sensors MDPI

Article

Rethinking the Role of Normalization and Residual Blocks for Spiking Neural Networks

Shin-ichi Ikegawa [1,*], Ryuji Saiin [2], Yoshihide Sawada [1] and Naotake Natori [1]

[1] Tokyo Research Center, Aisin Corporation, Akihabara Daibiru 7F 1-18-13, Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan; yoshihide.sawada@aisin.co.jp (Y.S.); naotake.natori@aisin.co.jp (N.N.)
[2] AISIN SOFTWARE Co., Ltd., Advance Square Kariya 7F 1-1-1, Aioicho, Kariya 448-0027, Aichi, Japan; ryuji.saiin@aisin-software.co.jp
* Correspondence: shinichi.ikegawa@aisin.co.jp

Abstract: Biologically inspired spiking neural networks (SNNs) are widely used to realize ultralow-power energy consumption. However, deep SNNs are not easy to train due to the excessive firing of spiking neurons in the hidden layers. To tackle this problem, we propose a novel but simple normalization technique called postsynaptic potential normalization. This normalization removes the subtraction term from the standard normalization and uses the second raw moment instead of the variance as the division term. The spike firing can be controlled, enabling the training to proceed appropriately, by conducting this simple normalization to the postsynaptic potential. The experimental results show that SNNs with our normalization outperformed other models using other normalizations. Furthermore, through the pre-activation residual blocks, the proposed model can train with more than 100 layers without other special techniques dedicated to SNNs.

Keywords: spiking neural networks; normalization; pre-activation residual blocks

check for updates

Citation: Ikegawa, S.-i.; Saiin, R.; Sawada, Y.; Natori, N. Rethinking the Role of Normalization and Residual Blocks for Spiking Neural Networks. *Sensors* **2022**, *22*, 2876. https://doi.org/10.3390/s22082876

Academic Editor: Ioannis E. Livieris

Received: 1 March 2022
Accepted: 5 April 2022
Published: 8 April 2022

1. Introduction

Recently, spiking neural networks (SNNs) [1] have attracted substantial attention due to ultra-low power consumption and high friendliness with hardware such as neuromorphic-chips [2,3] and field-programmable gate array (FPGA) [4]. In addition, SNNs are biologically more plausible than artificial neural networks (ANNs) because their neurons communicate with each other through spatio-temporal binary events (spike trains), similar to biological neural networks (BNNs). However, SNNs are difficult to train since spike trains are non-differentiable.

Several researchers have focused on the surrogate gradient to efficiently train SNNs [5–9]. The surrogate gradient is an approximation of the true gradient and is applied to the backpropagation (BP) algorithm [10]. Recent studies have successfully trained deep SNNs using this method [11]. However, it is still challenging to train deepened models due to the increasing difficulty of controlling spike firing.

To control the spike firing properly, we propose a novel and simple normalization: *postsynaptic potential normalization*. Contrary to the standard batch/layer normalizations, our normalization removes the subtraction term from the standard normalization and uses the second raw moment instead of the variance as the division term. We can automatically control the firing threshold of the membrane potential and spike firing by conducting this simple normalization to the postsynaptic potential (PSP). The experimental results on neuromorphic-MNIST (N-MNIST) [12] and Fashion-MNIST (F-MNIST) [13] show that SNNs with our normalization outperform other models using other normalizations. We also show that the proposed method can train the SNN, consisting of more than 100 layers without other special techniques dedicated to SNNs.

The contributions of this study are summarized as follows.

- We propose a novel and simple normalization technique based on the firing rate. The experimental results show that the proposed model can simultaneously achieve high classification accuracy and low firing rate;
- We trained deep SNNs based on the pre-activation residual blocks [14]. Consequently, we successfully obtained a model with more than 100 layers without other special techniques dedicated to SNNs.

The remainder of the paper is organized as follows. In Sections 2–4, we describe the related works, SNN used in this paper, and our normalization technique. Section 5 presents the experimental results. Finally, Section 6 presents the conclusion and future works.

2. Related Works

2.1. Spiking Neuron

SNN consists of spiking neurons that model the behavior of biological neurons and handle the firing timing of the spikes. Owing to the differences in approximations, several spiking neuron models have been proposed, such as the integrate-fire (IF) [15], leaky-integrate-and-fire (LIF) [16], Izhikevich [17], and Hodgkin–Huxley model [18]. In this study, we adopt the spike response model (SRM) [19] to deal with the refractory period (Section 3).

The refractory period is an essential function of biological neurons to suppress the spike firing. Spike firing occurs when the neuron's membrane potential exceeds the firing threshold. From a biological perspective, the membrane potential is calculated using PSP, representing the electrical signals converted from the chemical signals. These behaviors are represented within the chemical synapse model shown in Figure 1a [20]. SRM was implemented to approximate this synaptic model better than IF/LIF neurons, which are widely used in SNNs.

