Efficient Surrogate Gradients for Training Spiking Neural Networks

Hao Lin [1 *] **Shikuang Deng** [1 2 *] **Shi Gu** [1 2]

Abstract

To guide the shape optimization in applying surrogate gradients for training SNN, we propose an indicator k, which represents the proportion of membrane potential with non-zero gradients in backpropagation. Further we present a novel k-based training pipeline that adaptively makes trade-offs between the surrogate gradients' shapes and its effective domain, followed by a series of ablation experiments for verification. Our algorithm achieves 68.93% accuracy on the ImageNet dataset using SEW-ResNet34. Moreover, our method only requires extremely low external cost and can be simply integrated into the existing training procedure.

1. Introduction

Spiking Neural Networks (SNN) have gained increasing attention in recent years due to their biological rationale and potential energy efficiency as compared to the common real-value based Artificial Neural Networks (ANN). SNN communicates across layers by spiking signals. On the one hand, this spiking mechanism turns multiplicative operations to additive operations, increasing the inference procedure's efficiency. On the other hand, it introduces an intrinsic issue of differentiability, which makes training SNNs more challenging. At present, the method for obtaining practical SNNs can be roughly divided into three categories: converting a pretrained ANN to SNN (Sengupta et al., 2019; Deng & Gu, 2020; Li et al., 2021a; Bu et al., 2021), training with biological heuristics methods (Hao et al., 2020; Shrestha et al., 2017; Lee et al., 2018), and training with BP-like methods (Wu et al., 2018; Zheng et al., 2020; Li et al., 2021b; Yang et al., 2021).

The purpose of this work is to optimize the SNN training pipeline by adaptively altering the shape of surrogate gradient in order to control the effective domain for the surrogate gradients. We introduce an index k to denote the proportion of membrane potential with non-zero gradients in backpropagation and present a technique to control the proportion of non-zero gradients (CPNG) in the network. The CPNG technique modifies the shape of surrogate gradients during network training, progressively approaching the δ-function while maintaining the index k steady within an effective range to ensure training stability. Finally, each layer succeeds in finding a surrogate gradient shape that makes a better balance between the approximation error to the δ-function with the size of effective domain than the fixed-shape surrogate gradients. It's worth mentioning that our strategy only incurs minor additional costs during the training phase and has no effect on the inference phase. We verify the compatibility of CPNG to the existing mainstream SNN infrastructures such as VGG (Simonyan & Zisserman, 2014), ResNet (He et al., 2016), and Sew-ResNet (Fang et al., 2021). In all reported comparative experiments, training with CPNG gives more accurate models than training with vanilla surrogate gradients. Our main contributions can be summarized as follows:

- We identify and investigate the impact of the shape of surrogate gradients on SNN training. Our finding characterizes a special representative power for SNN that can be utilized to improve its performance.
- We propose a statistical indicator k for the domain efficiency of surrogate gradients and a k-based training method CPNG that adjusts the shape of surrogate gradients through the training process, driving the surrogate gradients close to the theoretical δ-function with ensured trainability on sufficiently large domains.
- Our CPNG method improves classification accuracy on both static image datasets including CIFAR10, CIFAR100 and ImageNet, as well as event-based image datasets such as CIFAR10-DVS. We achieve an accuracy of 68.93% in the experiment that trains ImageNet on Sew-ResNet34.

2. Preliminary

Through out the paper, we use bold letters to denote matrices and vectors, superscripts to identify specific layers,

*Equal contribution [1]University of Electronic Science and Technology of China [2] Shenzhen Institute for Advanced Study, UESTC. Correspondence to: Shi Gu <gus@uestc.edu.cn>.

Published at the Differentiable Almost Everything Workshop of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. July 2023.

subscripts to denote specific neurons, and indexes to identify specific moments.

