RMP-Loss: Regularizing Membrane Potential Distribution for Spiking Neural Networks

Yufei Guo[*], Xiaode Liu[*], Yuanpei Chen, Liwen Zhang, Weihang Peng, Yuhan Zhang, Xuhui Huang, Zhe Ma[†]
Intelligent Science & Technology Academy of CASIC, China
Scientific Research Laboratory of Aerospace Intelligent Systems and Technology, China

Abstract

Spiking Neural Networks (SNNs) as one of the biology-inspired models have received much attention recently. It can significantly reduce energy consumption since they quantize the real-valued membrane potentials to 0/1 spikes to transmit information thus the multiplications of activations and weights can be replaced by additions when implemented on hardware. However, this quantization mechanism will inevitably introduce quantization error, thus causing catastrophic information loss. To address the quantization error problem, we propose a regularizing membrane potential loss (RMP-Loss) to adjust the distribution which is directly related to quantization error to a range close to the spikes. Our method is extremely simple to implement and straightforward to train an SNN. Furthermore, it is shown to consistently outperform previous state-of-the-art methods over different network architectures and datasets.

1. Introduction

Recently, many efforts have been done to make deep neural networks (DNNs) lightweight, so that they can be deployed in devices where energy consumption is limited. To this end, several approaches have been proposed, including network pruning [61], network quantization [18, 37, 15], knowledge transfer/distillation [46], neural architecture search [65, 39], and spiking neural networks (SNNs) [23, 19, 22, 17, 52, 36, 44, 55, 54, 63, 49, 56, 57]. The SNN provides a special way to reduce energy consumption following the working mechanism of the brain neuron. Its neurons accumulate spikes from previous neurons and present spikes to posterior neurons when the membrane potential exceeds the firing threshold. This information transmission paradigm will convert the computationally expensive multiplication to computationally convenient additions thus making SNNs energy-efficient when implemented on hardware. Specialized neuromorphic hardware based on an event-driven processing paradigm is currently under various stages of development, *e.g.*, SpiNNaker [31], TrueNorth [1], Tianjic [45], and Loihi [9], where SNNs can be efficiently implemented further. Due to the advantage of computational efficiency and rapid development of neuromorphic hardware, the SNN has gained more and more attention.

Although SNNs have been widely studied, their performance is still not comparable with that of DNNs. This performance gap is largely related to the quantization of the real-valued membrane potential to 0/1 spikes for the firing of the SNN in implementation [21]. The excessive information loss induced by the firing activity forcing all information only to two values will cause accuracy to decrease. Although information loss is important for computer vision tasks and the quantization will cause information loss too, the essential role of the activation function is to introduce non-linearity for neural networks [43]. Therefore, how to effectively reduce the information loss of membrane potential quantization is of high research importance. However, as far as we know, few studies have focused on directly solving this problem. The quantization error problem also exists in these methods that convert the ANN model to an SNN model [11, 4, 36]. However, these methods solve the problem by changing the activation function in ANNs or increasing timesteps in SNNs, which don't work for SNN training. InfLoR-SNN [21] adds a membrane potential redistribution function in the spiking neuron to reduce the quantization error via redistributing the membrane potential. However, it will decrease the biological plausibility of the SNN and increase the inference burden. This paper focuses on reducing the quantization error in SNN training directly and aims to introduce no burden for the SNN.

Quantization error is smaller when the membrane potential is close to the spiking threshold or reset values [21]. Hence, to mitigate the information loss, we suggest redistributing the membrane potential to where the membrane potential is closer to the 0/1 spike. Then an additional

[*]Equal contribution.
[†]Corresponding author, mazhe_thu@163.com.

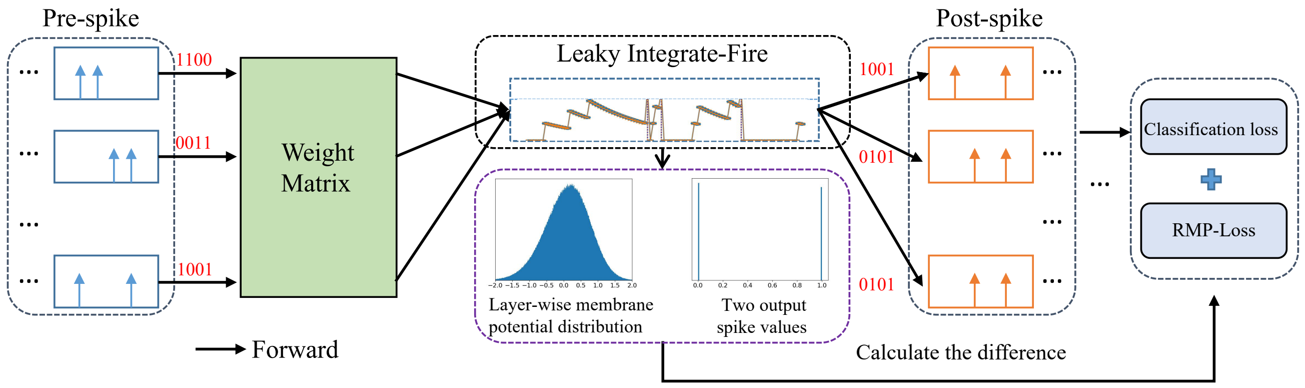

Figure 1: The overall workflow of the proposed method. We embed a membrane potential regularization loss in the task loss to redistribute the membrane potential in the training phase to reduce the quantization error.

loss term aims at **r**egularizing **m**embrane **p**otential is presented, called RMP-Loss, which can encourage the membrane potentials to gather around binary spike values during the training phase. The workflow of our method is shown in Fig. 1. Our main contributions can be concluded as follows:

- To our best knowledge, there have been few works noticing the quantization error in direct training of SNNs. To mitigate the quantization error, we present the RMP-Loss, which is of benefit to training an SNN model that enjoys a narrow gap between the membrane potential and its corresponding 0/1 spike. Furthermore, we also provide theoretical proof to clarify why the RMP-Loss can prevent information loss.
- Some existing methods can address information loss too. While achieving comparable performance, more parameters or computation burdens are also introduced in the inference phase. Different from those methods, the RMP-Loss can handle information loss directly without introducing any additional parameters or inference burden.
- Extensive experiments on both static and dynamic datasets show that our method performs better than many state-of-the-art SNN models.