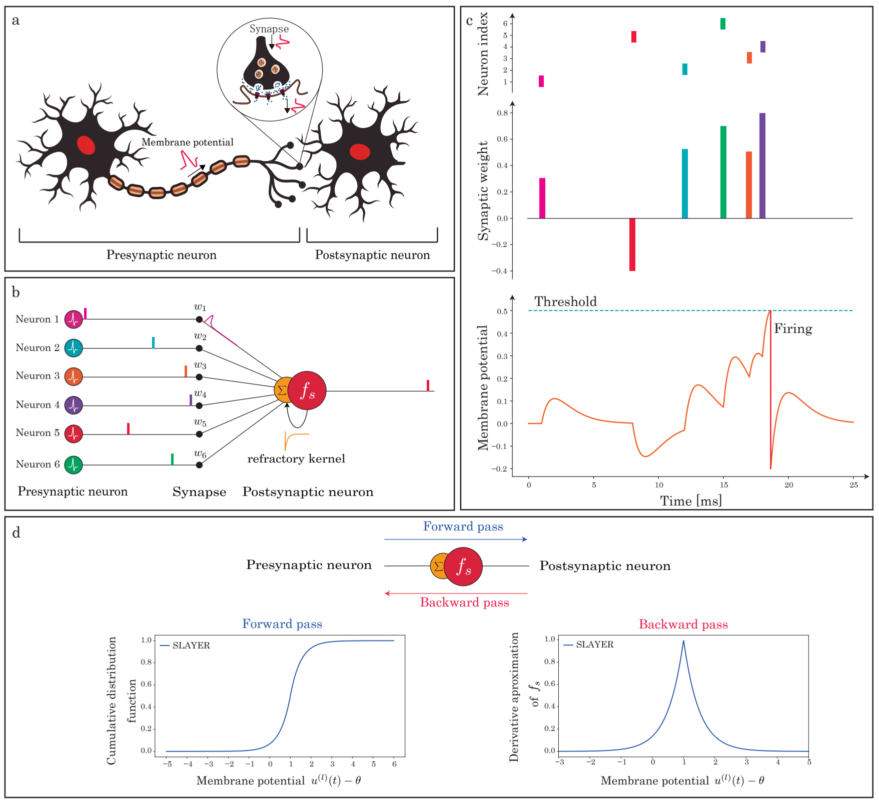

Figure 1. Illustration of fundamental components of SNN. (**a**) Biological information and signal processing between presynaptic and postsynaptic neurons in SNN. (**b**) A spiking neuron processes and communicates binary spiking events over time. (**c**) The postsynaptic neuron changes the membrane potential through each layer of the post-synaptic potential (PSP). It generates the output spikes when the membrane potential reaches the neuronal firing threshold [21]. (**d**) The cumulative distribution function ("Forward pass") of our surrogate gradient and itself ("Backward pass") [6].

2.2. Training of Spiking Neural Networks

It is well-known that SNNs are difficult to train due to non-differential spike trains. Researchers are working on this problem, and their solutions can be divided into two approaches: first, the ANN-SNN conversion [22–25], and second, the usage of the surrogate gradient [5–9]. The ANN-SNN conversion method uses the trained ANN parameters of SNN. The sophisticated and state-of-the-art ANN model can be reused through this method. However, this conversion approach requires many time-steps during inference and increases the power consumption. In contrast, the surrogate gradient is used to directly train SNNs by approximating the gradient of the non-differentiable spiking neurons. The surrogate gradient approach was adopted since the model obtained by surrogate gradient requires far fewer inference time-steps than the ANN-SNN conversion model [26].

2.3. Normalization

One of the techniques that have contributed to the success of ANNs is Batch Normalization (BN) [27]. BN is used to reduce the internal covariate shift, leading to a smooth landscape [28] while corresponding to the homeostatic plasticity mechanism of BNNs [29]. Using a mini-batch, BN computes the sample mean and standard deviation (STD). Meanwhile, several variants have been proposed to compute the sample mean and STD, such as Layer Normalization (LN) [30], Instance Normalization (IN) [31], and Group Normalization (GN) [32]. In particular, LN is effective at stabilizing the hidden state dynamics in recurrent neural networks for time-series processing [30].

Several normalization methods have also been proposed in the field of SNNs, such as threshold-dependent BN (tdBN) [33] and BN through time (BNTT) [34]. Thus, tdBN incorporates the firing threshold into BN, whereas BNTT computes BN at each time step. Furthermore, some studies used BN as is [35]. These studies applied the normalization to the membrane potential. In contrast, our method was applied to PSP, as shown in Figure 2b, to simplify the normalization form (Section 4).

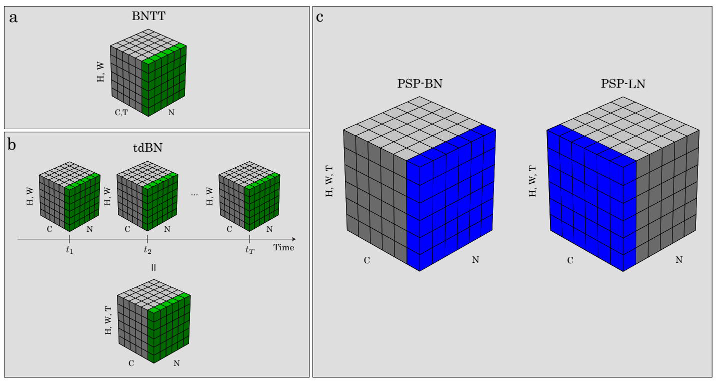

Figure 2. Normalization methods for spiking neural networks (SNNs). Each subplot shows a feature map (**a**,**b**) or post-synaptic input (**b**) tensor, with N as the batch axis, C as the channel axis, (H, W) as the spatial axis, and T as the time axis in the figure. Green voxels (**a**,**b**), previous methods, and blue (**c**), our proposed method, are normalized by the same second central moment and uncentered second moment, respectively.