2.1. Leaky Integrate-and-Fire Model

We use the Leaky Integrate-and-Fire (LIF) module for spiking neurons. Formally, given the pre-synaptic input (denoted by $c_i^{(l)}[t+1]$) of the i^{th} neuron in the l^{th} layer at time step $t+1$, we can model the iterative process in LIF as

$$c_i^{(l)}[t+1] = \sum_j^{N^{(l-1)}} w_{ij}^{(l)} s_j^{(l-1)}[t+1], \tag{1}$$

$$u_i^{(l)}[t+1] = \tau u_i^{(l)}[t](1 - s_i^{(l)}[t]) + c_i^{(l)}[t+1], \tag{2}$$

$$s_i^{(l)}[t+1] = H(u_i^{(l)}[t+1] - V_{th}). \tag{3}$$

Here, $N^{(l-1)}$ is the number of neurons in the $(l-1)^{th}$ layer, $s_j^{(l-1)}[t+1]$ is the output spike of the j^{th} neuron in the $(l-1)^{th}$ layer at time $t+1$, $w_{ij}^{(l)}$ is the weight between j^{th} neuron in $(l-1)^{th}$ layer and i^{th} neuron in l^{th} layer, $u_i^{(l)}[t]$ is the membrane potential of the i^{th} neuron in the l^{th} layer at time t, τ is the membrane potential attenuation factor, $H(\cdot)$ is the step function, and V_{th} is the activation threshold. When the membrane potential of a neuron exceeds the activation threshold, a spike is released and the membrane potential of the current neuron is set to zero.

2.2. Surrogate Gradient Function

There are various surrogate gradient shapes adopted by previous work (Wu et al., 2018; Neftci et al., 2019). In this work, we used triangle-like function, rectangular-like function and arctan-like function to verify the effectiveness of CPNG. These functions are described below:

$$\phi_{\text{triangle}}(x) = \begin{cases} \beta(1-\beta|x|) & \text{if } |x| < 1/\beta \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

$$\phi_{\text{rectangular}}(x) = \begin{cases} \beta & \text{if } |x| < 1/(2\beta) \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

$$\phi_{\text{arctan}}(x) = \frac{\beta}{1 + (\pi\beta x)^2}, \tag{6}$$

where β represents the maximum gradient value of current surrogate gradient function. Notably, the surrogate gradient satisfy $\int_{-\infty}^{+\infty} f(x) = 1$, which is also the property of the δ-function.

3. Method

3.1. Shape Parameters and Effective Domain Indicator

Shape Parameters. Intuitively, increasing the shape parameter β of surrogate gradients would drive it closer to the δ-function. One might expect to adopt a very high β to obtain SNN for a good performance. We first examine whether this intuitive approach is possible. We trained VGG16-structured SNN (replace ReLU with LIF, and use average pooling) on CIFAR100 using triangle-like surrogate gradient with β set from 0.25 to 2 respectively. As shown in Fig. 1, the test accuracy increases when β varies from 0.25 to 1.0 but remains at 1.0% when β is set to 1.5 and 2.0, indicating that properly increasing β may benefit the training but arbitrarily increasing β will drive the training collapse.

In fact, these results unveil that efficient training of SNNs requires not only the approximation to the δ-function but also the insurance for the surrogate gradients to work. Thus

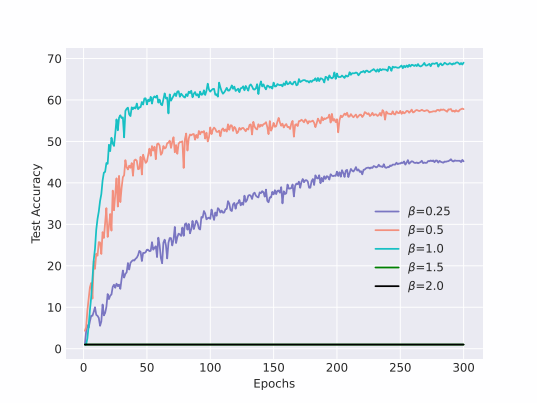

Figure 1. Test accuracy of different β when threshold is 1.0.

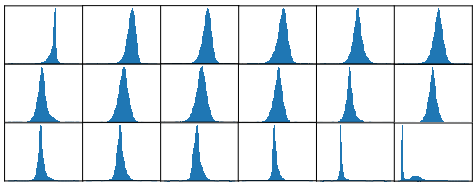

Figure 2. Membrane distribution of each layer in experiment training ResNet19 on CIFAR10.

