RecDis-SNN: Rectifying Membrane Potential Distribution for Directly Training Spiking Neural Networks

Yufei Guo[*], Xinyi Tong[*], Yuanpei Chen, Liwen Zhang, Xiaode Liu, Zhe Ma, Xuhui Huang[†]
X LAB, The Second Academy of CASIC, China
yfguo@pku.edu.cn, tongxinyi@buaa.edu.cn, starhxh@126.com

Abstract

*The brain-inspired and event-driven Spiking Neural Network (SNN) aiming at mimicking the synaptic activity of biological neurons has received increasing attention. It transmits binary spike signals between network units when the membrane potential exceeds the firing threshold. This bio-mimetic mechanism of SNN appears energy-efficiency with its power sparsity and asynchronous operations on spike events. Unfortunately, with the propagation of binary spikes, the distribution of membrane potential will shift, leading to degeneration, saturation, and gradient mismatch problems, which would be disadvantageous to the network optimization and convergence. Such undesired shifts would prevent the SNN from performing well and going deep. To tackle these problems, we attempt to **rec**tify the membrane potential **dis**tribution (MPD) by designing a novel distribution loss, MPD-Loss, which can explicitly penalize the undesired shifts without introducing any additional operations in the inference phase. Moreover, the proposed method can also mitigate the quantization error in SNNs, which is usually ignored in other works. Experimental results demonstrate that the proposed method can directly train a deeper, larger, and better-performing SNN within fewer timesteps.*

1. Introduction

Artificial Neural Networks (ANNs) have achieved huge success in many application fields, including image classification [16, 44, 45], object detection [14, 31, 39], machine translation [2], gaming [43, 46], *etc*. However, the increased computing resource required by ANNs poses a burden on latency-sensitive applications and energy-limited devices [28,30,51]. Recently, The Spiking Neural Networks (SNN) has received increasing attention and been regarded as a potential competitor of ANNs, due to their biology-inspired neural behavior and efficient computation [40]. SNNs utilize binary spike activity *i.e.*, 0 for nothing and 1 for spike event, to transmit information. This transmitting mode bridges the gap between real value-based information processing of ANNs and spike-based information processing of brains. With the characteristics of binary spike-based and sparse temporal communicating mechanisms, SNNs enjoy power-efficiency for implementation on specific neuromorphic hardwares [3, 7, 21, 33], and have been increasingly promising in neuromorphic computation [7], pattern recognition [20], robotics [18], brain-inspired devices [33], *etc*.

However, the discontinuous and non-differentiable spike activity poses difficulty to directly train SNNs using gradient descent in back-propagation. The current SNN training algorithms for avoiding the dilemma of non-differentiability can be categorized as (i) the ANN-to-SNN conversionand (ii) the surrogate gradient (SG) descent method. The conversion methods usually convert a pre-trained non-spike ANN to its SNN counterpart with the same architecture [4, 5, 9, 11, 15, 26, 38, 42]. Although the converted networks can achieve comparable performances to the original ANNs, they come at the cost of numerous inference timesteps, which are much more time and energy-consuming. The SG descent method adopts SGs (*i.e.*, $1_{|x-V_{th}|<0.5}$ or 0, otherwise, where V_{th} is the firing threshold and usually is 0.5) to replace the all-or-nothing gradients of the spike activity function [10, 17, 24, 29, 34, 41, 49]. It provides a chance to directly train SNNs with several timesteps. However, it also suffers from the problem of gradient vanishing or explosion. These problems will lead to performance degradation and shallow network architectures. Many efforts [12, 13, 23, 37, 52] have been made to solve these problems and drive the models to achieve state-of-the-art performance. Nevertheless, due to the lack of comprehensive analysis of the difficulty of training SNNs directly, there still has room for improvement in these works. Here, we provide a novel perspective to understand the dilemma of training an SNN by analyzing the membrane potential shifts. The brief analysis for a single channel is as follows.

As the binary spikes propagating through layers, the dis-

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

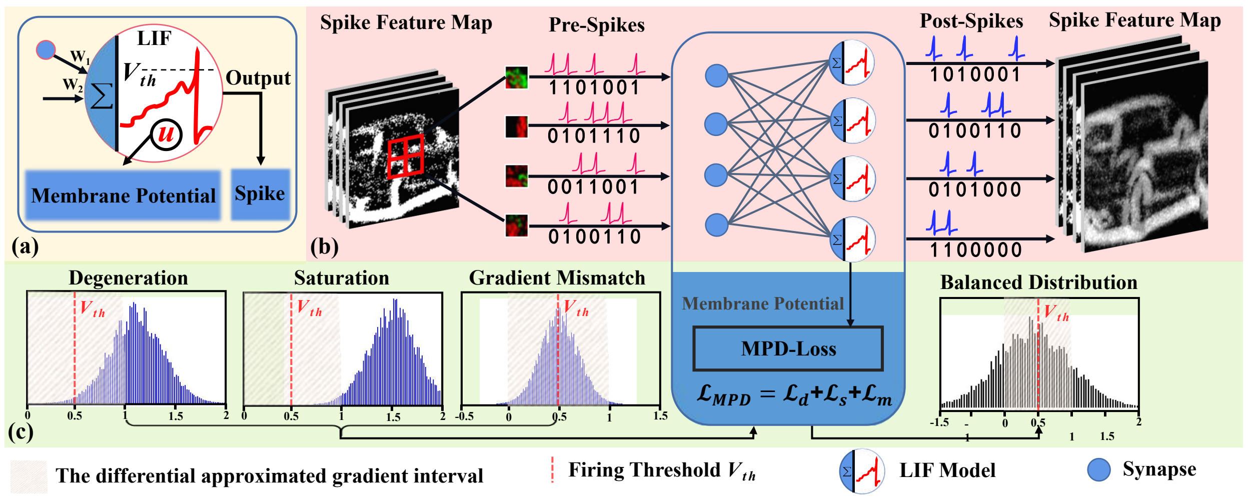

Figure 1. The overall framework of the proposed RecDis-SNN. (a) The detailed LIF model. (b) The propagation and update of the neuron spikes in two adjacent layers. (c) The examples of the three undesired membrane potential distribution shifts and the balanced distribution adjusted by the proposed MPD-loss.

