Training High-performance Spiking Neural Networks Through Reducing Quantization Error

Yufei Guo*, Xinyi Tong*, Yuanpei Chen, Xiashuang Wang, Xiuhua Liu, and Liwen Zhang✉

X Lab, The Second Academy of China Aerospace Science and Industry Corporation, Beijing 100854, China
yfguo@pku.edu.cn, tongxinyi@buaa.edu.cn, lwzhang9161@126.com

Abstract

Spiking neural networks (SNNs) as a kind of efficient model by mimicking the spiking nature of brain neurons has have attracted more and more attention. It transmits binary spike signals between network units when the membrane potential exceeds the firing threshold. Benefit from the information paradigm, its activations can be limited to 1-bit spikes thus multiplications can be replaced by additions, which are more energy saving. However, quantifying the membrane potential to 0/1 spikes will inevitably induce the error. In this paper, we first notice the quantization error in SNNs and propose the membrane potential rectification (MPR) function. The MPR function can reduce quantization error by redistributing the membrane potential to a new value closer to spikes than itself. Experimental results show that SNNs with MPR function outperform their vanilla counterparts on CIFAR-10(100)

Introduction

Deep neural networks (DNNs) have demonstrated success in many fields, including object detection and recognition (He et al. 2016), object segmentation(Ronneberger, Fischer, and Brox 2015), object tracking(Bewley et al. 2016), etc. To further improve accuracy, more and more complex models are proposed, ranging from ResNet(He et al. 2016) to transformer (Vaswani et al. 2017), and so on. However, their increasing complexity poses a new challenge to deploy such models to power-constrained devices, thus becoming an impediment to widespread deployment in many applications. To address this problem, there have been several approaches developed, such as quantization(Gong et al. 2019; Li, Dong, and Wang 2019; Li et al. 2021b), pruning(Zhang, He, and Jian 2017), knowledge distillation(Polino, Pascanu, and Alistarh 2018), spiking neural networks(SNNs)(Fang et al. 2020; Wu et al. 2018a; Li et al. 2021a,c), and so on. Among them, SNNs as a biology-inspired method mimicking the spiking nature of brain neurons provide a unique way to reduce energy consumption. A spiking neuron integrates the inputs over time and fires a spike output whenever the membrane potential exceeds a threshold. The way dealing with 0/1 spike to transmit information enjoys the advantage of multiplication-free inference by converting multiplication to additions. Furthermore, SNNs can greatly save energy and run efficiently implemented on specialized neuromorphic hardwares, such as SpiNNaker(Khan et al. 2008), TrueNorth(Akopyan et al. 2015), Darwin(Shen et al. 2015), Tianjic(Pei et al. 2019), and Loihi(Davies et al. 2018), due to these hardwares provides a new non-von Neumann computing paradigm for SNNs that the storage unit and the computing unit are integrated thus eliminating the cost if data transfer.

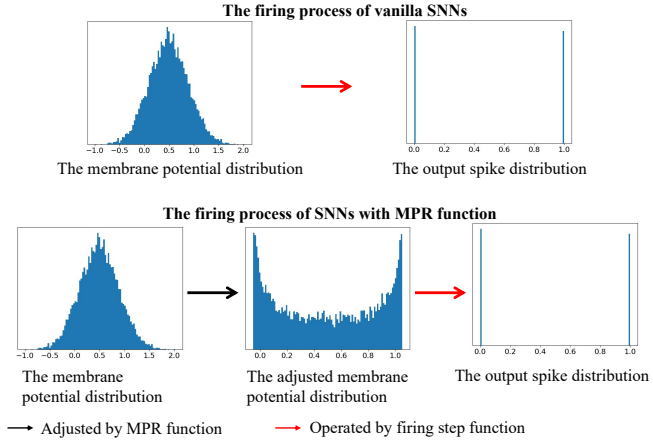

Figure 1: The difference of SNNs w/ & w/o MPR function for neuron firing process in a layer. The membrane potential will be redistributed to reduce the quantization error in the SNNs with MPR function.

Despite the attractive benefits, when transmitting information by quantifying the membrane potential to 0/1 spike, there is still a huge performance gap between existing SNN models and their DNN counterparts. Many works attribute the performance degradation to that the backward propagation can hardly access the accurate gradients caused by the discrete spike representation, and then surrogate gradient (SG) approaches are proposed by them(Wu et al. 2018b; Neftci, Mostafa, and Zenke 2019). In the literature, the rectangular function (Wu et al. 2018b,a; Zheng et al. 2020) has been widely used for approximation. Here, we provide a new perspective and argue that quantization itself inevitably brings large deviations between the original data and their

*Equal Contributions. ✉Corresponding author.

quantization values, thus inducing the error in information transmission and limited performance of SNNs.