3. Spiking Neural Networks Based on the Spike Response Model

In this section, we describe the SNN used in this study. Our SNN is constructed using SRM [19]; it uses SLAYER [6] as the surrogate gradient function to train the SRM.

3.1. Spike Response Model

We adopt SRM as a spiking neuron model [19]. SRM model is based on combining the effects of the incoming spike arriving at the spiking neuron. It also has a function to the spike firing when the membrane potential $u(t)(t = 1, 2, \cdots, T)$ reaches the firing threshold. Figure 1b,c indicate the behavior of this model. The equations are given as follows:

$$u_i(t) = \sum_j w_{ij}(\varepsilon * s_j)(t) + (\nu * s_i)(t), \tag{1}$$

$$s_i(t) = f_s(u_i(t) - \theta), \tag{2}$$

where $w_{i,j}$ is the synaptic weight from the presynaptic neuron j to the postsynaptic neuron i. $s_j(t)$ is the spike train inputted from the presynaptic neuron j, $s_i(t)$ is the output spike train of the postsynaptic neuron i, $*$ is a temporal convolution operator, and θ is a threshold used to control the spike generation. f_s is the Heaviside step function, which fires the spike when the membrane potential $u_i(t)$ exceeds the firing threshold θ as shown in Figure 3. In addition, $\varepsilon(\cdot)$ and $\nu(\cdot)$ are the spike response and refractory kernels formulated using the exponential function as follows:

$$\varepsilon(t) = \frac{t}{\tau_s} e^{1-\frac{t}{\tau_s}}, \tag{3}$$

$$\nu(t) = -2\theta e^{-\frac{t}{\tau_r}}, \tag{4}$$

where τ_s and τ_r are the time constants of spike response and refractory kernels, respectively. Note that $\varepsilon * s_j(t)$ represents the PSP. After firing, the postsynaptic neuron goes into the refractory period and cannot fire until its membrane potential resets to its resting potential.

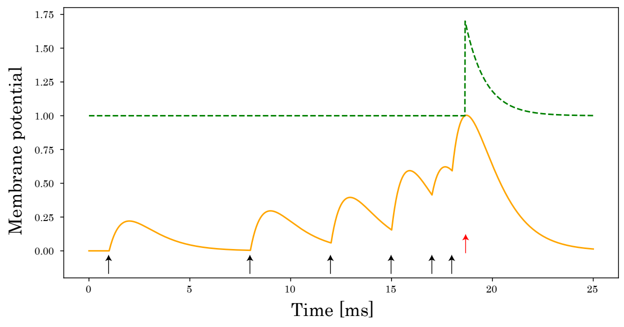

Figure 3. A dynamic threshold in SRM. The membrane potential (orange solid line) of the postsynaptic neuron is described by the superposition of the PSPs. Each input spike arrival and output spike are denoted by a black arrow and red arrow, respectively. At the moment of spiking, the firing threshold (green dashed line) increases and decreases with time to the initial value of the firing threshold.

The main role of the refractory period suppresses the firing rate at a given spike interval. If the spike interval T is constant, the firing rate without the refractory period is given by $1/T$. On the other hand, if the refractory period r is taken into account, it can be rewritten as $1/(T + r)$. Therefore, the firing rate decreases as the refractory period increases, as shown in Figure 4. In SNNs, the firing rate is proportional to the computational cost. Namely, using the refractory period ensures biological plausibility and reduces computational costs.

3.2. Multiple Layers Spike Response Model

By using Equations (1) and (2), the SNNs with multi-layers can be described as follows:

$$\boldsymbol{a}^{(l)}(t) = (\varepsilon * \boldsymbol{s}^{(l)})(t), \tag{5}$$

$$\boldsymbol{u}^{(l+1)}(t) = \boldsymbol{W}^{(l)}\boldsymbol{a}^{(l)}(t) + (\nu * \boldsymbol{s}^{(l+1)})(t), \tag{6}$$

$$\boldsymbol{s}^{(l+1)}(t) = f_s(\boldsymbol{u}^{(l+1)}(t)), \tag{7}$$

where $\boldsymbol{a}^{(l)}(t) \in \mathbb{R}_{\geq 0}^{\mathrm{C}\times\mathrm{W}\times\mathrm{H}}$ and $\boldsymbol{s}^{(l)}(t) \in \{0,1\}^{\mathrm{C}\times\mathrm{W}\times\mathrm{H}}$ are the PSP and input spike tensor of time step t; C is the number of channels; and W and H are the width and height of the input spike tensor, respectively. Since $\boldsymbol{a}^{(l)}(t)$ does not take a value less than zero, we consider an excitatory neuron. Furthermore, $\boldsymbol{W}^{(l)} \in \mathbb{R}^M$ is the weight matrix representing the synaptic strengths between the spiking neurons in l and $l+1$ layers; M is the number of neurons of $l+1$-th layer.