2. Related Work

There are two main problems to train an SNN model [22]. The first is that the firing activity is non-differentiable thus the SNN model cannot be trained with these back-propagation methods directly. One way to mitigate the optimization difficulty is through surrogate functions. This kind of method replaces the non-differentiable firing activity with a differentiable surrogate function to calculate the gradient in the back-propagation [34, 51, 42, 50]. In [2], the derivative of a truncated quadratic function was used to approximate the derivative of the firing activity function. In [59] and [7], the derivatives of a sigmoid [59] and a rectangular function were respectively adopted to construct the surrogate gradient. Furthermore, a dynamic evolutionary surrogate gradient that could maintain accurate gradients during the training was proposed in [38, 20, 5, 6]. Another way to solve the non-differentiable problem is converting an ANN model to an SNN model, known as ANN-SNN conversion methods [27, 26, 36, 32, 3, 29, 28]. The ANN-SNN method trains a high-accuracy ANN with ReLU activation first and then transforms the network parameters into an SNN under the supposition that the activation of the ANN can be approximated by the average firing rates of an SNN. However, conversion methods will introduce conversion errors inevitably. Many efforts were made to tackle this problem, such as long inference time [14], threshold rescaling [48], soft reset [27], threshold shift [36], and the quantization clip-floor-shift activation function [4]. This paper adopts the surrogate gradient method.

The second problem is information loss. Though transmitting information with binary spikes is more efficient than that with real values, quantizing these real-valued membrane potentials to 0/1 spikes will cause information loss. There have been some works handling this problem indirectly. Such as learning the appropriate membrane leak-and-firing threshold [47, 64]. These are of benefit to finding a preferable quantization choice. Similarly, in [17, 58], some learnable neuron parameters are incorporated into the SNN to optimize the quantization choice. In [24], three regularization losses are introduced to penalize three undesired shifts of membrane potential distribution to make easy optimization and convergence for the SNN. The regularization is also beneficial to reduce quantization error. However, all these methods do not handle the information loss problem directly and most of them require more parameters. InfLoR-SNN [21] suggests adding a membrane potential redistribution function in the spiking neuron to redistribute the membrane potential closer to spike values, which can be seen as a direct method to reduce the quantization error. However, adding another function in spiking neurons will decrease the biological plausibility of the SNN and increase the inference burden. In this work, we focus on handling the quantization error problem directly without introducing extra parameters or computation burden in the inference phase.

3. Preliminary

In this section, a brief review of the primary computing element of SNNs is first provided in Sec. 3.1. Then we introduce the proposed surrogate gradient that handles the non-differentiability challenge in Sec. 3.2. Finally, in Sec. 3.3, we describe the threshold-dependent batch normalization technique used in this work.

3.1. Spiking Neuron Model

Spiking neuron. The primary computing element, *i.e.*, the neuron of an SNN is much different from that of an ANN. The neuron of an ANN plays the role of nonlinear transformation and can output real-valued values. Meanwhile, the neuron in an SNN accumulates the information from previous neurons into its membrane potential and presents the spike to the following neurons when its membrane potential exceeds the firing threshold. The SNN is more efficient with this special information transmission paradigm while suffering the information loss problem. A unified form of this spiking neuron model can be formulated as follows,

$$u^{(t),\mathrm{pre}} = \tau u^{(t-1)} + x^{(t)}, \tag{1}$$

$$o^{(t)} = \left\{ \begin{array}{ll} 1, & \text{if } u^{(t),\mathrm{pre}} \geq V_{\mathrm{th}} \\ 0, & \text{otherwise} \end{array} \right. , \tag{2}$$

$$u^{(t)} = u^{(t),\mathrm{pre}}(1 - o^{(t)}). \tag{3}$$

Where $u^{(t),\mathrm{pre}}$ and $u^{(t)}$ are pre-membrane potential and membrane potential, respectively, $x^{(t)}$ is input information, $o^{(t)}$ is output spike at the timestep t. The range of constant leaky factor τ is $(0, 1)$, and we set τ as 0.25. V_{th} is the firing threshold and is set to 0.5 here.

Output layer neuron. In the general classification task, the output of the last layer will be presented to the `Softmax` function first and then find the winner class. We make the neuron model in the output layer only accumulate the incoming inputs without any leakage as final output like doing in recent practices[47, 62], described by

$$u^{(t)} = u^{(t-1)} + x^{(t)}. \tag{4}$$

Then the cross-entropy loss is calculated according to the final membrane potential at the last timesteps, $u^{(T)}$.