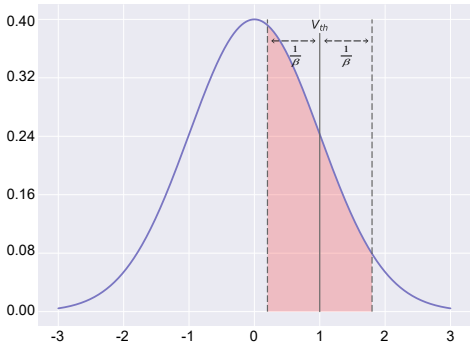

Figure 3. The proportion of neurons with non-zero gradient under a surrogate gradient with a certain β.

it is necessary to employ a dynamic shape-changing strategy rather than using a fixed-shape surrogate gradient. This issue is also covered by (Li et al., 2021b) as well and was owed to the lack of adaption to the dataset variation.

Effective Domain Indicator.

To quantitatively guide the choice of β, we propose a statistical indicator to denote the percentage of membrane potentials that fall into the domain of surrogate gradients. As illustrated in Fig. 2, the distribution of membrane potentials on each layer takes the normal shape. Thus, for the simplicity of calculation, we regard the membrane potential distribution of all neurons within the same layer as a Gaussian one. By calculating the mean μ and the standard deviation σ of the membrane potential before this layer releases spikes, we can obtain the proportion of the neuron with a non-zero gradient during a certain iteration (the area between the red lines in Fig. 3). For a given β, the effective gradient domain of triangle-like surrogate gradient is $[V_{th} - 1/\beta, V_{th} + 1/\beta]$, we can obtain the definite integral of the current normal distribution in this effective gradient domain, which is the *Effective Domain Indicator* k:

$$k = \int_{V_{th}-1/\beta}^{V_{th}+1/\beta} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx. \tag{7}$$

For each layer, we record the membrane potential of all neurons in every time step (a tensor shaped like batchsize-by-timestep-by-channels-by-H-by-W) and calculate the mean and variance. Based on this indicator k, we can then effectively determine to what extent we can tune the β while ensuring that there are enough membrane potentials located within the effective range of surrogate gradients to make the training progress.

3.2. CPNG Method

In this section, we will cover how to combiningly optimize β and k to maximize the effectiveness of surrogate gradients. To train the network successfully, there must be sufficient membrane potential values in the effective domain of the surrogate gradient, i.e., k must be large enough. The most extreme case is $\beta \to 0$, which gives $k \to 1$. Obviously, this is not an optimal solution as it introduces substantial error for the gradients. If we reasonably restrict the effective interval of the surrogate gradients, it is possible to drive the SNN to select those more essential membrane potentials for backpropagation.

We also need to ensure that the new k does not make the network difficult to train, for this, CPNG sets the target k of each layer to the smallest k that has occurred in the current layer during the training iteration, rather than an artificial goal. If the network can be trained when the smallest k appears, then the network should still be trained after we adjust the k and maintain the smallest k. When using CPNG, we expect the network parameters to have been reasonably updated, that is, the network has traversed the whole dataset and has minimal training loss at the current moment. This can reduce the misleading of data randomness to CPNG. For example, if we use CPNG once per batch, the network parameters are mostly affected by the first few batches in the early stages of network training, and the statistical indicator k obtained by using such network parameters will have a lot of randomnesses.

Algorithm 1 Control the Proportion of Non-Zero Gradient

Input: SNN model with L layer, current iterator epoch e, k_{limit}, and vector k_{recorder}: store each layer's smallest k
Output: Each layer's surrogate gradient parameter β
if e == 0 **then**
  for $l = 1, 2, \cdots L$ **do**
    calculate current k by Eqn.7 for layer-l and store at $k_{\text{recorder}}[l]$
  end for
else
  for $l = 1, 2, \cdots L$ **do**
    calculate current k_{cur} by Eqn.7 for layer-l
    if $k_{\text{recorder}}[l] < k_{\text{limit}}$ **then**
      $k_{\text{recorder}}[l] = k_{\text{limit}}$
    else if $k_{\text{cur}} < k_{\text{recorder}}[l]$ **then**
      $k_{\text{recorder}}[l] = k_{\text{cur}}$
    end if
    $k_{\text{min}} = k_{\text{recorder}}[l]$
    if $k_{\text{min}} \neq k_{\text{cur}}$ **then**
      use k_{min} to update β using binary search method.
    end if
  end for
end if
return β for each layer