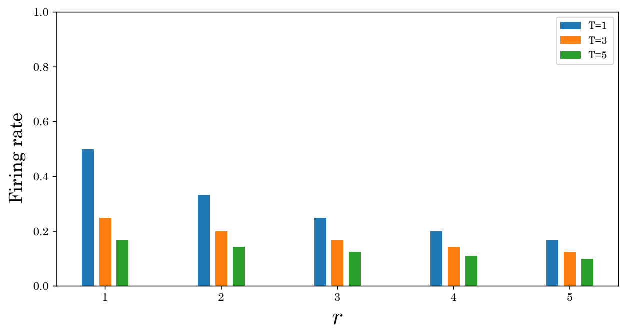

Figure 4. Relationship between refractory period $r = (\nu * s_i)$ and firing rate for the different spike intervals $T = 1, 3, 5$ [ms].

3.3. Deep SNNs by Pre-Activation Blocks

A deep neural network is essential to recognize complex input patterns. In particular, ResNet is widely used in ANNs [14,36], and its use in SNNs is expanding.

The ResNet's networks are divided into the pre-activation and post-activation residual blocks, as follows (Figure 5):

$$\text{Pre}: \boldsymbol{h}^{(k+1)}(t) = \boldsymbol{h}^{(k)}(t) + G(\boldsymbol{h}^{(k)}(t)), \tag{8}$$

$$\text{Post}: \boldsymbol{h}^{(k+1)}(t) = F(\boldsymbol{h}^{(k)}(t) + G(\boldsymbol{h}^{(k)}(t))), \tag{9}$$

where $\boldsymbol{h}^{(k)}$ and $\boldsymbol{h}^{(k+1)}$ are the input and output in the $k+1$ block, respectively. G represents the residual function, corresponding to "Conv-Func-Conv" and "Func-Conv-Func-Conv" in Figure 5; F represents the Func layer ("Spike-PSP-Norm"). Note that the refractory period is used in F. In the experimental section, we compare these blocks and show that deep SNNs can be trained using the pre-activated residual blocks. This result shows that identity mapping is an essential tool to train the deep SNNs, similar to ANNs [14].

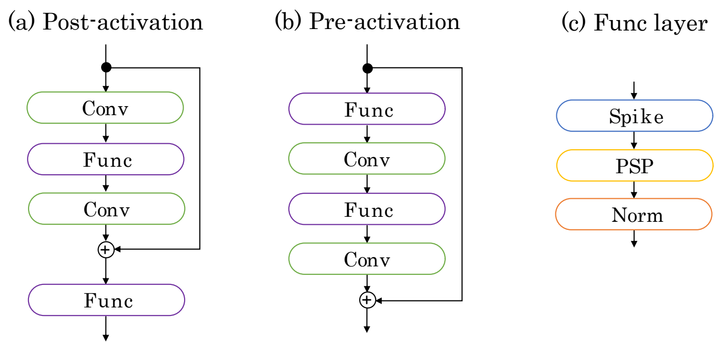

Figure 5. Network blocks. (**a**) Post-activation residual block, (**b**) Pre-activation residual block, and (**c**) Func layer. Postsynaptic potential (PSP) and Norm in the Func layer represent Equation (5) and normalization. Spike represents $f_s(\cdot + r)$, where r is the refractory period.

3.4. Surrogate-Gradient

We use SLAYER [6] as one of the surrogate gradient algorithms to tarin the SNN with multi-layers. In SLAYER, the derivative of the spike activation function f_s of the $l+1$ layer is approximated as follows (Figure 1d):

$$\boldsymbol{\rho}^{(l+1)}(t) = \frac{1}{\alpha}\exp(-\beta|\boldsymbol{u}^{(l+1)}(t) - \boldsymbol{\theta}|), \tag{10}$$

where α and β are hyperparameters to adjust the peak value and sharpness for the surrogate gradient, and $\boldsymbol{\theta} \in \mathbb{R}^M$ is the firing threshold. SLAYER can be used to train SRM as described in [6].

4. Normalization of Postsynaptic Potential

In this section, we explain the derivation of our normalization, which is called *postsynaptic-potential normalization*, as shown in Figure 6a.