3.2. Surrogate Gradients of SNNs

As shown in Eq. 2. the firing activity of SNNs can be seen as a step function. Its derivative is 0 everywhere except at V_{th}. This non-differentiability problem of firing activity will cause gradient vanishing or explosion and make the back-propagation unsuitable for training SNNs, directly. As mentioned before, many prior works adopted the surrogate function to replace the firing function to obtain a suitable gradient [7, 62, 52]. In InfLoR-SNN [21], an extra membrane potential redistribution function is added before the firing function. It argues that this redistribution function will reduce the information loss in forward propagation. However, investigating it from a backward propagation perspective and considering the redistribution function being next to the firing function, it can be seen as a more suitable surrogate gradient if we gather the STE (used in [21]) gradients of the firing function and redistribution function together. In this sense, the suitable surrogate gradient can also be seen as a route to reduce quantization error. Therefore, in this paper, we adopt the gathered redistribution function in InfLoR-SNN as our surrogate function, which indeed performs better than other methods [7, 62, 52] in our experiments, denoted as

$$\varphi(u) = \left\{ \begin{array}{ll} 0, & u<0, \\ \frac{1}{2\tanh(3/2)}\tanh(3(u-1/2)) + 1/2, & 0 \leq u \leq 1, \\ 1, & u>1. \end{array} \right. \tag{5}$$

3.3. Threshold-dependent Batch Normalization

Normalization techniques can effectively reduce the training time and alleviate the gradient vanishing or explosion problem for training DNNs. A most widely used technique called batch normalization (BN) [30] uses the distribution of the summed input to a neuron over a mini-batch of training cases to compute a mean and variance which are then used to normalize the summed input to that neuron on each training case. This is significantly effective for convolutional neural networks (CNNs). However, directly applying BN to SNNs will ignore the temporal characteristic and can not achieve the desired effect. To this end, a more suitable normalization technique for SNNs, named threshold-dependent BN (tdBN) was further proposed in [62]. It normalizes feature inputs on both temporal and spatial dimensions. Besides, as its name implies, tdBN makes normalized variance dependent on the firing threshold, *i.e.*, the pre-activations are normalized to $N(0, (\alpha V_{\mathrm{th}})^2)$ instead of $N(0, 1)$. this can balance pre-synaptic input and threshold to maintain a reasonable firing rate. In this paper, we also adopt the tdBN, as follows,

$$\bar{\mathbf{X}} = \alpha V_{\mathrm{th}} \frac{\mathbf{X} - \mathrm{mean}(\mathbf{X})}{\sqrt{\mathrm{mean}((\mathbf{X} - \mathrm{mean}(\mathbf{X}))^2) + \epsilon}}, \tag{6}$$

$$\tilde{\mathbf{X}} = \lambda\bar{\mathbf{X}} + \beta. \tag{7}$$

where $\mathbf{X} \in \mathbb{R}^{T\times B\times C\times H\times W}$ (T: timesteps; B: batch size; C: channel; (H, W): spatial domain) is a 5D tensor and α is a hyper-parameter set as 1.

4. RMP-Loss

As aforementioned, we suggest reducing the quantization error to avoid information loss in supervised training-based SNNs. In this section, we first try to provide a metric to measure the quantization error in SNNs. Then based on the metric, we introduce our method, a loss term for regularizing membrane potential, *i.e.*, RMP-Loss, which can push the membrane potentials close to spike values to reduce the quantization error. Finally, theoretical proof to clarify why the RMP-Loss can prevent information loss is given.

4.1. Quantization Error Estimation

Estimating the quantization error precisely is important for designing a method to reduce the prediction error. Hence, before giving the full details of the RMP-Loss, here we try to formulate the quantization error first. As shown in Eq. 2, the output of a neuron is dependent on the magnitude of the membrane potential. When the membrane potential exceeds the firing threshold, the output is 1, otherwise, is 0; that is, these full-precision values will be mapped to only two spike values. The difference between output, o, and membrane potential, u, can be measured as

$$\mathrm{Dis}(o,u) = \begin{cases} 1-u, & \text{if } u \geq V_{\mathrm{th}} \\ 0-u, & \text{otherwise} \end{cases}. \tag{8}$$

Obviously, the closer a membrane potential is to its corresponding spike, the smaller the quantization error. The quantization error depends on the margin between the membrane potential and its corresponding spike. Then we define the quantization error as

$$q(u) = |\mathrm{Dis}(o,u)|^p, \tag{9}$$

where $p > 0$, which is usually set to 1 and/or 2 corresponding to L1-norm and/or L2-norm. Unless otherwise specified, we set it as 2 in the paper.

4.2. Regularizing Membrane Potential Distribution

To mitigate the information loss problem in SNNs, the quantization error should be reduced to some extent. To this end, we propose RMP-Loss to carve the distribution to a range close to the spikes based on $q(u)$, as

$$\mathcal{L}_{\mathrm{RMP}} = \frac{1}{TLBCWH}\sum_{t,l,b,c,w,h} q(u). \tag{10}$$

where T, L, B, C, W, and H denote the number of timesteps, the number of layers, batch size, channel length, width, and height of the membrane potential map, respectively. Finally, taking classification loss into consideration, the total loss can be written as

Algorithm 1 Training an SNN with RMP-Loss.

Input: An SNN to be trained; Training dataset, total number of training epochs N, total number of training iterations per epoch I.

