Z. Ternary spike: Learning ternary spikes for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 12244–12252, 2024.

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.

Hao, Y., Huang, X., Dong, M., and Xu, B. A biologically plausible supervised learning method for spiking neural networks using the symmetric stdp rule. *Neural Networks*, 121:387–395, 2020.

Hao, Z., Bu, T., Ding, J., Huang, T., and Yu, Z. Reducing ann-snn conversion error through residual membrane potential. *arXiv preprint arXiv:2302.02091*, 2023a.

Hao, Z., Ding, J., Bu, T., Huang, T., and Yu, Z. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023b.

Ho, N.-D. and Chang, I.-J. Tcl: an ann-to-snn conversion with trainable clipping layers. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 793–798. IEEE, 2021.

Hu, Y., Wu, Y., Deng, L., and Li, G. Advancing residual learning towards powerful deep spiking neural networks. *arXiv preprint arXiv:2112.08954*, 2021.

Hu, Y., Deng, L., Wu, Y., Yao, M., and Li, G. Advancing spiking neural networks towards deep residual learning, 2023.

Ikegawa, S.-i., Saiin, R., Sawada, Y., and Natori, N. Rethinking the role of normalization and residual blocks for spiking neural networks. *Sensors*, 22(8), 2022. ISSN 1424-8220. doi: 10.3390/s22082876. URL `https://www.mdpi.com/1424-8220/22/8/2876`.

Kim, S., Park, S., Na, B., and Yoon, S. Spiking-yolo: Spiking neural network for energy-efficient object detection, 2019.

Kim, Y. and Panda, P. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience*, pp. 1638, 2021.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*, 5(4):1, 2010.

Lan, Y., Zhang, Y., Ma, X., Qu, Y., and Fu, Y. Efficient converted spiking neural network for 3d and 2d classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9211–9220, 2023.

Li, H., Liu, H., Ji, X., Li, G., and Shi, L. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

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, 2021a.

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021b.

Lobov, S. A., Mikhaylov, A. N., and Kazantsev, V. B. Spatial properties of stdp in a self-learning spiking neural network enable controlling a mobile robot. *Frontiers in Neuroscience*, 14:–, 2020.

Ma, D., Shen, J., Gu, Z., Zhang, M., Zhu, X., Xu, X., Xu, Q., Shen, Y., and Pan, G. Darwin: A neuromorphic hardware co-processor based on spiking neural networks. *Journal of Systems Architecture*, 77:43–51, 2017.

Meng, Q., Xiao, M., Yan, S., Wang, Y., Lin, Z., and Luo, Z.-Q. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12444–12453, 2022.

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.

Park, S., Kim, S., Na, B., and Yoon, S. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Pei, J., Deng, L., Song, S., Zhao, M., Zhang, Y., Wu, S., Wang, G., Zou, Z., Wu, Z., He, W., et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767): 106–111, 2019.

Qu, J., Gao, Z., Zhang, T., Lu, Y., Tang, H., and Qiao, H. Spiking neural network for ultra-low-latency and high-accurate object detection, 2023.

Rathi, N. and Roy, K. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*, 2020.

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.

Ren, D., Ma, Z., Chen, Y., Peng, W., Liu, X., Zhang, Y., and Guo, Y. Spiking pointnet: Spiking neural networks for point clouds. *arXiv preprint arXiv:2310.06232*, 2023.

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, J., Chua, Y., Zhang, M., Li, G., Li, H., and Tan, K. C. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021a.

Wu, J., Xu, C., Han, X., Zhou, D., Zhang, M., Li, H., and Tan, K. C. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7824–7840, 2021b.

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.

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., and Shi, L. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1311–1318, 2019.

Xiao, M., Meng, Q., Zhang, Z., Wang, Y., and Lin, Z. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in Neural Information Processing Systems*, 34:14516–14528, 2021.

Xu, Q., Li, Y., Shen, J., Liu, J. K., Tang, H., and Pan, G. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. *arXiv preprint arXiv:2304.05627*, 2023.

Yang, Q., Wu, J., Zhang, M., Chua, Y., Wang, X., and Li, H. Training spiking neural networks with local tandem learning. *arXiv preprint arXiv:2210.04532*, 2022.

Yao, X., Li, F., Mo, Z., and Cheng, J. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *arXiv preprint arXiv:2210.13768*, 2022.

Zenke, F. and Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

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.

Zhang, Y., Liu, X., Chen, Y., Peng, W., Guo, Y., Huang, X., and Ma, Z. Enhancing representation of spiking neural networks via similarity-sensitive contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16926–16934, 2024.

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.

Zou, S., Mu, Y., Zuo, X., Wang, S., and Cheng, L. Event-based human pose tracking by spiking spatiotemporal transformer, 2023.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes] .

   Justification: We clearly state the claims made and the contributions made in both the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [NA] .

   Justification: We find no limitation which we feel must be specifically highlighted here.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes] .

Justification: We provide the full set of assumptions and complete proofs in the Section 4.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes] .

   Justification: We provide the detail experiment settings in the Section 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes] .

Justification: We provide open access to the data and code with sufficient instructions in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] .

   Justification: All implementations are described in the experiments section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes] .

   Justification: We report the mean as well as the standard deviation accuracy in experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes] .

   Justification: The computation resources description is provided in the experiment section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes] .

   Justification: The research conducted with the NeurIPS Code of Ethics

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No] .

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA] .

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes] .

    Justification: The original paper for datasets we used are all cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA] .

    Justification: We adopt public datasets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA] .

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.