CPNG computes the smallest k of each layer during the iteration process as k_{min} and records it. If the k value of a certain layer rises after an epoch, CPNG adjusts the k value of the current layer to k_{min} by increasing the β, otherwise, keep the current β fixed and update k_{min}. Since the k_{min} of each layer of neurons may be different, different layers may have different surrogate gradient shapes. In addition, we set a safe lower bound k_{limit}. When k falls below the lower bound, CPNG will pause adjusting the current neuron until the k of the current neuron exceeds the lower bound. The CPNG algorithm is detailed in Algo.1.

3.3. The Cost of CPNG Method

The extra cost of CPNG occurs in two steps: (1) collecting the mean and variance of the membrane potential before releasing the spikes of each layer; (2) altering the β using the indicator k. In our experiment, we only use the mean and variance of one batch to calculate the indicator k, which

makes the cost of the first step in the same order of magnitude as the batch normalization (Ioffe & Szegedy, 2015) operation. For the latter step, we provide a binary search method that solves the problem very fast, and further optimization algorithms can further improve the solution speed. Quantitatively, in the VGG16+CIFAR100 experiment, it takes an average of 1.7GFLOPs to obtain the output corresponding to an input without CPNG, and the first step of CPNG will only add 5.59×10^{-3} additional GFLOPs. Using CPNG once per epoch takes an average of 3.3 seconds of overhead (1.57% of total training time).

4. Experiment

To verify the effectiveness of the CPNG method, we provide groups of comparative experiments (Sec. 4.2) and compare CPNG with existing works in Sec. 4.3. More experimental results are presented in the appendix.

4.1. Implementation Details

All the SNN architectures include the tdBN layer (Zheng et al., 2020) with the average-pooling layer, and compared to their ANN versions, we replace the activation function ReLU with LIF. Our experiment settings, such as optimizer, learning rate, are detailed in Appendix. A. Except for applying the CPNG method at the end of each epoch, all other conditions, such as learning rate, batch size, etc., are consistent. We used TIT (Deng et al., 2022) when training CIFAR, which reduces the training time by initializing the SNN with a smaller simulation length.

Table 1. k_{limit} Experiments on ResNet19+CIFAR100.

k_{limit}	—	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Accuracy	74.05%	74.66%	74.66%	74.66%	73.96%	74.24%	74.02%	73.89%

4.2. Performance improvement with CPNG

First we explored the effect of different k_{limit} on CPNG, Tab. 1 shows that we may don't need to focus too much on k_{limit}. When k_{limit} is 0.1-0.3, it can bring about a 0.61% improvement. In our experiments, we set k_{limit} to 0.2 uniformly.

Additionally, we tested the applicability of CPNG to various surrogate gradient functions. For surrogate gradients with non-zero gradient everywhere, such as arctan, we directly use $[V_{th} - 1/\beta, V_{th} + 1/\beta]$ as the integration interval to calculate the indicator k, then use Algo. 1 to solve new β.

Finally, we verify the compatibility of CPNG with existing direct training methods TET (Deng et al., 2022)(Tab. 2). In all experiments, CPNG can achieve better results. It is worth mentioning that the improvement brought by CPNG is not due to bad counterpart. For example, in the experiment of using TET to train CIFAR100, we can achieve **75.36%** accuracy without using CPNG in our experimental setting.

Table 2. Examine CPNG under various conditions.