Output: The trained SNN.

```
1:  for All $n = 1, 2, \ldots, N-th$ epoch do
2:    for All $i = 1, 2, \ldots, I-th$ iteration do
3:      Feed-Forward:
4:      Calculate the SNN output, $\mathbf{O}^i$, membrane potential, $\mathbf{U}$, and class label, $\mathbf{Y}^i$;
5:      Compute classification loss $\mathcal{L}_{\mathrm{CE}} = \mathcal{L}_{\mathrm{CE}}(\mathbf{O}, \mathbf{Y}^i)$;
6:      Compute RMP-Loss $\mathcal{L}_{\mathrm{RMP}} = \mathcal{L}_{\mathrm{RMP}}(\mathbf{U})$;
7:      Compute the total loss $\mathcal{L}_{\mathrm{CE-RMP}} = \mathcal{L}_{\mathrm{CE}} + \lambda(n)\mathcal{L}_{\mathrm{RMP}}$.
        Backward-Propagation:
8:      Calculate the gradients $\Delta\mathbf{W} = \frac{\partial\mathcal{L}_{\mathrm{CE-RMP}}}{\partial\mathbf{W}} = \sum_t \frac{\partial(\mathcal{L}_{\mathrm{CE}}+\lambda(n)\mathcal{L}_{\mathrm{RMP}})}{\partial\mathbf{y}^t}\frac{\partial\mathbf{y}^t}{\partial\mathbf{U}}\frac{\partial\mathbf{U}}{\partial\mathbf{W}}$, where $\mathbf{y}^t$ is feature maps at $t$-th timesteps and $\frac{\partial\mathbf{y}^t}{\partial\mathbf{U}}$ is the gradient of firing function, which can be calculated as $\frac{\partial\varphi(\mathbf{U})}{\partial\mathbf{U}}$.
        Parameters Update:
9:      Update all parameters with the learning rate $\eta$.
10:   end for
11: end for
12: return the trained SNN.
```

$$\mathcal{L}_{\mathrm{CE-RMP}} = \mathcal{L}_{\mathrm{CE}} + \lambda(n)\mathcal{L}_{\mathrm{RMP}}, \tag{11}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss, and $\lambda(n)$ is a balanced coefficient, which changes dynamically with the training epoch. Obviously, $\mathcal{L}_{RMP}$ does not take into account the classification problem at hand and will add a new constraint in the network optimization. To make the network parameter update have enough freedom at the beginning, which is important to train a well-performance network, and still focus on the classification task at the end. We adopt a strategy of increasing first and decreasing later to adjust the $\lambda(n)$ as follows,

$$\lambda(n) = \begin{cases} 2k\frac{n}{N}, & \text{if } n \leq N/2 \\ 2k(1-\frac{n}{N}), & \text{otherwise} \end{cases}, \tag{12}$$

where n is the n-th epoch and k is a coefficient, which controls the amplitude of regularization. Unless otherwise specified, we set it as 0.1 in the paper.

The final resulting RMP-Loss training method is defined in Algo. 1.

4.3. Analysis and Discussion

In this work, we assume that RMP-Loss can help reduce the quantization loss of the SNNs. To verify our assumption, a theoretical analysis is conducted by using the information entropy concept. To mitigate the information loss, the output spike tensor **O** should reflect the information of the membrane potential tensor **U** as much as possible. Since KL-divergence is a metric to measure the difference between two random variable distributions. We use KL-divergence to compute the difference between **O** and **U**, which refers to the difficulty of reconstructing **U** from **O**, *i.e.*, information loss degree. Then we provide the theoretical analysis below.

Let $x^u \sim \mathcal{U}$ where x^u are samples $\in$ **U** and $\mathcal{U}$ represents the distribution of x^u. Similarily, we have $x^o \sim \mathcal{O}$ where $x^o \in \mathbf{O}$. We use $P_U(x)$ and $P_O(x)$ to denote the probability density function for $\mathcal{U}$ and $\mathcal{O}$, respectively. Then the information loss for the spike firing process can be described as

$$\mathcal{L}_{KL}(\mathcal{U}||\mathcal{O}) = \int_{-\infty}^{\infty} P_U(x^u)\log\frac{P_U(x^u)}{P_O(x^o)}dx. \tag{13}$$

Since output spike x^o is discrete, we can update it as

$$\mathcal{L}_{KL}(\mathcal{U}||\mathcal{O}) = \int_{o-\epsilon}^{o+\epsilon} P_U(x^u)\log\frac{P_U(x^u)}{P_O(x^o)}dx, \tag{14}$$

where ϵ is a small constant, o is the spike value, $P_O(x^o)$ can be seen as a very large constant in this situation, and $\int_{o-\epsilon}^{o+\epsilon} P_O(x^o)dx = 1$. With RMP-Loss, it will become

$$\mathcal{L}_{KL}(\hat{\mathcal{U}}||\mathcal{O}) = \int_{o-\epsilon}^{o+\epsilon} P_{\hat{U}}(x^{\hat{u}})\log\frac{P_{\hat{U}}(x^{\hat{u}})}{P_O(x^o)}dx, \tag{15}$$

where $x^{\hat{u}} \in \hat{\mathcal{U}}$ and $\hat{\mathcal{U}}$ is the new distribution of $x^{\hat{u}}$ adjusted by RMP-Loss. Here, we have the following propositions.