tribution of membrane potential will shift accumulatively during the training and may fall into an inappropriate range, which causes training difficulties. As illustrated in Fig. 1, the membrane potential shift will appear three imbalances in some extreme cases: (i) **Degeneration**: if almost all the membrane potential values of the neurons in a channel are beyond or below the firing threshold, the spikes of this channel will be homogeneity (*i.e.*, all 0s or 1s) and the feature information of the channel will be negligible; (ii) **Saturation**: if almost all the membrane potential values of the neurons in a channel are out of the interval $[0, 1]$, the gradients for these neurons will be 0 and result in the back-propagation inability. (iii) **Gradient mismatch**: if almost all the membrane potential values in a channel fall into the interval $[0, 1]$, it is equivalent to use SGs for all gradient computation, which will enlarge the approximated errors from the accurate gradients, and therefore lead to more severe gradient mismatches. The detailed analysis is in Sec. 3.2.

The uncontrollable shift of membrane potential distribution will increase the training difficulty for SNNs, and restrict the network scale. In this paper, the three undesired membrane potential distribution (MPD) shifts will be analyzed in detail, firstly. Based on this analysis, a novel distribution loss *i.e.*, MPD-Loss is proposed to explicitly penalize those undesired shifts. By incorporating MPD-Loss into an SNN, we present RecDis-SNN, which will enjoy **Rec**tified membrane potential **Dis**tribution. The overall framework of RecDis-SNN is illustrated in Fig. 1. The main contributions are as follows:

- We present a new perspective to understand the difficulty of training SNNs by analyzing three undesired shifts of membrane potential distribution in forward propagation. Then, the MPD-Loss is proposed to penalize the undesired shifts.
- The MPD-Loss is beneficial for alleviating the gradient vanishing or explosion, adjusting the spike rate, and speeding up the convergence. In this sense, it plays a role of the normalization without extra operations in the inference phase. As far as we know, this is one of the few works that can directly train deep SNNs without normalization techniques or any warm-start.
- The MPD-Loss can also mitigate the problem of quantization error, which is usually ignored in other works. To our best knowledge, this is the first work that has noticed the problem in SNNs and provides a solution.
- The RecDis-SNN is evaluated on both standard non-spiking and neuromorphic benchmarks. Experimental results show that by using MPD-Loss, the RecDis-SNNs can achieve state-of-the-art performance. Meanwhile, MPD-Loss can be easily combined with other methods and further improve their performance.

2. Related Work

The unwanted MPD shifts will aggravate the instability of gradients (*i.e.*, gradient vanishing or explosion) in deep SNNs. Moreover, those shifts can cause over-firing or under-firing of the SNN. The approaches for solving these problems in previous works can be mainly categorized into (i) normalization techniques and (ii) extensions with more learnable parameters. These approaches alleviate the MPD shift problems to some extent in an implicit way. We introduce those related works in this section.

Normalization. Normalization techniques are commonly used to avoid gradient vanishing or explosion by shrinking the internal variance shift in Deep Neural Networks (DNNs). With the verified advantages of increasing accuracy and accelerating convergence in DNNs, many variants had been presented, *e.g*., batch normalization [19], group normalization [48], layer normalization [1], and weight normalization [27]. There are also some other normalization-like techniques adopting empirical settings to redistribute data [35, 36]. For example, [35] presented the similar degeneration problem of binarization and solved it using a specially designed additional regularization loss. We are inspired by this idea and apply it in SNN field. We further reduce the quantization error through improvement. Inspired by the huge success in DNNs, many improved normalization techniques had also been presented for training SNNs [11, 42, 47, 52]. The NeuNorm [47] was proposed to balance neural selectivity and normalize the neuron activity by using the input statistics (moving average firing rate). Theoretically, NeuNorm is similar with the batch normalization. By normalizing the spatio-temporal pre-activations to $\mathcal{N}(\mu, V_{th}^2)$, the threshold-dependent batch normalization (tdBN) [52] can weaken the dependence on the firing threshold in pre-activations. It is clear that normalization restricts the neuron pre-activation range, which could be beneficial to overcome the degeneration and saturation problems caused by the undesired shifts. However, the existing normalization methods in SNNs all ignore the gradient mismatch problem to our best knowledge.

Extension with more learnable parameters. To adjust the neuron firing, plenty of works attempted to use more learnable parameters to balance the firing threshold and membrane potential in neurons. For example, the parametric Leaky Integrate-and-Fire (PLIF) neuron [13] was proposed as an alternative for the original neuron, in which, the membrane leak is a learnable parameter rather than an empirical hyper-parameter. Furthermore, the Diet-SNN [37] is presented to jointly optimize the membrane leak and the firing threshold. Nevertheless, a better relationship between membrane potentials and firing threshold can be explored by introducing more learnable parameters, this kind of method is still plagued by the gradient mismatch problem.

Different from the current SNN methods, the proposed method simultaneously addresses the three undesired MPD shift problems without losing sight of the quantization error problem. Like [6, 35], we try to carve the distribution by embedding the regularization in the loss function, *i.e*., MPD-Loss. Furthermore, it can also alleviate the quantization error, which is usually ignored in other works. We suppose that this new insight on SNNs including the quantization error problems may bring some interesting enlightenment to the following-up research.