Dataset	Method	Architecture	Time Step	Accuracy
CIFAR100	Triangular	ResNet19	2	73.71±0.005%
	Triangular+CPNG	ResNet19	2	74.38±0.005%
	Rectangular	ResNet19	2	72.35±0.002%
	Rectangular+CPNG	ResNet19	2	73.63±0.026%
	ArcTan	ResNet19	2	71.46±0.080%
	ArcTan+CPNG	ResNet19	2	72.43±0.035%
	Triangular+TET	ResNet19	2	75.36±0.049%
	Triangular+TET+CPNG	ResNet19	2	75.90±0.001%

Table 3. Result of training spiking neural network.

Dataset	Method	Architecture	Time Step	Accuracy
CIFAR10	STBP-tdBN (Zheng et al., 2020)	ResNet19	6	93.16%
	ANN-to-SNN (Li et al., 2021a)	ResNet20	32 64 128	94.78% 95.30% 95.42%
	ANN-to-SNN (Bu et al., 2021)	ResNet20	8 16 32	89.55% 91.62% 92.24%
	Dspike (Li et al., 2021b)	ResNet18	2 4 6	93.13% 93.66% 94.25%
	TET (Deng et al., 2022)	ResNet19	2 4 6	94.16% 94.44% 94.50%
	CPNG	ResNet19	2	**93.79**±0.002%
	CPNG	ResNet19	4	**94.14**±0.009%
	CPNG	ResNet19	6	**94.10**±0.005%
CIFAR100	Diet-SNN (Rathi & Roy, 2020)	VGG16	5	69.67%
	ANN-to-SNN (Li et al., 2021a)	VGG16	32 64 128	73.55% 76.64% 77.40%
	ANN-to-SNN (Bu et al., 2021)	VGG16	8 16 32	73.96% 76.24% 77.01%
	Dspike (Li et al., 2021b)	ResNet18	2 4 6	71.68% 73.35% 74.24%
	TET (Deng et al., 2022)	ResNet19	2 4 6	72.87% 74.47% 74.72%
	CPNG	VGG16	5	**71.32**±0.20%
	CPNG	ResNet19	2	**74.40**±0.005%
	CPNG	ResNet19	4	**75.29**±0.001%
	CPNG	ResNet19	6	**75.37**±0.056%
CIFAR10-DVS	Dspike (Li et al., 2021b)	ResNet18	10	75.40%
	TET(Deng et al., 2022)	VGGSNN	10	83.17%
	CPNG	ResNet18	10	**76.5**±0.007%
	CPNG + TET	VGGSNN	10	**83.27**±0.162%
ImageNet	ANN-to-SNN (Li et al., 2021a)	ResNet34	32 64 128	64.54% 71.12% 73.45%
	ANN-to-SNN (Bu et al., 2021)	ResNet34	16 32 64	59.35% 69.37% 72.35%
	Sew-ResNet (Fang et al., 2021)	Sew-ResNet34	4	67.04%
	TET (Deng et al., 2022)	Spiking-ResNet34	6	64.79%
	TET (Deng et al., 2022)	Sew-ResNet34	4	68.00%
	CPNG	Sew-ResNet34	4	**68.93%**

4.3. Comparison to Existing Works

In this section, the experimental results we report all use triangle-like surrogate gradient. On some datasets, the current SOTA conversion method performs better than direct training, but they need lengthy simulation time steps, especially on the ImageNet dataset. Our main purpose is not to chase SOTA, but to demonstrate the effectiveness of CPNG itself and the compatibility of CPNG with other methods. The results of the experiment are shown in Tab. 3.

5. Conclusion

This work proposes a new perspective for directing the shape change of the surrogate gradient, we propose a statistical indicator k that guides the shape change of the surrogate gradient, and propose the CPNG method for modifying the shape of the surrogate gradient during training while guaranteeing the proportion of membrane potential with non-zero gradients. It's possible that the failure to produce satisfactory results when pulling surrogate gradient directly to δ-function is due to a failure to meet the premise that the network can be trained normally. In other words, there may exists a trade-off between the approximation to the δ-function and the effective domain of gradients under the given dataset, and CPNG helps us approach the equilibrium point.

References

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.

Deng, S. and Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2020.

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

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. Deep residual learning in spiking neural networks. *arXiv preprint arXiv:2102.04159*, 2021.