Proposition 1 $\frac{d\mathcal{L}_{KL}}{dP_U(x^u)} < 0$, *i.e.,* $\mathcal{L}_{KL} \downarrow$ *as* $P_U(x^u) \uparrow$.

proof: When a membrane potential u is below the firing threshold, it will be pushed to 0 by the RMP-Loss, considering its effect, and otherwise, 1. Hence the firing rate of the neuron can be seen as the same, whether using the RMP-Loss or not, *i.e.*, $P_O(x^o)$ keep the same for $\hat{\mathbf{U}}$ and **U**.

$$\frac{d\mathcal{L}_{KL}}{dP_U(x^u)} = \frac{d\int_{o-\epsilon}^{o+\epsilon} P_U(x^u)\log\frac{P_U(x^u)}{P_O(x^o)}dx}{dP_U(x^u)} \tag{16}$$

$$= \int_{o-\epsilon}^{o+\epsilon}(\log\frac{P_U(x^u)}{P_O(x^o)} + \frac{1}{\ln 2})dx. \tag{17}$$

Since $P_O(x^o)$ is much larger than $P_U(x^u)$, we can get $\int_{o-\epsilon}^{o+\epsilon}(\log\frac{P_U(x^u)}{P_O(x^o)} + \frac{1}{\ln 2})dx < 0$, then $\frac{d\mathcal{L}_{KL}}{dP_U(x^u)} < 0$. ■

Proposition 2 $P_{\hat{U}}(x^{\hat{u}}) > P_U(x^u)|x = o$.

proof: We assume that there are n_ϵ^u samples in the interval $(o-\epsilon, o+\epsilon)$ for $\mathcal{U}$, where ϵ is a small constant. Then we can get $P_U(x^u) \approx \frac{n_\epsilon^u}{2\epsilon}$. And we can simply see the RMP-Loss as a function to push the vanilla **U** to $\hat{\mathbf{U}}$, which is closer to **O**, based on its effect. Therefore, we can assume that these samples will be gathered to a new interval $(o-\epsilon_l, o+\epsilon_r)$, where $\epsilon_l, \epsilon_r < \epsilon$. Then we can get $P_{\hat{U}(x^{\hat{u}})} \approx \frac{n_\epsilon^u}{\epsilon_l+\epsilon_r}$. Thereby we can have $P_{\hat{U}(x^{\hat{u}})} > P_U(x^u)|_{x=o}$. ■

Along with **Proposition 1** and **Proposition 2**, we can conclude that $\mathcal{L}_{KL}(\hat{U}||O) < \mathcal{L}_{KL}(U||O)$, *i.e.*, our method with RMP-Loss enjoys lesser information loss.

5. Experiment

In this section, we first conducted extensive ablation studies to compare the SNNs with RMP-Loss and their vanilla counterparts to verify the effectiveness of the method. Next, we fully compared our method with other state-of-the-art (SoTA) methods. Finally, some further experimental visualizations are provided to understand the RMP-Loss.

5.1. Datasets and Settings

Datasets. We conducted experiments on four benchmark datasets: CIFAR-10 (100) [33], CIFAR10-DVS [35], and ImageNet (ILSVRC12) [10]. The CIFAR-10 (100) dataset includes 60,000 images in 10 (100) classes with 32×32 pixels. The numbers of training images and test images are 50,000 and 10,000. The CIFAR10-DVS dataset is the neuromorphic version of the CIFAR-10 dataset. It is composed of 10,000 images in 10 classes. We split the dataset into 9000 training images and 1000 test images similar to [52]. ImageNet dataset consists of 1,250,000 training images and 50,000 test images.

Preprocessing. We applied data normalization on all static datasets to make input images have 0 mean and 1 variance. Besides, we conducted random horizontal flipping and cropping on these datasets to avoid overfitting. For CIFAR, the AutoAugment [8] and Cutout [13] were also used for data augmentation as doing in [24, 20]. For the neuromorphic dataset, we resized the training image frames to 48×48 as in [62] and adopted random horizontal flip and random roll within 5 pixels for augmentation. And the test images were merely resized to 48×48 without any additional processing.

Training setup. For all the datasets, the firing threshold V_{th} was set as 0.5. For static image datasets, the images were encoded to binary spike using the first layer of the SNN, as in recent works [47, 17, 16]. This is similar to rate coding. For the neuromorphic image dataset, we used the 0/1 spike format directly. The neuron models in the output layer accumulated the incoming inputs without generating any spike as the output like in [47]. For CIFAR-10 (100) and CIFAR10-DVS datasets, we used the SGD optimizer with the momentum of 0.9 and learning rate of 0.01 with cosine decayed [40] to 0 as in [21]. All models were trained within 400 epochs with the same batch size of 128. For the ImageNet dataset, we adopted the SGD optimizer with

Table 1: Ablation Study for RMP-Loss

Dataset	Architecture	Type	Timestep	Accuracy	Quantization error
CIFAR-10	ResNet20	Without RMP-Loss	4	91.26%	0.186
		With RMP-Loss	4	**91.89%**	0.121
	ResNet19	Without RMP-Loss	4	95.04%	0.128
		With RMP-Loss	4	**95.51%**	0.104
	VGG16	Without RMP-Loss	4	92.80%	0.174
		With RMP-Loss	4	**93.33%**	0.135

a momentum of 0.9 and a learning rate of 0.1 with cosine decayed to 0. All models are trained within 320 epochs.

5.2. Ablation Study for RMP-Loss

We conducted a set of ablation experiments on CIFAR-10 using ResNet20, ResNet19, and VGG16 as backbones. The results are shown in Tab. 1. It can be seen that with the RMP-Loss, these SNNs can achieve higher accuracy than their vanilla counterparts. We also show the membrane potential distribution of the first layer of the second block in the ResNet20 with and without RMP-Loss on the test set of CIFAR-10 in Fig. 2. It can be seen that the models trained with RMP-Loss can shrink the membrane potential distribution range which enjoys less quantization error.