To verify our guess, in this paper the Membrane Potential Rectification (MPR) function is introduced to reduce the quantization error through adjusting the membrane potential to approach the spikes (see the different neuron firing process of SNNs w/ & w/o MPR function in figure 1). Extensive experiments over CIFAR-10(100) show that SNNs with MPR can consistently outperform vanilla SNNs. To our best knowledge, this is the first work that has noticed the quantization error in SNNs and provides a simple but effective method to handle this problem.

Preliminary

An SNN adopts a new biology-inspired spiking neuron model that accumulates inputs along the time dimension as its membrane potential and fires a spike when the potential exceeds the threshold, which makes it much different from its DNN counterpart. Here, for better expression we will first introduce the special neuron model in SNNs, then the threshold-dependent Batch Normalization (tdBN), a normalization technique focuses on both spatial dimension and temporal dimension, which is also adopted in our paper.

Spiking Neuron Model

The fundamental computing unit of SNNs is the spiking neuron model. A unified model to describe the dynamics of all kinds of spiking neurons is given in a recent work (Fang et al. 2020), and we borrow the description here as follows.

$$H[t] = f(V[t-1], X[t]), \tag{1}$$

$$O[t] = \Theta(H[t] - V_{th}), \tag{2}$$

$$V[t] = H[t](1 - O[t]) + V_{reset}O[t], \tag{3}$$

where $X[t]$, $H[t]$, and $O[t]$ are the input, membrane potential, and output spike at the timestep t, respectively. V_{th} is the firing threshold, and is set as 0.5 in the work. Θ is the step function defined by $\Theta(x) = 1$ for $x \geq 0$ and $\Theta(x) = 0$ for $x < 0$. V_{reset} denotes the reset potential set as 0 in the paper. The different choice of function $f(\cdot)$ can describe different spiking neuron models. Leaky Integrate-and-Fire (LIF) model is adopted in the work and then Eq. 4 can be updated as

$$H[t] = V[t-1] + \tau X[t], \tag{4}$$

where τ denotes the membrane time constant and is 0.25 in our method.

Time-dependent Batch Normalization

Batch Normalization (BN) techniques are widely used to train very deep layers in DNNs, since they can avoid gradient vanishing or explosion during the optimization. However, BN is designed to normalize the spatial feature maps in DNNs, and the additional temporal dimension and special activation mechanism of SNNs need a specially-designed normalization method. Recently, a threshold-dependent Batch Normalization technique is proposed, which not only normalizes the feature maps of SNNs in the spatial paradigm but also in the temporal dimension (Zheng et al. 2020). In more detail, let $\mathbf{X}^t$ represent the a pre-synapse input maps at timestep t. Then $\mathbf{X} = (\mathbf{X}^1, \mathbf{X}^2, \cdots, \mathbf{X}^T)$ will be normalized by

$$\tilde{\mathbf{X}} = \frac{\alpha V_{th}(\mathbf{X} - E[\mathbf{X}])}{\sqrt{Var[\mathbf{X}] + \epsilon}}, \tag{5}$$

$$\mathbf{Y} = \lambda \tilde{\mathbf{X}} + \beta, \tag{6}$$

where α is a hyper-parameter, ϵ is a tiny constant, λ and β are two learnable parameters, $E[\mathbf{X}]$ and $Var[\mathbf{X}]$ are the mean and variance of $\mathbf{X}$ statistically estimated over the Mini-Batch. It can be seen that it's a bit like 3-dimension BN in DNNs, but the normalized tensor of tdBN will multiply αV_{th}, which can assist the SNN model to maintain appropriate firing rate. We adopt this technique in the work.

Methodology

In this section, we want to propose a function to redistribute the membrane potential before it is operated by the step function, Θ. Before we give the details of the MPR function, we try to formulate the quantization error first.

Quantization Error

Since the SNN needs to quantify the real values into $0/1$ spikes, it will undoubtedly result in the quantization error. However, the quantization errors corresponding to different membrane potentials are different. Obviously, a value closer to the quantization spike enjoys less quantization error. We define the quantization error as the square of the distance between the membrane potential and its corresponding quantization value the following:

$$\mathcal{L}_q = (x - v_q)^2 \tag{7}$$

Membrane Potential Rectification (MPR) Function

The MPR function should satisfy three conditions as follows:

- The value of derived membrane potential by it should be closer than the value of the initial membrane potential to ensure the quantization error reduction.
- It should not put a value less than V_{th} to a value greater than V_{th} and vice versa. Otherwise, the neuron output will be changed.
- When the value of membrane potential is 0/1, the obtained membrane potential value should be 0/1 at the same time.