3. Materials and Methodology

This section first theoretically introduces the Leaky-Integrate-and-Fire (LIF) neuron model and the SGs utilized in the proposed approach. Then the proposed method of rectifying MPD will be introduced in detail. Finally, its effectiveness and performance are demonstrated in an analytic way.

3.1. Spiking Neural Networks

Leaky-Integrate-and-Fire Neuron Model. The spiking neuron is the fundamental computing unit of SNNs. Its membrane potential and internal voltage change response to the input signals through the connected synapses. Unlike ANNs, which process information with a series of continuous and high-precision values, the spiking neuron propagates information by emitting a binary spike signal and transmitting it to the next connected one when membrane potential exceeds the firing threshold, following the principle of biological neuron firing rules. As a commonly used neuron model in SNNs, the Leaky-Integrate-and-Fire (LIF) model is adopted in this paper. It can be described as the following differential equations:

$$\tau \frac{du}{dt} = -u + RI, u < V_{th} \tag{1}$$

$$u = u_{reset} \quad \& \quad fire \quad a \quad spike, u \geq V_{th} \tag{2}$$

where u is the neuron membrane potential, u_{reset} denotes the initial neuron resting potential, which is usually assumed as 0, the product of the input current I and resistance R denotes the changing voltage towards the pre-synaptic signals, τ is a time constant, and V_{th} is the firing threshold. Equation (1) describes the membrane potential accumulation below V_{th}. Equation (2) shows that when u is up to V_{th}, the neuron fires a spike and the u is reset to the resting potential, u_{reset}.

Note that the differential representation with continuously varying voltages is not easy to be implemented in mainstream machine learning frameworks. Hence, an iterative model [47] is conducted as follows,

$$u^t = (1 - \frac{dt}{\tau})u^{t-1} + \frac{dt}{\tau}RI \tag{3}$$

where u^t is the membrane potential of the neuron at timestep t. The factor $(1 - \frac{dt}{\tau})$ can be denoted as λ_τ, the input voltage $\frac{dt}{\tau}RI$ can be expanded to the weighted summation of pre-synaptic signals as $\sum_j w_j o_j^t$, where the weight w_j is used to connect the j-th pre-neuron and current neuron, and o_j^t indicates the binary spike from the j-th pre-neuron at timestep t. Then the iterative LIF model is updated as

$$u^t = \lambda_\tau u^{t-1}(1 - \hat{o}^{t-1}) + \sum_j w_j o_j^t \tag{4}$$

$$\hat{o}^{t-1} = \begin{cases} 1, & if\ u^{t-1} \geq V_{th}, \\ 0, & otherwise. \end{cases} \tag{5}$$

Equation (4) indicates that the membrane potential of the current neuron, u^t is affected by the spike o_j^t from the pre-neuron. And the spike emission, $\hat{o}^{t-1}$ from the previous timestep also contributes to u^t in current timestep t.

In this paper, we set the firing threshold V_{th} to 0.5, the leaky factor τ to 0.2, and the resting potential u_{reset} to 0. Since the LIF model in the last layer would damage the network performance severely, we replace the LIF model in the output layer with a modified model that only accumulates the incoming inputs without using any leakage or emitting spike following recent works [13, 37, 47, 52]. This modified model can be described by

$$u^t = u^{t-1} + \sum_j w_j o_j^t \tag{6}$$

Surrogate Gradients in Back-Propagation. As aforementioned, using SGs to replace the all-or-nothing gradients of the spike activity function is one of the most commonly used SNN training methods. In this paper, the rectangular function is adopted as in the prior works [47, 52]. It can be denoted as

$$\frac{do}{du} = \begin{cases} 1, & if\ 0 < u < 1, \\ 0, & otherwise. \end{cases} \tag{7}$$

3.2. Rectifying Membrane Potential Distribution

To explain the proposed method clearly, we first introduce some notations and give formal definitions of the unwanted distribution shifts during the training process. Then, the distribution loss, MPD-Loss for explicitly penalizing the shifts is elaborated in detail. Next, how MPD-Loss is incorporated into the objective function is given. Finally, a deeper understanding of the proposed method is presented by the experiment and analysis.

We use $\mathbf{U}^{b,l,c}$ to denote the membrane potentials for all timesteps in the c-th channel of the l-th layer for the b-th batch of data. Thus, $\mathbf{U}^{b,l,c}$ is a 4-D tensor with a size of $B \times T \times W \times H$, where B, T, W and H denote the batch size, the number of timesteps, width and height of the membrane potential map, respectively. For brevity, we simplify $\mathbf{U}^{b,l,c}$ as $\mathbf{U}$. And $U_{(q)}$ denotes the q quantile of the ascending ordered elements of $\mathbf{U}$, where $0 \leq q \leq 1$. Then the three undesired distribution shifts shown in Fig. 1 can be defined as follows,

下分位数

最小值 最大值

- **Degeneration**: $U_{(0)} \geq V_{th}$ or $U_{(1)} \leq V_{th}$
- **Saturation**: $U_{(0)} \geq 1$ or $U_{(1)} \leq 0$
- **Gradient mismatch**: $0 \leq U_{(0)} \leq 1$ and $0 \leq U_{(1)} \leq 1$

We will show how to design the distribution loss for penalizing the three undesired shifts and explain the reasons for such design, as below.

Degeneration. In the forward-propagation, if $U_{(0)} \geq V_{th}$ or $U_{(1)} \leq V_{th}$, the spike rate of neurons will be 1 or 0. To avoid this situation, the loss function can be written as

$$\mathcal{L}_D = \left[(U_{(0)} - V_{th})_+\right]^2 + \left[(V_{th} - U_{(1)})_+\right]^2 \tag{8}$$

where $(\cdot)_+$ is equivalent to $\max\{\cdot, 0\}$.