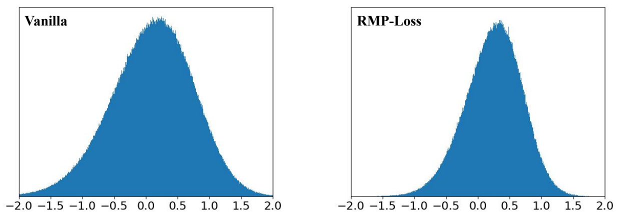

Figure 2: The effect of RMP-Loss. The overall original membrane potential distribution (left) and the redistributed membrane potential distribution by RMP-Loss (right) of the first layer of the second block in ResNet20 on CIFAR-10 test sets.

5.3. Comparison with State-of-the-art Methods

We evaluated the proposed method with the accuracy performance on various widely used static and neuromorphic datasets using spiking ResNet20 [47, 48], VGG16 [47], ResNet18 [16], ResNet19 [62], and ResNet34 [16]. The results with the mean accuracy and standard deviation of 3-trials are listed in Tab. 2.

CIFAR-10. For CIFAR-10, we tested three different network architectures. The performance of our method is shown in Tab. 2. It can be seen that our method achieves the best results in most of these cases. In special, using ResNet19 [62], the RMP-Loss achieves 96.10% averaged top-1 accuracy, which improves 1.60% absolute accuracy compared with the existing state-of-the-art TET [12]. Though InfLoR-SNN [21] is slightly better than our method for 4 and 6 timesteps, it will decrease the biological plausibility of the SNN and increase the inference burden since a complex transform function is added in its spiking neuron. On ResNet20, our method can achieve higher accuracy with only 6 timesteps, while Diet-SNN [47] with 10 timesteps. On VGG16, the RMP-Loss also shows obvious advantages.

CIFAR-100. For CIFAR-100, we also experimented with these three different network structures. For all these configurations, our method achieves consistent and significant accuracy over prior work. ResNet20-based and VGG16-based RMP-Loss achieve 66.65% and 72.55% top-1 accuracy with only 4 timesteps, which outperform their Diet-SNN counterparts with 2.58% and 2.88% higher accuracy and Real spike counterparts with 0.05% and 1.93% respectively but with fewer timesteps. Noteworthy, our method significantly surpasses TET with 4.26% higher accuracy on ResNet19, which is not easy to achieve in the SNN field. Overall, our results on CIFAR-100 show that, when applied to a more complex dataset, the RMP-Loss achieves an even more favorable performance compared to competing methods.

CIFAR10-DVS. We also verified our method on the popular neuromorphic dataset, CIFAR10-DVS. It can be seen that RMP-Loss also shows amazing performance. RMP-Loss outperforms STBP-tdBN by 12.39%, RecDis-SNN by 3.78%, and InfLoR-SNN by 0.70% respectively with 76.20% top-1 accuracy in 10 timesteps using ResNet19 as the backbone. With ResNet20 as the backbone, RMP-Loss can also achieve well-performed results.

ImageNet. For ImageNet, we conducted experiments with ResNet18 and ResNet34 as backbones. Our results are presented in Tab. 4. In these normal spiking structures, our method achieves the highest accuracy among these SoTA prior works. However, our method is slightly worse than SEW ResNet [16]. This is because that SEW ResNet uses an atypical architecture, which abandons binary spike forms but will output arbitrary integers to transmit information, thus enjoying higher accuracy but the efficiency from mul-

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

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR-10	STBP [52]	SNN training	CIFARNet	12	90.53%
	TSSL-BP [60]	SNN training	CIFARNet	5	91.41%
	PLIF [17]	SNN training	PLIFNet	8	93.50%
	DSR [41]	SNN training	ResNet18	20	95.40%
	Joint A-SNN [23]	SNN training	ResNet18	4	95.45%
	Diet-SNN [47]	SNN training	VGG16	5	92.70%
				10	93.44%
			ResNet20	5	91.78%
				10	92.54%
	RecDis-SNN [24]	SNN training	ResNet19	2	93.64%
				4	95.53%
				6	95.55%
	STBP-tdBN [62]	SNN training	ResNet19	2	92.34%
				4	92.92%
				6	93.16%
	TET [12]	SNN training	ResNet19	2	94.16%
				4	94.44%
				6	94.50%
	InfLoR-SNN [21]	SNN training	ResNet19	2	94.44%
				4	96.27%
				6	96.49%
	RMP-Loss	SNN training	ResNet19	2	**95.31%**±0.07
				4	**95.51%**±0.08
				6	**96.10%**±0.08
			ResNet20	4	**91.89%**±0.05
				6	**92.55%**±0.06
			VGG16	4	**93.33%**±0.07
				10	**94.39%**±0.08
CIFAR-100	DSR [41]	SNN training	ResNet18	20	78.50%
	InfLoR-SNN [21]	SNN training	VGG16	5	71.56%
	IM-Loss [20]	SNN training	VGG16	5	70.18%
	Diet-SNN [47]	SNN training	ResNet20	5	64.07%
			VGG16	5	69.67%
	Real Spike [25]	SNN training	ResNet20	5	66.60%
			VGG16	5	70.62%
	TET [12]	SNN training	ResNet19	2	72.87%
				4	74.47%
				6	74.72%
	RMP-Loss	SNN training	ResNet20	4	**66.65%**±0.10
			VGG16	4	**72.55%**±0.08
				10	**73.30%**±0.11
			ResNet19	2	**74.66%**±0.12
				4	**78.28%**±0.10
				6	**78.98%**±0.08

tiplication free of SNNs will be lost. Hence, it would be limited in some applications while our method not.

5.4. Visualization

Furthermore, we also provide some experimental visualizations to show the regularizing effects of RMP-Loss.

Table 3: Comparison with SoTA methods on CIFAR10-DVS.