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.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Lee, C., Panda, P., Srinivasan, G., and Roy, K. Training deep spiking convolutional neural networks with stdp-based unsupervised pre-training followed by supervised fine-tuning. *Frontiers in neuroscience*, 12:435, 2018.

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. *arXiv preprint arXiv:2106.06984*, 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, 2021b.

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

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

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.

Shrestha, A., Ahmed, K., Wang, Y., and Qiu, Q. Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning. In *2017 international joint conference on neural networks (IJCNN)*, pp. 1999–2006. IEEE, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

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

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. Going deeper with directly-trained larger spiking neural networks. *arXiv preprint arXiv:2011.05280*, 2020.

A. More Experiment Details

In this section, we provide more experimental details. We use cosine decay to gradually reduce the learning rate to 0 and the initial β for all experiments is 1.0. In the last batch (a fixed batch randomly selected at the beginning), the mean and variance of the membrane potential are additionally stored in the forward process, then the stored mean and variance are used to obtain a new β according to Algo. 1. Hyperparameters are shown in the table 4.

Table 4. Experiment Setting

Experiment	CIFAR100	CIFAR10	CIFAR-DVS (ResNet18)	CIFAR-DVS (VGG11)	ImageNet
learning rate	0.1	0.1	0.01	0.001	0.03
weight decay	1e-4	1e-4	4e-5	4e-5	4e-5
momentum	0.9	0.9	0.9	—-	0.9
optimizer	sgd	sgd	sgd	adam	sgd
warm-up	False	False	False	False	False
batch size	256	256	72	72	256

B. Results of Comparative Experiment

Only the results of CPNG on the CIFAR100 dataset are shown in Sec. 4.2, while the results on more datasets are presented in Tab. 5. For all comparative experiments, using CPNG can yield better results than not using CPNG.

C. Surrogate Gradient Shapes of Different Layers

Fig. 4 describes the change of k during the training process. CPNG does control the rise of k during the training process and makes it stabilize at the end. Fig. 4 also shows that we don't need to pay too much attention to k_{limit}. k_{limit} only provides a guarantee in extreme cases, but extreme cases do not necessarily appear during the network training process.

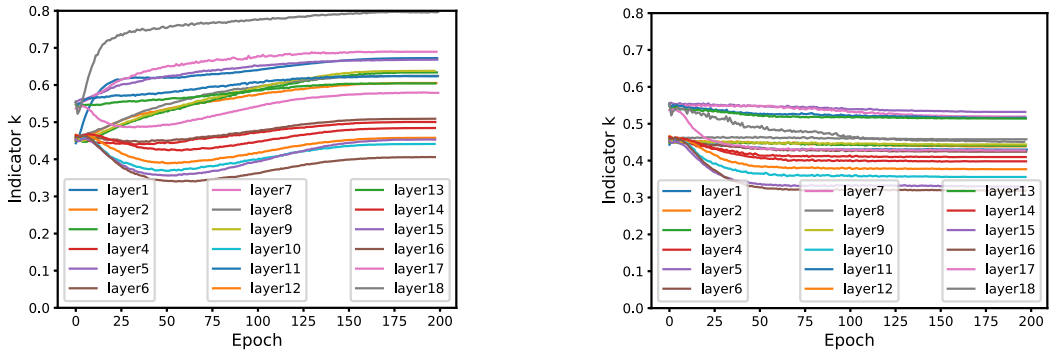

Figure 4. In the experiment of ResNet19+CIFAR100, the proportion of non-zero gradient membrane potentials of neurons in different layers without CPNG (a) and with CPNG (b).

We show the final β of some experiments in Fig. 5. Various layers' β are different, which demonstrates that various layers match distinct surrogate gradient shapes as a result of their varying membrane potential distributions. CPNG eventually increases the β of most layers. Compare to CPNG, randomly increasing the β can make the network difficult to train (Fig. 1). Even with β set to 1.5, which most of the neuron layers shown in Fig. 5a can approach or reach, the network is still difficult to train. This demonstrates that it is safe to increase β using CPNG, while it is unsafe to increase β arbitrarily.