$\mathcal{L}_D$ is only related to the maximum and minimum of U, and it is an isolated value. Hence it will cause limited back-propagation. To enhance the robustness of $\mathcal{L}_D$, the punishment can be relaxed by a pair of quantiles, ϵ and $1-\epsilon$. Then $\mathcal{L}_D$ can be rewritten as

分位数

$$\mathcal{L}_D = \left[(U_{(\epsilon)} - V_{th})_+\right]^2 + \left[(V_{th} - U_{(1-\epsilon)})_+\right]^2 \tag{9}$$

To further make $\mathcal{L}_D$ fit in the back-propagation phase, it should be transformed into a differentiable form. In such case, according to the experience in [52], we assume that the values of U can be sampled from a Gaussian distribution, $\mathcal{N}(\mu, \sigma^2)$, where μ and σ can be estimated by the mean and standard deviation over the U. Thus, the quantile ϵ can be formulated as $\mu - k_\epsilon \sigma$ following the predefined Gaussian distribution, where k_ϵ is constantly determined by ϵ. Then, $\mathcal{L}_D$ can be further updated as

假设

$$\begin{aligned} \mathcal{L}_D &= [(\mu - k_\epsilon\sigma - V_{th})_+]^2 + [(V_{th} - (\mu + k_\epsilon\sigma))_+]^2 \\ &= [(\mu - k_\epsilon\sigma - V_{th})_+]^2 + [(V_{th} - \mu - k_\epsilon\sigma)_+]^2 \end{aligned} \tag{10}$$

Saturation. This case occurs when most of the membrane potentials are out of the interval $[0, 1]$, so it can be penalized by

$$\mathcal{L}_S = \left[(U_{(0)} - 1)_+\right]^2 + \left[0 - U_{(1)})_+\right]^2 \tag{11}$$

Similar to $\mathcal{L}_D$, it can also be relaxed by the pair of quantiles, ϵ and $1 - \epsilon$ as follows

$$\mathcal{L}_S = \left[(U_{(\epsilon)} - 1)_+\right]^2 + \left[(0 - U_{(1-\epsilon)})_+\right]^2. \tag{12}$$

Similar to $\mathcal{L}_D$, $\mathcal{L}_S$ can be formulated into a differentiable form by introducing the Gaussian distribution estimation, as follows

$$\begin{aligned} \mathcal{L}_S &= [(\mu - k_\epsilon\sigma - 1)_+]^2 + [0 - (\mu + k_\epsilon\sigma)_+]^2 \\ &= [(\mu - k_\epsilon\sigma - 1)_+]^2 + [-\mu - k_\epsilon\sigma)_+]^2. \end{aligned} \tag{13}$$

Gradient Mismatch. In the back-propagation, a rectangular function is utilized to approximate the gradient of the spike activity function with the interval of $[0, 1]$. Hence, the more membrane potentials are in the range of $[0, 1]$, the worse the gradient mismatch problem will be. Therefore, the loss function can be designed as

$$\mathcal{L}_M = \left[\min(U_{(0)} - 0, 1 - U_{(1)})_+\right]^2, \tag{14}$$

Algorithm 1 Training SNNs for classification with MPD-Loss for one epoch.

Input: an SNN to be trained; Timestep: T; Firing threshold: V_{th}; Training dataset; Validation dataset; Total training iteration in one epoch I_{trian}; Total validation iteration in one epoch: I_{val}.
Output: The trained SNN.

```
 1: for all $i = 1, 2, \dots, I_{train}$ iteration do
 2:     Get mini-batch training data and class label: $\mathbf{Y}^i$;
 3:     Calculate the SNN output $\mathbf{O}^i(t)$ of each time step;
 4:     Compute classification loss $L_{CE} = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}_{CE}(\mathbf{O}^i(t), \mathbf{Y}^i)$;
 5:     for all $l = 1, 2, \dots, L$ layers do
 6:         Compute $\mathcal{L}_D$, $\mathcal{L}_S$, and $\mathcal{L}_M$ for every layer $l$.
 7:         Compute MPD-Loss $\mathcal{L}_{MPD}$ by Eq. (19).
 8:     end for
 9:     Compute the total loss $\mathcal{L}_{CE\text{-}MPD}$ by Eq. (20);
10:     Backpropagation and update model parameters.
11: end for
12: for all $i = 1, 2, \dots, I_{val}$ iteration do
13:     Get mini-batch training data and class label: $\mathbf{Y}^i$;
14:     Calculate the SNN average output $\mathbf{O}^i_{mean} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{O}^i(t)$ over all timestep;
15:     Compare the classfication factor $\mathbf{O}^i_{mean}$ and $\mathbf{Y}^i$ for classification.
16: end for
```

or

$$\mathcal{L}_M = \left[\max(U_{(0)} - 0, 1 - U_{(1)})_+\right]^2. \tag{15}$$

Obviously, Eq. (14) tends to bias the potential distribution to 0 or 1, while Eq. (15) tends to pull away $U_{(0)}$ and $U_{(1)}$, so that the distribution will expand out of the interval $[0, 1]$. Though the two both can mitigate the problem of gradient mismatch, Eq. (14) will cause over-firing or under-firing of the SNN. In addition, Eq. (15) can result in a bimodal potential distribution, while Eq. (14) can not in practice. Since a bimodal potential distribution enjoys quantization error reduction (see details in the next subsection), we choose Eq. (15) here. Then, the loss function can be relaxed as

$$\mathcal{L}_M = \left[\max(U_{(\epsilon)} - 0, 1 - U_{(1-\epsilon)})_+\right]^2. \tag{16}$$