As the depth of the SNN becomes deeper, it becomes more difficult to control spike firing properly (Figure 6b,c). To tackle this problem, we first introduce the following typical normalization into the PSP.

$$\hat{\boldsymbol{u}}^{(l+1)}(t) = \boldsymbol{W}^{(l)}\hat{\boldsymbol{a}}^{(l)}(t) + (\nu * \boldsymbol{s}^{(l+1)})(t), \tag{11}$$

$$\hat{\boldsymbol{a}}^{(l)}(t) = \frac{\boldsymbol{a}^{(l)}(t) - \mathbb{E}_X[\boldsymbol{a}^{(l)}]}{\sqrt{\mathbb{V}_X[\boldsymbol{a}^{(l)}] + \lambda}} \odot \gamma + \xi, \tag{12}$$

where γ and ξ are trainable parameters, and the $\odot$ operator denotes the Hadamard product; each variable of $\mathbb{E}_X[\boldsymbol{a}^{(l)}]$ and $\mathbb{V}_X[\boldsymbol{a}^{(l)}]$ is approximated as follows:

$$\mathbb{E}_X[a_i^{(l)}] \approx \frac{1}{X}\sum_{x=1}^{X} a_i^{(l)}(x), \tag{13}$$

$$\mathbb{V}_X[a_i^{(l)}] \approx \frac{1}{X}\sum_{x=1}^{X} (a_i^{(l)}(x) - \mathbb{E}_X[a_i^{(l)}]) \odot (a_i^{(l)}(x,t) - \mathbb{E}_X[a_i^{(l)}]), \tag{14}$$

where $a_i^{(l)}(x)$ represents the x-th variable required to compute these statistics of the i-th variable of $\boldsymbol{a}^{(l)} \in \mathbb{R}_{\geq 0}^{\mathrm{C}\times\mathrm{W}\times\mathrm{H}\times\mathrm{N}\times\mathrm{T}}$ (N is the mini-batch size), and X depends on what kind of summation to compute. For example, if we compute these equations as in BN, $X = W \times H \times N \times T$. In addition, if we compute them as in LN, $X = W \times H \times C \times T$. Note that the normalization to PSP means that it inserts before the convolution or fully connected layers. This position differs from the other normalization ones, which use normalization to the membrane potential [33–35].

As shown in Equation (12), $\hat{\boldsymbol{a}}^{(l)}(t)$ may take minus. Therefore, $\hat{\boldsymbol{a}}^{(l)}(t) < 0$ is not valid since neurons of SLAYER represent excitatory neurons. This phenomenon clearly arises from the trainable parameter ξ and the shift parameter $\mathbb{E}_X[\boldsymbol{a}^{(l)}]$. Thus, we modify Equation (12) as follows:

$$\hat{\boldsymbol{a}}^{(l)}(t) = \frac{\boldsymbol{a}^{(l)}(t)}{\sqrt{\mathbb{V}_X[\boldsymbol{a}^{(l)}] + \lambda}} \odot \gamma. \tag{15}$$

Next, we consider the case when $\hat{\boldsymbol{u}}^{(l+1)}(t)$ reaches the firing threshold $\boldsymbol{\theta}$.

$$\boldsymbol{\theta} = \boldsymbol{W}^{(l)}\hat{\boldsymbol{a}}(\boldsymbol{t}) + (\nu * \boldsymbol{s}^{(l+1)})(t), \tag{16}$$

$$= \hat{\boldsymbol{W}}^{(l)}\frac{\boldsymbol{a}^{(l)}(t)}{\sqrt{\mathbb{V}_X[\boldsymbol{a}^{(l)}] + \lambda}} + (\nu * \boldsymbol{s}^{(l+1)})(t). \tag{17}$$

Here, we have merged the trainable parameter γ and the weight matrix $\boldsymbol{W}^{(l)}$ into $\hat{\boldsymbol{W}}^{(l)}$. This merging is possible because of the normalization performed before multiplying $\boldsymbol{W}^{(l)}$. Then, we express Equation (17) as follows:

$$\hat{\boldsymbol{W}}^{(l)}\boldsymbol{a}^{(l)}(t) = \sqrt{\mathbb{V}_X[\boldsymbol{a}^{(l)}] + \lambda}(\boldsymbol{\theta} - \nu * \boldsymbol{s}^{(l+1)}(t)) := \hat{\boldsymbol{\theta}}. \tag{18}$$

Equation (18) shows that the firing threshold varies dynamically as shown in Figure 3, which is consistent with the activity of cortical neurons in the human brain [37–40]. The refractory period $(\nu * \boldsymbol{s}^{(l+1)})(t)$ and $\sqrt{\mathbb{V}_X[\boldsymbol{a}^{(l)}] + \lambda}$ can decrease $\hat{\boldsymbol{\theta}}$ and scaling, respectively.