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR10-DVS	STBP-tdBN [62]	SNN training	ResNet19	10	67.80%
	LIAF [53]	Conv3D	LIAF-Net	10	71.70%
	LIAF [53]	LIAF	LIAF-Net	10	70.40%
	RecDis-SNN [24]	SNN training	ResNet19	10	72.42%
	InfLoR-SNN [21]	SNN training	ResNet19	10	75.50%
			ResNet20	10	75.10%
	RMP-Loss	SNN training	ResNet19	10	**76.20%**±0.20
			ResNet20	10	**75.60%**±0.30

Table 4: Comparison with SoTA methods on ImageNet.

Method	Type	Architecture	Timestep	Spike form	Accuracy
STBP-tdBN [62]	SNN training	ResNet34	6	Binary	63.72%
TET [12]	SNN training	ResNet34	6	Binary	64.79%
SEW ResNet [16]	SNN training	ResNet18	4	Integer	**63.18%**
		ResNet34	4	Integer	**67.04%**
Spiking ResNet [16]	SNN training	ResNet18	4	Binary	62.32%
		ResNet34	4	Binary	61.86%
RMP-Loss	SNN training	ResNet18	4	Binary	63.03%±0.07
		ResNet34	4	Binary	65.17%±0.07

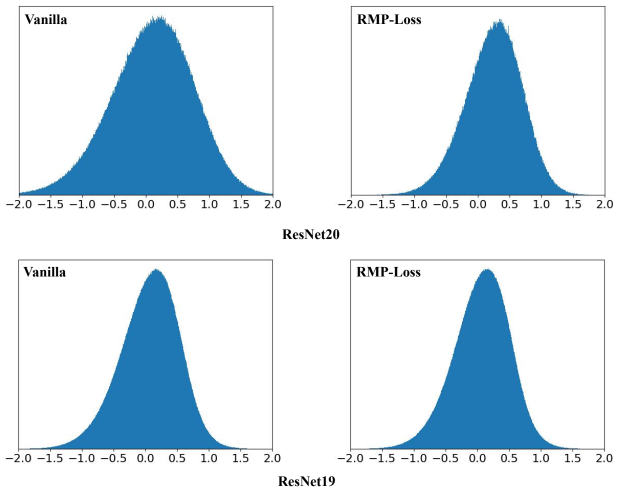

Figure 3: The original membrane potential distribution (left) and the redistributed membrane potential distribution by RMP-Loss (right) of the first layer of the second block in ResNet20/19 on CIFAR-10 test sets.

Fig. 3 shows the membrane potential distribution of the first layer of the second block in the ResNet20/19 with and without RMP-Loss on the test set of CIFAR-10. It can be seen that the SNN models trained with RMP-Loss can shrink the membrane potential distribution range which enjoys less quantization error. On the other hand, comparing the membrane potential distribution difference between ResNet20 and ResNet19, it can be found that the membrane potential distribution of ResNet19 is thinner than that of ResNet20 too. Considering that ResNet19 can achieve higher accuracy than ResNet20, it also shows that reducing quantization error is effective to improve the accuracy of SNN models and our route to improve the SNN accuracy by reducing the quantization error is reasonable.

6. Conclusion

This paper aims at addressing the information loss problem caused by the $0/1$ spike quantization of SNNs. We introduce RMP-Loss to adjust the membrane potential to reduce the quantization error. Different from other methods that reduce the quantization error indirectly or will induce more parameters, RMP-Loss focuses on handling this problem directly and will introduce no extra parameters in the inference phase. We show that our method outperforms SoTA methods on both static and neuromorphic datasets.

Acknowledgment

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

References

[1] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, and Dharmendra S. Modha. 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. 1

[2] S. M. Bohte. Error-backpropagation in networks of fractionally predictive spiking neurons. *Springer Berlin Heidelberg*, 2011. 2

[3] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11–20, 2022. 2

[4] Tong Bu, Wei Fang, Jianhao Ding, PENGLIN DAI, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022. 1, 2

[5] Kaiwei Che, Luziwei Leng, Kaixuan Zhang, Jianguo Zhang, Qinghu Meng, Jie Cheng, Qinghai Guo, and Jianxing Liao. Differentiable hierarchical and surrogate gradient search for spiking neural networks. In *Advances in Neural Information Processing Systems*. 2

[6] Yi Chen, Silin Zhang, Shiyu Ren, and et al. Gradual surrogate gradient learning in deep spiking neural networks. In *ICASSP*, pages 8927–8931, 2022. 2

[7] X. Cheng, Y. Hao, J. Xu, and B. Xu. Lisnn: Improving spiking neural networks with lateral interactions for robust object recognition. In *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence IJCAI-PRICAI-20*, 2020. 2, 3

[8] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data, 2019. 5

[9] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, pages 82–99, 2018. 1

[10] J. Deng, W. Dong, R Socher, L. J. Li, and F. F. Li. Imagenet: a large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 2009. 5

[11] S. Deng and S. Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. 2021. 1

[12] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022. 6, 7, 8

[13] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout, 2017. 5

[14] P. U. Diehl, D. Neil, J. Binas, M. Cook, and S. C. Liu. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, 2015. 2

[15] Jason K Eshraghian, Corey Lammie, Mostafa Rahimi Azghadi, and Wei D Lu. Navigating local minima in quantized spiking neural networks. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 352–355. IEEE, 2022. 1

[16] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, and Yonghong Tian. Deep residual learning in spiking neural networks. 2021. 5, 6, 8