Similarly, we have

$$\begin{aligned}\mathcal{L}_M &= [\max(\mu - k_\epsilon\sigma - 0, 1 - (\mu + k_\epsilon\sigma))_+]^2 \\ &= [\max(\mu - k_\epsilon\sigma, 1 - \mu - k_\epsilon\sigma)_+]^2.\end{aligned} \tag{17}$$

Then, the proposed distribution loss, MPD-Loss, which aims at addressing the three challenges can be defined as the integration of the above loss functions as follows,

$$\mathcal{L}_{MPD} = \mathcal{L}_D + \mathcal{L}_S + \mathcal{L}_M. \tag{18}$$

In practice, except for the last output layer, the undesired shifts are penalized in every channel for each layer in the training phase. Specifically, L_{MPD} for the b-th batch of input data is determined as

$$\mathcal{L}^b_{MPD} = \frac{1}{l}\sum_l\left(\frac{1}{c}\sum_c \mathcal{L}_D^{b,l,c} + \mathcal{L}_S^{b,l,c} + \mathcal{L}_M^{b,l,c}\right). \tag{19}$$

Finally, taking classification loss into consideration, the total loss can be written as

$$\mathcal{L}^b_{CE\text{-}MPD} = \mathcal{L}^b_{CE} + \lambda\mathcal{L}^b_{MPD} \tag{20}$$

where $\mathcal{L}_{CE}$ is the cross-entropy loss, and λ is a balanced coefficient, which is set as 2 in the paper.

For implementation, we only replace $\mathcal{L}_{CE}$ with $\mathcal{L}_{CE\text{-}MPD}$ in the training process, and keeps the inference rules of SNNs in the evaluation phase. To penalize the degeneration, saturation, and gradient mismatch, the corresponding parameter k_ϵ are set to 1, 0.25, and 1.25, respectively. The training algorithm for one epoch is detailed in Algorithm 1.

3.3. Analysis and Discusion

In contrast with the prior works for alleviating the undesired shifts implicitly using normalization techniques, we propose to rectify the membrane potential distribution by the loss, so that those undesired shifts can be alleviated in an explicit way. Besides, the proposed method has another three advantages compared to normalization: (i) the SNN trained with normalization will inevitably introduce additional operations in the inference phase than its counterpart that trained without normalization, while the SNN trained with MPD-Loss will not; (ii) MPD-Loss can address the gradient mismatch problem; (iii) MPD-Loss helps reduce quantization error, *i.e.*, the information loss in forward propagation.

To verify the advantage of MPD-Loss in terms of the gradient mismatch problem, we investigate the membrane potential distribution change of the last layer of the first block in the ResNet-19 model trained with tdBN (threshold-dependent batch normalization) [52] and MPD-Loss, respectively. As shown in Fig. 2, MPD-Loss gradually changes an offset-free unimodal distribution into a unimodal distribution with offset, and finally into a bimodal distribution. However, for the tdBN-based model, the distribution is eventually turned into a unimodal distribution with offset. Both methods tend to maintain the MPD within the interval of $[-1, 2]$, hence they are all able to address the degeneration and saturation problems well. However, compared with MPD-Loss-based SNN, the SNN trained with tdBN has more membrane potential values in the interval $[0, 1]$. This difference indicates that without an explicit penalizing mechanism for such the shift like MPD-Loss, tdBN would not alleviate gradient mismatch well.

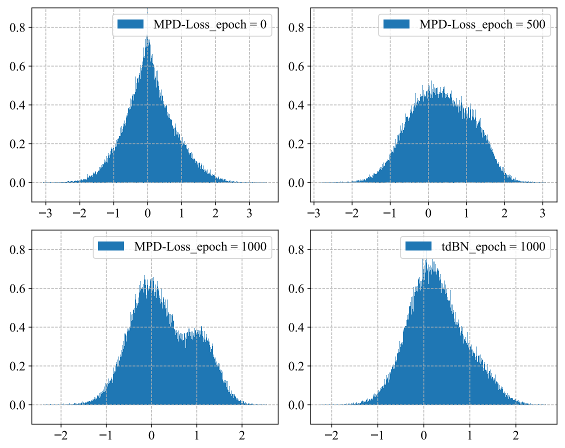

Figure 2. The membrane potential distribution of the specific layer in ResNet-19 trained with MPD-Loss in epoch 0, 500, and 1000, as well as the one trained with tdBN normalization in epoch 1000.

Moreover, the bimodal MPD also helps SNNs reduce quantization error, which is introduced by the transformation from real values into $0/1$ spikes. In specific, the firing threshold divides the membrane potentials into two parts, the part with smaller values is assigned to ”0” spike, and the one with larger values is assigned to ”1” spike. Then the quantization error depends on the margin between the membrane potential and its corresponding spike. Hence, the quantization error can be defined as the square of the difference between the membrane potential and its corresponding quantization spike value. It can be seen that, compared to a unimodal distribution, a bimodal distribution will be more friendly to this mechanism, since the latter one can naturally gather the membrane potentials near ”0” and ”1”, but the former one squeeze those potentials close to some value in $[0,\ 1]$. Therefore, constraining the membrane potentials into a bimodal distribution is more likely to obtain an SNN with less quantization error, and it also takes care of the information expressivity of the network.

4. Experiment

We evaluated the proposed training framework on the standard non-spiking datasets, CIFAR10/100 [22] and ImageNet [8], as well as the neuromorphic dataset, DVS-CIFAR10 [25]. The network architectures in this paper include CIFARNet [47], ResNet-19 [52], Spiking-ResNet-34 [52], and VGG-16 [37].