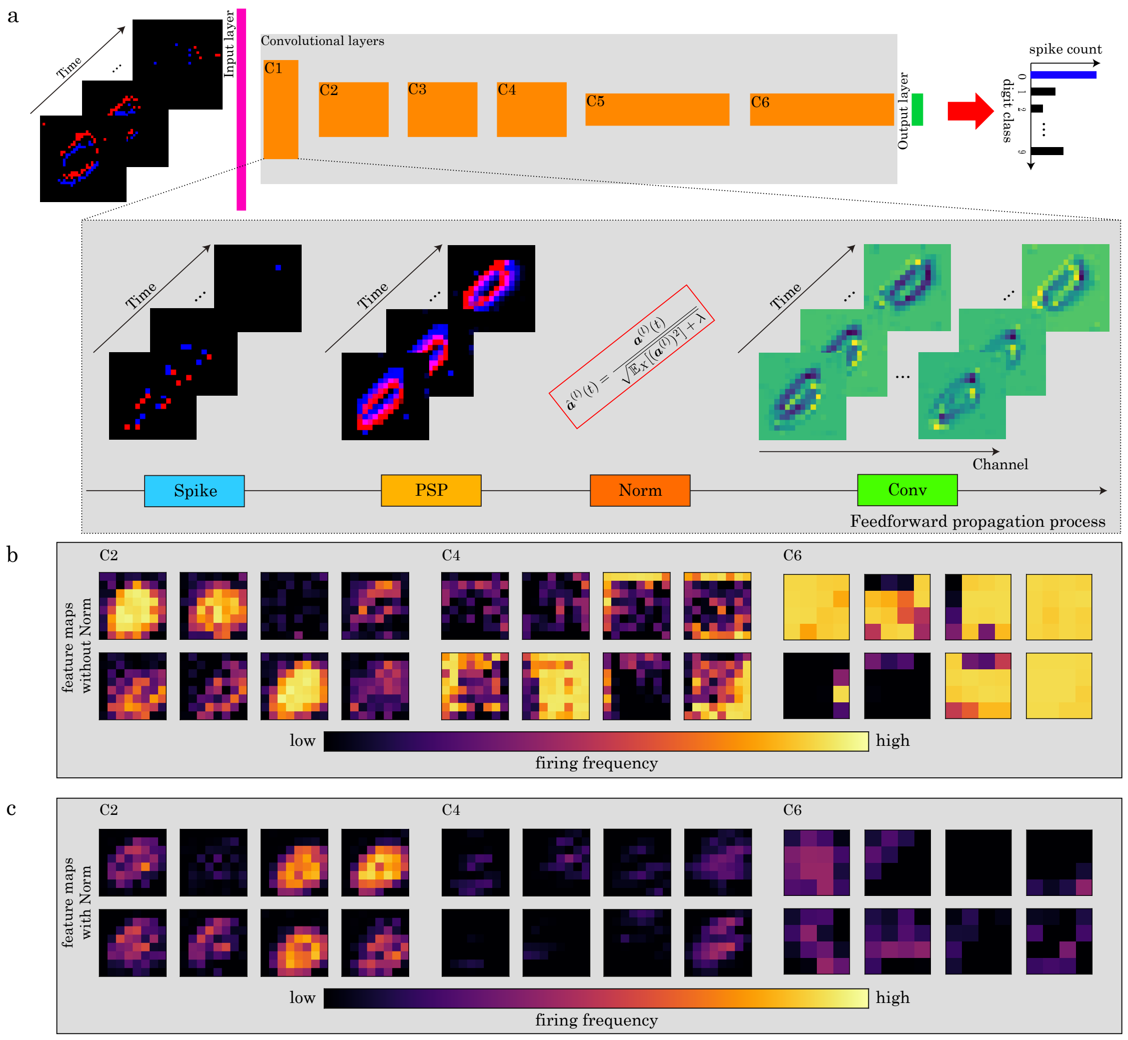

Figure 6. (**a**) Overview of the forward propagation phase using postsynaptic potential (PSP) normalization on the N-MNIST dataset. (**b**) Example of feature maps, which are integrated along the time axis, without PSP normalization. (**c**) As in (**b**) but for feature maps with it. PSP normalization controls the activity of the network and prevents the over-firing of the neurons.

Next, we focus on the scale factor $\sqrt{\mathbb{V}_X[\boldsymbol{a}^{(l)}] + \lambda}$. As shown in Equation (18), the firing threshold $\hat{\boldsymbol{\theta}}$ becomes larger as the variance (second central moment) $\mathbb{V}_X[\boldsymbol{a}^{(l)}]$ increases. However, considering the behavior of the membrane potential, $\hat{\boldsymbol{\theta}}$ should become larger when the value of PSP (not variance) increases. Thus, we modify the equation as follows.

$$\hat{\boldsymbol{a}}^{(l)}(t) = \frac{\boldsymbol{a}^{(l)}(t)}{\sqrt{\mathbb{E}_X[(\boldsymbol{a}^{(l)})^2] + \lambda}}, \tag{19}$$

where $\mathbb{E}_X[(\boldsymbol{a}^{(l)})^2]$ represents the second raw moment consisting of the following variable,

$$\mathbb{E}_X[(a_i^{(l)})^2] \approx \frac{1}{X}\sum_{x=1}^{X} a_i^{(l)}(x) \odot a_i^{(l)}(x). \tag{20}$$

By using this equation, we do not have to compute the mean beforehand, in contrast to using the variance.

In addition to $\mathbb{E}_X[(\boldsymbol{a}^{(l)})^2]$, there is a hyperparameter λ in the scale factor. λ is usually set to a small constant, e.g., $\lambda = 10^{-3}$ because it plays the role of the numerical stability. Figure 7 shows the relationship between $\mathbb{E}_X[(a_i^{(l)})^2]$ and $\hat{\theta}$ when changing θ and λ. As shown in this figure, $\hat{\theta}$ monotonically decreases as $\mathbb{E}_X[(a_i^{(l)})^2]$ decreases. In particular, $\hat{\theta}$ is close to zero when λ is sufficiently small, regardless of the initial threshold θ. $\hat{\theta} \approx 0$ means that spikes fire at all times even if the membrane potential is significantly small, making it difficult to train a proper model. Thus, we set a relatively large value ($\lambda = 0.1$) as the default value.