Table 5. Examine CPNG on various surrogate gradients.

Dataset	Method	Architecture	Time Step	Accuracy
CIFAR10	Triangular	ResNet19	2	93.73±0.015%
	Triangular+CPNG	ResNet19	2	**93.77**±0.003**%**
	Triangular	ResNet19	4	94.07±0.002%
	Triangular+CPNG	ResNet19	4	**94.14**±0.009**%**
	Triangular	ResNet19	6	93.99±0.002%
	Triangular+CPNG	ResNet19	6	**94.10**±0.005**%**
	Rectangular	ResNet19	2	92.51±0.038%
	Rectangular+CPNG	ResNet19	2	**92.92**±0.014**%**
	ArcTan	ResNet19	2	92.12±0.016%
	ArcTan+CPNG	ResNet19	2	**92.84**±0.024**%**
	Triangular+TET	ResNet19	2	93.76±0.006%
	Triangular+TET+CPNG	ResNet19	2	**93.81**±0.006**%**
CIFAR100	Triangular	ResNet19	2	73.71±0.005%
	Triangular+CPNG	ResNet19	2	**74.38**±0.005**%**
	Triangular	ResNet19	4	75.06±0.005%
	Triangular+CPNG	ResNet19	4	**75.37**±0.010**%**
	Triangular	ResNet19	6	75.00±0.016%
	Triangular+CPNG	ResNet19	6	**75.40**±0.056**%**
	Rectangular	ResNet19	2	72.35±0.002%
	Rectangular+CPNG	ResNet19	2	**73.63**±0.026**%**
	ArcTan	ResNet19	2	71.46±0.080%
	ArcTan+CPNG	ResNet19	2	**72.43**±0.035**%**
	Triangular+TET	ResNet19	2	75.36±0.049%
	Triangular+TET+CPNG	ResNet19	2	**75.90**±0.001**%**
	Triangular	VGG16	2	69.13±0.110%
	Triangular+CPNG	VGG16	2	**71.14**±0.098**%**
CIFAR10-DVS	Triangular+TET	VGG11	10	82.93±0.116%
	Triangular+TET+CPNG	VGG11	10	**83.27**±0.162**%**
	Triangular	ResNet18	10	75.8±1.787%
	Triangular+CPNG	ResNet18	10	**76.5**±0.007 **%**
	Rectangular	ResNet18	10	74.3±0.053%
	Rectangular+CPNG	ResNet18	10	**75.0**±0.772 **%**
	ArcTan	ResNet18	10	70.8±0.345%
	ArcTan+CPNG	ResNet18	10	**70.87**±0.002 **%**
	Triangular+TET	ResNet18	10	80.37±0.616%
	Triangular+TET+CPNG	ResNet18	10	**80.43**±0.309‘ **%**

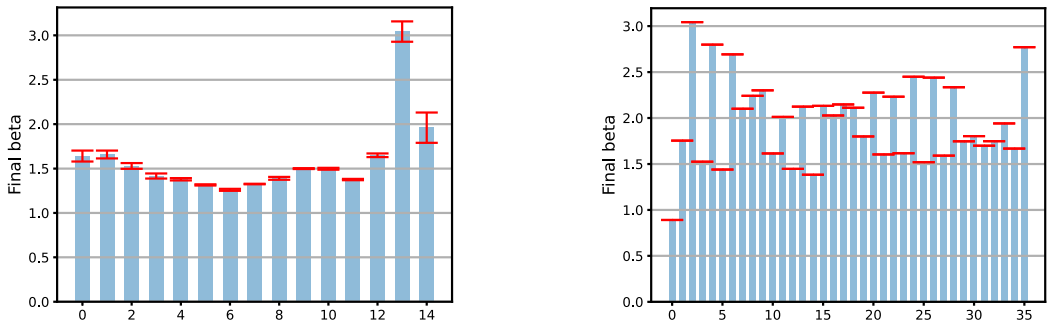

Figure 5. a: Final β of each layer in VGG16 experiment, the initial β is 1.0. b: Final β of each layer in Sew-ResNet34 experiment, the initial β is 1.0