4.1. Ablation Study for MPD-Loss

We first conducted a set of ablation experiments to verify the effectiveness of the three proposed distribution losses on CIFAR10 using CIFARNet and ResNet-19 as backbones with timestep = 4. The results are shown in Tab. 1.

Table 1. Ablation study for MPD-Loss.

Architecture	Method	Accuracy
CIFARNet	None	89.83%
	w/ $\mathcal{L}_D$	91.29%
	w/ $\mathcal{L}_S$	91.01%
	w/ $\mathcal{L}_M$	91.72%
	w/ MPD-Loss	92.08%
ResNet-19	None	91.23%
	w/ $\mathcal{L}_D$	93.84%
	w/ $\mathcal{L}_S$	93.83%
	w/ $\mathcal{L}_M$	94.04%
	w/ MPD-Loss	94.34%

Benefitting from the 0/1 spike information processing paradigm, SNNs run efficiently on neuromorphic hardwares. However, this information processing paradigm also causes SNN models not easy to be trained, since the spike rate of an SNN model with random weights always decreases layer by layer. The spike rate will soon become 0 after several layers when an SNN is trained from scratch, resulting in information flow disappearing and difficulty to train large and deep SNNs. As shown in Tab. 1, the models initialized by random weights without any training techniques were stuck in dilemma, and the accuracy was nearly the same after hundreds of epochs (details can be seen in Fig. 3). After applying any proposed distribution loss, the trapped networks all can jump out of the dilemma. Our method shows the potential to train deep SNNs. To further compare the accuracy, we use the model trained by the MDP-Loss after several iterations when the spike rate of its last layer is not 0 as the initialization of the vanilla SNNs.

The CIFARNet based on $\mathcal{L}_D$, $\mathcal{L}_S$, $\mathcal{L}_M$ achieve 91.29%, 91.01%, 91.72% accuracy, surpassing the vanilla one by 1.46%, 1.18%, 1.89% respectively. For ResNet-19, The three proposed distribution losses also demonstrate their superiority. These distribution losses are all effective for training deep SNNs and improving their performance. Moreover, it can be seen that the improvements brought by these distribution losses together can be superimposed.

4.2. Ablation Study for Training Cost

We then conducted some experiments to explore the added training cost by MPD-Loss. The results of training cost experiments for CIFAR10 using CIFARNet and ResNet-19 as backbones with 4 timesteps on a single RTX 2080Ti in the Tab. 2. We set batchsize and epoch as 64 and 1000. It can be seen that the extra training cost do not exceed 17% for MPD-Loss. In the inference, our method will not add extra time since MPD-Loss will not be computed in this phase.

Table 2. Ablation study for training cost.

Architecture	Method	Time cost (min.)
CIFARNet	None	2524.82
	w/ tdBN	2753.65
	w/ MPD-Loss	2948.88
ResNet-19	None	7685.88
	w/ tdBN	8099.43
	w/ MPD-Loss	8648.85

Table 3. The result of quantization error.

Dataset	Method	Ave. error
CIFAR10	w/ MPD-Loss	0.66
	w/ tdBN	0.78
CIFAR100	w/ MPD-Loss	0.64
	w/ tdBN	0.77
CIFAR10-DVS	w/ MPD-Loss	0.61
	w/ tdBN	0.83

4.3. Study for Quantization Error Reduction

To verify the advantages of the MPD-Loss in reducing quantization error, we computed the average quantization error of the last layer of the first block in the ResNet-19 with 4 timesteps trained by tdBN and the MPD-loss on CIFAR10, CIFAR100, and CIFAR10-DVS respectively. The result is reported in the Tab. 3. It shows that the average quantization error of tdBN is much bigger than that of the MPD-Loss.

4.4. Comparison with The Normalization

We further conducted experiments to compare the proposed MPD-Loss with tdBN on CIFAR10 using CIFARNet and ResNet-19 with timestep = 4. The comparison results of the SNNs trained without any techniques as well as the one trained with tdBN only, MPD-Loss only, and the combination of tdBN and MPD-Loss are shown in Tab. 4.

As abovementioned, the models with random initialization maybe not be trainable. Tab. 4 shows that after applying tdBN or MPD-Loss, these all can be trained then. However, it can be seen in Fig. 3, that our method enjoys easier convergence than tdBN technique. Furthermore, the tdBN needs more epochs to get out of the trouble. The MPD-Loss based on CIFARNet achieves 92.08% accuracy, surpassing the tdBN by 1.39%. For ResNet-19, The proposed framework can reach 94.34%, outperforming the tdBN technique (92.92%) with 1.42% improvement. It can be claimed that the regularization conducted by the proposed method has a similar positive effect as normalization techniques in the SNN training, or even better.

It can also be seen that after combining the MPD-loss and the tdBN, the SNNs can show higher performance, reaching the top-1 accuracy of 92.20% on CifarNet and 95.53% on ResNet-19. It proves that the MPD-loss is compatible with other normalization techniques and can be used in combination with them.

Table 4. Comparison with the normalization.

Architecture	Method	Accuracy
CIFARNet	None	-
	w/ tdBN	90.69%
	w/ MPD-Loss	92.08%
	w/ tdBN & MPD-Loss	**92.20%**
ResNet-19	None	-
	w/ tdBN	92.92%
	w/ MPD-Loss	94.34%
	w/ tdBN & MPD-Loss	**95.53%**

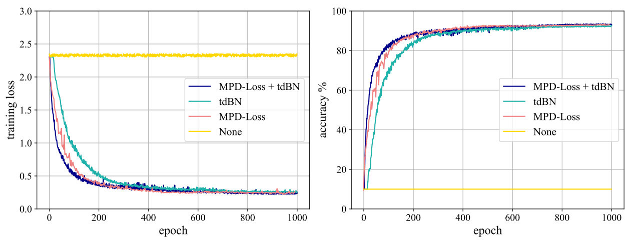

Figure 3. Comparisons of the training loss (left) and test accuracy (right) on CIFAR10.