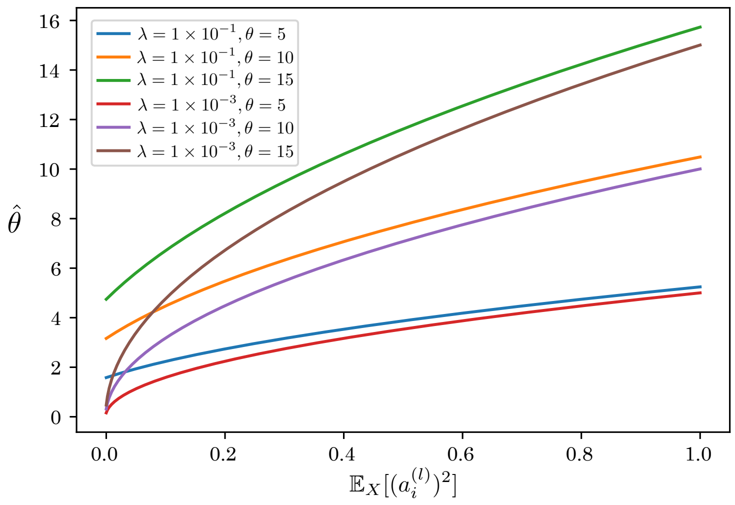

Figure 7. Relationship between $\mathbb{E}_X[(a_i^{(l)})^2]$ and $\hat{\theta}$.

5. Experiments

In this section, we evaluate two PSP normalizations: BN (the most common normalization) and LN (which is effective in time-series processing, such as SNN). We called them *PSP-BN* ($X = W \times H \times N \times T$) and *PSP-LN* ($X = W \times H \times C \times T$).

5.1. Experimental Setup

We evaluated PSP-BN and PSP-LN on the spatio-temporal event and static image datasets. We used N-MNIST [12] and F-MNIST [13]. N/F-MNISTs are widely used datasets containing 60 K training and 10K test samples with 10 classes. Each size is $34 \times 34\times$ 30,000 events (N-MNIST), and 28×28 pixels (F-MNIST). We partitioned the 60 K data using 54 K and 6 K as our training and validation data, respectively. We also resized the F-MNIST image from 28×28 to 34×34 to achieve higher accuracy.

We evaluated the performance of several spiking convolutional neural network models, such as 14-convolutional layers on N/F-MNIST. We also used more deep models, such as ResNet-106 on N-MNIST and F-MNIST, respectively.

We used hyperparameters shown in Table 1 in all experiments and implemented by PyTorch. We used the default initialization of PyTorch and showed the best accuracies of all models. All experiments were conducted using a single Tesla V100 GPU. In addition to this computational resource limitation, we randomly sampled 6 K of the training data for both datasets to train in each epoch since SLAYER requires a significant amount of time to train.

Table 1. Hyperparameter setting on N-MNIST and F-MNIST.

Hyperparameter	**N-MNIST**	**F-MNIST**
τ_s	10	10
τ_r	10	10
α	10	10
β	10	10
θ	10	10
optimizer	AdaBelief	AdaBelief
learning rate	10^{-2}	10^{-2}
weight decay	10^{-4}	10^{-4}
weight scale	10	10
mini-batch size	10	10
time step	300	100
epoch	100	100

5.2. Effectiveness of Postsynaptic Potential Normalization

We first evaluate the effectiveness of our normalizations. Table 2 presents the accuracies of PSP-BN and PSP-LN and other approaches. Note that we set our normalization before the convolution as described in Section 4, which is different from the position proposed in previous studies [33–35]. This table illustrates that PSP-BN and PSP-LN achieve high accuracies on both datasets compared to the other approaches.

We also investigate the effect of the proposed method on the firing rate. Figures 8 and 9 show the firing rates of each method. As shown in Figure 8, our normalized models can suppress the firing rate in most layers compared to the unnormalized model. Furthermore, Figure 9 and Table 2 show that our normalized models can simultaneously achieve high classification accuracy and low firing rate compared to other normalizations. These results verify the effectiveness of our normalizations.

Then, we also analyze the training and inference times of the proposed method. Figure 10 shows the computational cost of BN, PSP-BN, and PSP-LN. The training time of PSP-BN and PSP-LN is shorter than BN because our normalization method does not require training parameters (γ and ξ) as Equation (17). On the other hand, the training time of PSP-BN and PSP-LN are almost the same because these differ only in X. In addition, there is no significant difference in inference time for each normalization. These results show that our normalization is suitable for training SNN.

Table 2. Accuracies of N-MNIST and F-MNIST obtained from different methods. PSP-BN and PSP-LN are our normalization methods, and None is the model without normalization. Here, "c", "n", and "o" represent the convolution, normalization, and output neurons, respectively. In addition, each layer and spatial dimension in the network are separated by "-" and "×".