Based on the above three points, we advise the MPR Function resort to an asymptotic symmetrical function:

$$\varphi(x) = \frac{1}{2tanh(k/2)} tanh(k(x - 1/2)) + \frac{1}{2}, \tag{8}$$

where the coefficient k determines the shape of the asymptotic function. Obviously, the above three conditions can all be satisfied by the recommended function. Fig. 2 illustrates the influences of k on the asymptotic function. When

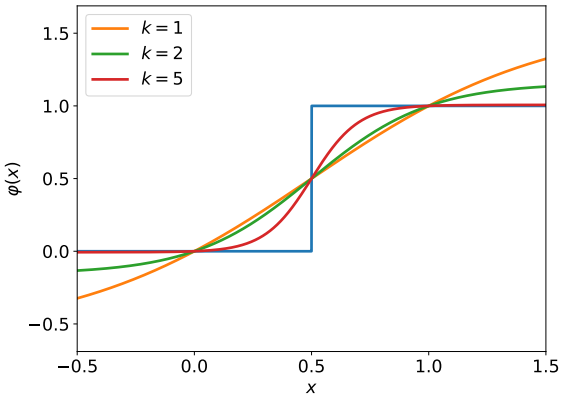

Figure 2: The response curves of asymptotic function under different values of the coefficient, k. The blue curves represent the spike activity function.

k becomes larger, the response curve of the MPR function gradually approaches the step function, and this undoubtedly will reduce the quantization error further. However, large k will induce the gradient vanishing and explosion problem in the backward propagation. Hence, an idea k should be chosen through experiments. It should be noted that the redistributed membrane potential by MPR function is only used for narrowing the gap between the true membrane potential and the quantization spike, but will not replace the true potential in the LIF neural model. The new dynamics of the LIF model can be updated as follows:

$$H[t] = V[t-1] + \tau X[t], \tag{9}$$

$$\hat{H}[t] = \varphi H[t], \tag{10}$$

$$O[t] = \Theta(\hat{H}[t] - V_{th}), \tag{11}$$

$$V[t] = H[t](1 - O[t]) + V_{reset}O[t] \tag{12}$$

Experiments

We evaluate the performance of SNNs with MPR function where the k ranges from 1 to 7 for classification tasks on CIFAR-10(Krizhevsky, Nair, and Hinton) and CIFAR-100(Krizhevsky, Nair, and Hinton) datasets. We employ the widely-used spiking ResNet20(Rathi and Roy 2020; Sengupta et al. 2019) as the backbone and encode the images to binary spike using the first layer of the SNN, as adopted in recent works (Rathi and Roy 2020; Fang et al. 2020, 2021). We adopt the SGD optimizer with 0.9 momentum and a learning rate of 0.01 cosine decayed(Loshchilov and Hutter 2016) to 0. The batch size is set to 128. The rectangular function is also appointed as the particular pseudo derivative of spike firing in the work. All of the experiment results are summarized in Tab.1.

It can be seen in the results, the SNN models with MPR function where k is no more than 5 can achieve higher accuracy than the vanilla SNN counterparts, and The SNN models with $k = 3$ can obtain the best results on both CIFAR10 and CIFAR100. It verifies our guess that the k of MPR function can not be too large, otherwise, the gradient vanishing and explosion problem will be induced and thus affects the SNN performance. The improvement by MPR function is notable. On CIFAR-10, our models can achieve 1.22%, 1.43%, and 0.69% top-1 accuracy increments with 2, 4, and 6 timesteps respectively. On CIFAR-100, MPR function demonstrates a more excellent ability. The SNN with the MPR function achieves a 3.16% absolute increment for timestep = 4.

Table 1: Results for MPR function.

Datasets	Timesteps	Methods	Accuracy
CIFAR-10	2	w/o MPR	89.29%
		w/ MPR, $k = 1$	90.26%
		w/ MPR, $k = 3$	90.51%
		w/ MPR, $k = 5$	89.80%
		w/ MPR, $k = 7$	87.21%
	4	w/o MPR	90.81%
		w/ MPR, $k = 1$	92.13%
		w/ MPR, $k = 3$	92.24%
		w/ MPR, $k = 5$	91.65%
		w/ MPR, $k = 7$	90.56%
	6	w/o MPR	91.62%
		w/ MPR, $k = 1$	92.69%
		w/ MPR, $k = 3$	92.84%
		w/ MPR, $k = 5$	92.31%
		w/ MPR, $k = 7$	91.33%
CIFAR-100	2	w/o MPR	62.59%
		w/ MPR, $k = 1$	63.03%
		w/ MPR, $k = 3$	63.29%
		w/ MPR, $k = 5$	63.14%
		w/ MPR, $k = 7$	60.58%
	4	w/o MPR	63.57%
		w/ MPR, $k = 1$	66.03%
		w/ MPR, $k = 3$	66.73%
		w/ MPR, $k = 5$	66.38%
		w/ MPR, $k = 7$	64.65%
	6	w/o MPR	65.09%
		w/ MPR, $k = 1$	67.43%
		w/ MPR, $k = 3$	67.61%
		w/ MPR, $k = 5$	67.57%
		w/ MPR, $k = 7$	65.21%

To further show the effect MPR function in SNNs intuitively, in Fig. 3, we visualize the comparison of the membrane potential before and after redistribution of the last layer of the first block in ResNet20 on CIFAR-10 respectively. From the figure, we can observe that the SNN with MPR function enjoys less quantization error. Experimental results also confirm the superiority of our method. We computed the average quantization error of the membrane potential before and after redistribution respectively. The average quantization error of the proposed method is 0.053, while that of the baseline method is 0.216, which is much bigger.