4.5. Comparisons with Other Methods

In this section, we compared our experimental results with previous works. The experimental results of the classification accuracy on non-spiking datasets and neuromorphic datasets are listed in Tab. 5 and Tab. 6, respectively. Since the tdBN [52] can improve the performance of the proposed RecDis-SNN, we combined it with our MPD-Loss. For each run, we report the mean accuracy as well as the standard deviation with 3 trials.

For CIFAR10, the RecDis-SNN based on ResNet-19 [52] is superior to all the other compared methods by achieving the top-1 accuracy of 95.55% with only 6 timesteps. Especially, our RecDis-SNN significantly outperforms the STBP-tdBN [52] with 2.39% improvement, in which the network architecture is consistent with ours. The CIFARNet [47] trained with the proposed method is also competitive among the other state-of-the-art approaches, reaching 92.20% with 4 timesteps and 92.91% with 6 timesteps.

On CIFAR100, the top-1 accuracy of the RecDis-SNN using ResNet-19 as backbone gets to 74.10% with only 4

Table 5. Comparisons with state-of-the-art SNNs on CIFAR10, CIFAR100, and ImageNet.

Dataset	Method	Type	Architecture	Timestep	Accuracy
CIFAR10	RMP-SNN [15]	ANN2SNN	ResNet-20	2048	91.36%
	ANN-SNN [9]	ANN2SNN	CIFARNet	128	90.58%
	Hybrid Training [38]	Hybrid training	ResNet-20	250	92.22%
	STBP [47]	SNN training	CIFARNet	12	90.53%
	Spike-basedBP [23]	SNN training	ResNet-11	100	90.95%
	STBP-tdBN [52]	SNN training	ResNet-19	6	93.16%
	PLIF [13]	SNN training	PLIFNet	8	93.50%
	TSSL-BP [50]	SNN training	CIFARNet	5	91.41%
	Diet-SNN [37]	SNN training	CIFARNet	5	91.59%
	RecDis-SNN (**ours**)	SNN training	CIFARNet	2	90.58% ±0.12
				4	92.20% ±0.10
				6	92.91% ±0.09
			ResNet-19	2	93.64% ±0.07
				4	95.53% ±0.05
				6	**95.55%** ±0.05
CIFAR100	RMP-SNN [15]	ANN2SNN	ResNet-20	2048	67.82%
	ANN-SNN [9]	ANN2SNN	VGG-16	128	70.47%
	BinarySNN [32]	BNN2SNN	VGG-15	62	63.20%
	Hybrid Training [38]	Hybrid training	VGG-11	125	67.78%
	Diet-SNN [37]	SNN training	VGG-16	5	69.67%
	RecDis-SNN (**ours**)	SNN training	VGG-16	5	69.88% ±0.08
			ResNet-19	4	**74.10%** ±0.13
ImageNet	RMP-SNN [15]	ANN2SNN	ResNet-34	1024	66.61%
	Hybrid Training [38]	Hybrid training	ResNet-34	250	61.48%
	STBP-tdBN [52]	SNN training	ResNet-34	6	63.72%
	PLIF [13]	SNN training	ResNet-34	7	67.04%
	RecDis-SNN (**ours**)	SNN training	ResNet-34	6	**67.33%** ±0.10

timesteps. It is also 0.21% higher than Diet-SNN [37] based on the VGG-16 with 5 timesteps.

For ImageNet, the standard ResNet-34 [16] was selected as the backbone for the SNNs. Our RecDis-SNN achieves 67.33% top-1 accuracy with 6 timesteps, it significantly outperforms the STBP-tdBN [52] with 3.61% improvement. In addition, the reported accuracy is also higher than PLIF [13] but with fewer parameters and timesteps.

Table 6. Performance comparisons on DVS-CIFAR10.

Methods	Architecture	Accuracy
STBP [47]	CIFARNet	60.50%
STBP-tdBN [52]	ResNet-19	67.80%
RecDis-SNN (**ours**)	CIFARNet	**67.30%** ±0.05
	ResNet-19	**72.42%** ±0.06

For DVS-CIFAR10, our CIFARNet and ResNet-19 achieve 67.30% and 72.42% top-1 accuracy with 10 timesteps, respectively, with 6.80% and 4.62% absolute accuracy improvements compared with the other state-of-the-art methods with the same backbones and the timesteps.

5. Conclusion

In this paper, we present a new perspective to understand the difficulty of training SNNs. With the propagation of spikes, the distribution of membrane potential will shift, causing degeneration, saturation, and gradient mismatch problems. We then introduce MPD-Loss to penalize the undesired shifts. Experimental results and analysis show that the proposed distribution loss performs better than other normalization techniques by regularizing the distribution in many aspects. Our method can also help reduce the gradient mismatch and quantization error of SNNs, which are usually ignored in other prior works. Based on these advantages, the MPD-Loss-based SNN can outperform the current methods on both typical non-spiking and neuromorphic datasets with fewer timesteps. The proposed method has the great potential to expand the scale and depth of the SNN and is also flexible to be combined with other advanced techniques to further improve the performance.