[17] W. Fang, Z. Yu, Y. Chen, T. Masquelier, and Y. Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. 2020. 1, 2, 5, 7

[18] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4852–4861, 2019. 1

[19] Yufei Guo and Yuanpei Chen. Neuroclip: Neuromorphic data understanding by clip and snn. *arXiv preprint arXiv:2306.12073*, 2023. 1

[20] Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. Im-loss: Information maximization loss for spiking neural networks. In *Advances in Neural Information Processing Systems*. 2, 5, 7

[21] Yufei Guo, Yuanpei Chen, Liwen Zhang, YingLei Wang, Xiaode Liu, Xinyi Tong, Yuanyuan Ou, Xuhui Huang, and Zhe Ma. Reducing information loss for spiking neural networks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pages 36–52. Springer, 2022. 1, 2, 3, 5, 6, 7, 8

[22] Yufei Guo, Xuhui Huang, and Zhe Ma. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023. 1, 2

[23] Yufei Guo, Weihang Peng, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Xuhui Huang, and Zhe Ma. Joint a-snn: Joint training of artificial and spiking neural networks via self-distillation and weight factorization. *Pattern Recognition*, page 109639, 2023. 1, 7

[24] Yufei Guo, Xinyi Tong, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, and Xuhui Huang. 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)*, pages 326–335, June 2022. 2, 5, 7, 8

[25] Yufei Guo, Liwen Zhang, Yuanpei Chen, Xinyi Tong, Xiaode Liu, YingLei Wang, Xuhui Huang, and Zhe Ma. 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*, pages 52–68. Springer, 2022. 7

[26] B. Han and K. Roy. *Deep Spiking Neural Network: Energy Efficiency Through Time Based Coding*. Computer Vision – ECCV 2020, 2020. 2

[27] B. Han, G. Srinivasan, and K Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. *IEEE*, 2020. 2

[28] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential, 2023. 2

[29] Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes, 2023. 2

[30] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv e-prints*, page arXiv:1502.03167, Feb. 2015. 3

[31] M. M. Khan, D. R. Lester, L. A. Plana, A. D. Rast, and S. B. Furber. Spinnaker: Mapping neural networks onto a massively-parallel chip multiprocessor. In *IEEE International Joint Conference on Neural Networks*, 2008. 1

[32] S. Kim, S. Park, B. Na, and S. Yoon. Spiking-yolo: Spiking neural network for energy-efficient object detection. 2019. 2

[33] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 5

[34] J. H. Lee, T. Delbruck, and M. Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10, 2016. 2

[35] H. Li, H. Liu, X. Ji, G. Li, and L. Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11, 2017. 5

[36] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. *arXiv preprint arXiv:2106.06984*, 2021. 1, 2

[37] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 1

[38] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34, 2021. 2

[39] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1

[40] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. 2016. 5

[41] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation, 2023. 7

[42] E. Neftci, C. Augustine, S. Paul, and G. Detorakis. Event-driven random backpropagation: Enabling neuromorphic deep learning machines. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017. 2

[43] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research for deep learning. 2018. 1

[44] Alexander Ororbia. Spiking neural predictive coding for continual learning from data streams. *arXiv preprint arXiv:1908.08655*, 2019. 1

[45] J. Pei, L. Deng, S. Song, M. Zhao, and L. Shi. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106, 2019. 1

[46] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. 2018. 1

[47] N. Rathi and K. Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. 2020. 2, 3, 5, 6, 7

[48] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in Neuroscience*, 13, 2019. 2, 6

[49] Jiangrong Shen, Qi Xu, Jian K Liu, Yueming Wang, Gang Pan, and Huajin Tang. Esl-snns: An evolutionary structure learning strategy for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 86–93, 2023. 1

[50] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. 2018. 2

[51] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. 2017. 2

[52] Y Wu, L. Deng, G. Li, J. Zhu, and L. Shi. Direct training for spiking neural networks: Faster, larger, better. 2018. 1, 3, 5, 7

[53] Zhenzhi Wu, Hehui Zhang, Yihan Lin, Guoqi Li, Meng Wang, and Ye Tang. LIAF-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–14, 2021. 8

[54] Qi Xu, Yaxin Li, Xuanye Fang, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. Biologically inspired structure learning with reverse knowledge distillation for spiking neural networks. *arXiv preprint arXiv:2304.09500*, 2023. 1

[55] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. *arXiv preprint arXiv:2304.05627*, 2023. 1

[56] Qi Xu, Yaxin Li, Jiangrong Shen, Pingping Zhang, Jian K Liu, Huajin Tang, and Gang Pan. Hierarchical spiking-based model for efficient image classification with enhanced feature extraction and encoding. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 1

[57] Qi Xu, Jiangrong Shen, Xuming Ran, Huajin Tang, Gang Pan, and Jian K Liu. Robust transcoding sensory information with neural spikes. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):1935–1946, 2021. 1

[58] B. Yin, F. Corradi, and SM Bohté. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. 2020. 2

[59] F. Zenke and S. Ganguli. Superspike: Supervised learning in multi-layer spiking neural networks. *Neural Computation*, 30(6):1514–1541, 2017. 2

[60] W. Zhang and P. Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. 2020. 7

[61] X. Zhang, Y. He, and S. Jian. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision*, 2017. 1

[62] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li. Going deeper with directly-trained larger spiking neural networks. 2020. 3, 5, 6, 7, 8

[63] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng YAN, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[64] R. Zimmer, T. Pellegrini, S. F. Singh, and T. Masquelier. Technical report: supervised training of convolutional spiking neural networks with pytorch. 2019. 2

[65] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning, 2016. 1