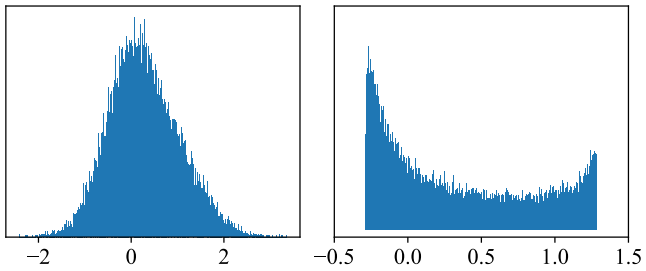

Figure 3: The membrane potential distribution of before (left) and after (right) being adjusted for the specific layer in ResNet20.

Conclusion

We first noticed the quantization error in the SNNs and presented the membrane potential rectification (MPR) function to solve it. Extensive experiments verified that SNNs with MPR function consistently achieve better performance than the vanilla SNNs. Although the improvement in model performance is very significant, more datasets and more neural backbones still need to be verified.

References

Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; and Modha, D. S. 2015. 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.

Bewley, A.; Ge, Z.; Ott, L.; Ramos, F.; and Upcroft, B. 2016. Simple Online and Realtime Tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*.

Davies, M.; Srinivasa, N.; Lin, T. H.; Chinya, G.; Joshi, P.; Lines, A.; Wild, A.; and Wang, H. 2018. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 82–99.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; and Tian, Y. 2021. Deep Residual Learning in Spiking Neural Networks.

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; and Tian, Y. 2020. Incorporating Learnable Membrane Time Constant to Enhance Learning of Spiking Neural Networks.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4852–4861.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Khan, M. M.; Lester, D. R.; Plana, L. A.; Rast, A. D.; and Furber, S. B. 2008. SpiNNaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor. In *IEEE International Joint Conference on Neural Networks*.

Krizhevsky, A.; Nair, V.; and Hinton, G. ???? CIFAR-10 (Canadian Institute for Advanced Research).

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021a. A Free Lunch From ANN: Towards Efficient, Accurate Spiking Neural Networks Calibration. *arXiv preprint arXiv:2106.06984*.

Li, Y.; Dong, X.; and Wang, W. 2019. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv preprint arXiv:1909.13144*.

Li, Y.; Gong, R.; Tan, X.; Yang, Y.; Hu, P.; Zhang, Q.; Yu, F.; Wang, W.; and Gu, S. 2021b. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*.

Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; and Gu, S. 2021c. Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks. *Advances in Neural Information Processing Systems*, 34.

Loshchilov, I.; and Hutter, F. 2016. SGDR: Stochastic Gradient Descent with Warm Restarts.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. Surrogate Gradient Learning in Spiking Neural Networks. *IEEE Signal Processing Magazine*, 36(6): 51–63.

Pei, J.; Deng, L.; Song, S.; Zhao, M.; and Shi, L. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767): 106.

Polino, A.; Pascanu, R.; and Alistarh, D. 2018. Model compression via distillation and quantization.

Rathi, N.; and Roy, K. 2020. DIET-SNN: Direct Input Encoding With Leakage and Threshold Optimization in Deep Spiking Neural Networks.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Springer International Publishing*.

Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going Deeper in Spiking Neural Networks: VGG and Residual Architectures. *Frontiers in Neuroscience*, 13.

Shen, J.; Ma, D.; Gu, Z.; Ming, Z.; and Pan, G. 2015. Darwin: a neuromorphic hardware co-processor based on Spiking Neural Networks. *Sciece China. Information Sciences*, 59(2): 1–5.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention Is All You Need. *arXiv*.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018a. Direct Training for Spiking Neural Networks: Faster, Larger, Better.

Wu, Y.; Lei, D.; Li, G.; Zhu, J.; and Shi, L. 2018b. Spatio-Temporal Backpropagation for Training High-performance Spiking Neural Networks. *Frontiers in Neuroscience*, 12: 331–.

Zhang, X.; He, Y.; and Jian, S. 2017. Channel Pruning for Accelerating Very Deep Neural Networks. In *IEEE International Conference on Computer Vision*.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2020. Going Deeper With Directly-Trained Larger Spiking Neural Networks.