References

[1] J. Ba, J. Kiros, and G. Hinton. Layer normalization. *arXiv*, 07 2016. 3

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv*, 2014. 1

[3] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith. Benchmarking keyword spotting efficiency on neuromorphic hardware. *arXiv*, 2018. 1

[4] R. Bodo, L. Iulia-Alexandra, Y. Hu, P. Michael, and S. C. Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11:682, 2017. 1

[5] Y. Cao, Y. Chen, and D. Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015. 1

[6] S. Darabi, M. Belbahri, M. Courbariaux, and V. P. Nia. Regularized binary network training. *arXiv*, 2018. 3

[7] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, H. Wang, and et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. 1

[8] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: a large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 06 2009. 6

[9] S. Deng and S. Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv*, 02 2021. 1, 8

[10] S. Deng, Y. Li, S. Zhang, and S. Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv*, 2022. 1

[11] P. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 07 2015. 1, 3

[12] W. Fang, Z. Yu, Y. Chen, T. Huang, Timothée M., and Y. Tian. Deep residual learning in spiking neural networks. *arXiv*, 2021. 1

[13] W. Fang, Z. yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. *arXiv*, 08 2021. 1, 3, 4, 8

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 580–587, 2014. 1

[15] B. Han, G. Srinivasan, and K. Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13555–13564, 2020. 1, 8

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 06 2016. 1, 8

[17] D. Huh and T. Sejnowski. Gradient descent for spiking neural networks. *arXiv*, 06 2017. 1

[18] T. Hwu, J. Isbell, N. Oros, and J. Krichmar. A self-driving robot using deep convolutional neural networks on neuromorphic hardware. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 635–641, 05 2017. 1

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *2015 Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 02 2015. 3

[20] Y. Jin, P. Li, and W. Zhang. Hybrid macro/micro level backpropagation for training deep spiking neural networks. In *2018 Proceedings of the 32nd International Conference on Neural Information Processing(NIPS)*, page 7005–7015, 09 2018. 1

[21] M. Khan, D. Lester, L. Plana, A. Rast, X. Jin, E. Painkras, and S. B. Furber. Spinnaker: Mapping neural networks onto a massively-parallel chip multiprocessor. In *2008 International Joint Conference on Neural Networks (IJCNN)*, pages 2849–2856, 2008. 1

[22] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 (canadian institute for advanced research). 2010. 6

[23] C. Lee, S. Sarwar, P. Panda, G. Srinivasan, and K. Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14:119, 02 2020. 1, 8

[24] J. H. Lee, T. Delbruck, and M. Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10(508):1662–4548, 2016. 1

[25] H. Li, H. Liu, X. Ji, G. Li, and L. Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11:309, 2017. 6

[26] Y. Li, S. Deng, X. Dong, R. Gong, and S. Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *2021 Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 6316–6325. PMLR, 18–24 Jul 2021. 1

[27] Y. Li, X. Dong, and W. Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. *arXiv*, 2019. 3

[28] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv*, 2021. 1

[29] Y Li, Y Guo, S Zhang, S Deng, Y Hai, and S Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *2021 Advances in Neural Information Processing Systems (NIPS)*, 34, 2021. 1

[30] Y. Li, F. Zhu, R. Gong, M. Shen, X. Dong, F. Yu, S. Lu, and S. Gu. Mixmix: All you need for data-free compression are feature and data mixing. In *2021 IEEE International Conference on Computer Vision (ICCV)*, pages 4410–4419, 2021. 1

[31] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *2016 European Conference on Computer Vision(ECCV)*, pages 21–37, 2016. 1

[32] S. Lu and A. Sengupta. Exploring the connection between binary and spiking neural networks. *Frontiers in Neuroscience*, 14:535, 2020. 8

[33] P. Merolla, J. Arthur, R. Alvarez-Icaza, A. Cassidy, J. Sawada, F. Akopyan, and et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science (New York, N.Y.)*, 345(6197):668–673, 08 2014. 1

[34] 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)*, pages 1–4, 2017. 1

[35] Ding R., Chin T., Liu Z., and Marculescu D. Regularizing activation distribution for training binarized deep networks, 2019. 3

[36] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *2016 European Conference on Computer Vision (ECCV)*, pages 525–542, 2016. 3

[37] N. Rathi and K. Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv*, 08 2020. 1, 3, 4, 6, 8

[38] N. Rathi, G. Srinivasan, P. Panda, and K. Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv*, 05 2020. 1, 8

[39] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1

[40] K. Roy, A. Jaiswal, and P. Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019. 1

[41] A. Samadi, T. P. Lillicrap, and D. B. Tweed. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural Computation*, 29(3):578–602, 2017. 1

[42] 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. 1, 3

[43] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, and et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016. 1

[44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv*, 2014. 1

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, and et al.. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 1

[46] M. Volodymyr, K. Koray, S. David, A. A. Rusu, V. Joel, M. G. Bellemare, G. Alex, R. Martin, A. K. Fidjeland, O. Georg, and et al. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. 1

[47] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L.P. Shi. Direct training for spiking neural networks: Faster, larger, better. *2019 Proceedings of the AAAI Conference on Artificial Intelligence(AAAI)*, 33:1311–1318, 07 2019. 3, 4, 6, 7, 8

[48] Y. Wu and K. He. Group normalization. *International Journal of Computer Vision*, 128:742–755, 03 2020. 3

[49] Y. Wu, D. Lei, G. Li, J. Zhu, and L. Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:331, 2018. 1

[50] W.i Zhang and P. Li. Spike-train level backpropagation for training deep recurrent spiking neural networks. In *2019 Proceedings of the International Conference on Neural Information Processing(NIPS)*, 08 2019. 8

[51] X. Zhang, H. Qin, Y. Ding, R. Gong, Q. Yan, R. Tao, Y. Li, F. Yu, and X. Liu. Diversifying sample generation for accurate data-free quantization. In *2021 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15658–15667, 2021. 1

[52] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li. Going deeper with directly-trained larger spiking neural networks. *arXiv*, 10 2020. 1, 3, 4, 5, 6, 7, 8