Method	**Dataset**	**Network Architecture**	**Acc. (%)**
BN [35]	N-MNIST	34×34×2-8c3n-{16c3n}*5-16c3n-{32c3n}*5-10o	85.1
BNTT [34]	N-MNIST	34×34×2-8c3n-{16c3n}*5-16c3n-{32c3n}*5-10o	90.0
tdBN [33]	N-MNIST	34×34×2-8c3n-{16c3n}*5-16c3n-{32c3n}*5-10o	81.8
PSP-BN	N-MNIST	34×34×2-n8c3-{n16c3}*5-n16c3-{n32c3}*5-10o	97.4
PSP-LN	N-MNIST	34×34×2-n8c3-{n16c3}*5-n16c3-{n32c3}*5-10o	98.2
None	N-MNIST	34×34×2-8c3-{16c3}*5-16c3-{32c3}*5-10o	40.6
BN [35]	F-MNIST	34×34-16c3n-{32c3n}*5-32c3n-{64c3n}*5-10o	10
BNTT [34]	F-MNIST	34×34-16c3n-{32c3n}*5-32c3n-{64c3n}*5-10o	10
tdBN [33]	F-MNIST	34×34-16c3n-{32c3n}*5-32c3n-{64c3n}*5-10o	40.5
PSP-BN	F-MNIST	34×34-n16c3-{n32c3}*5-n32c3-{n64c3}*5-10o	88.6
PSP-LN	F-MNIST	34×34-n16c3-{n32c3}*5-n32c3-{n64c3}*5-10o	89.1
None	F-MNIST	34×34-16c3-{32c3}*5-32c3-{64c3}*5-10o	84.1

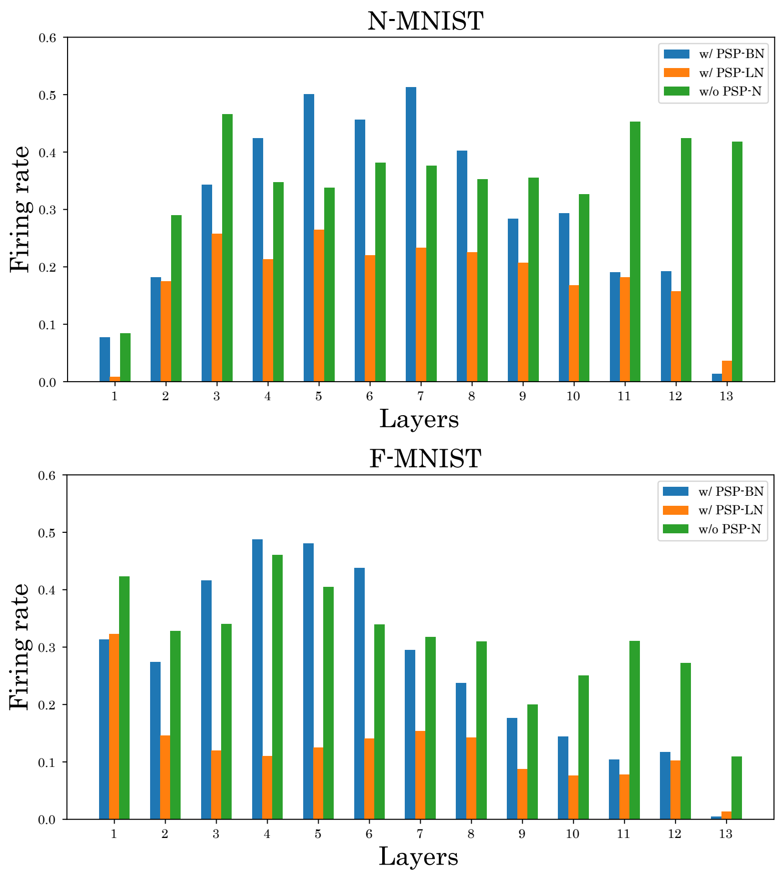

Figure 8. Comparison of firing rates. The top is N-MNIST and the bottom is F-MNIST.

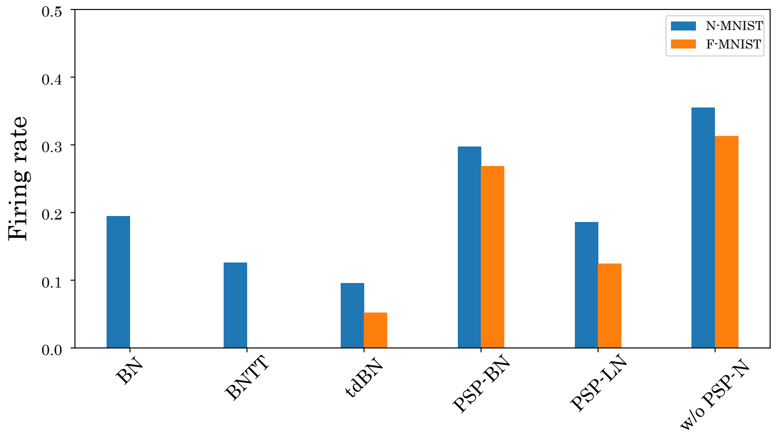

Figure 9. Total firing rates of each model.

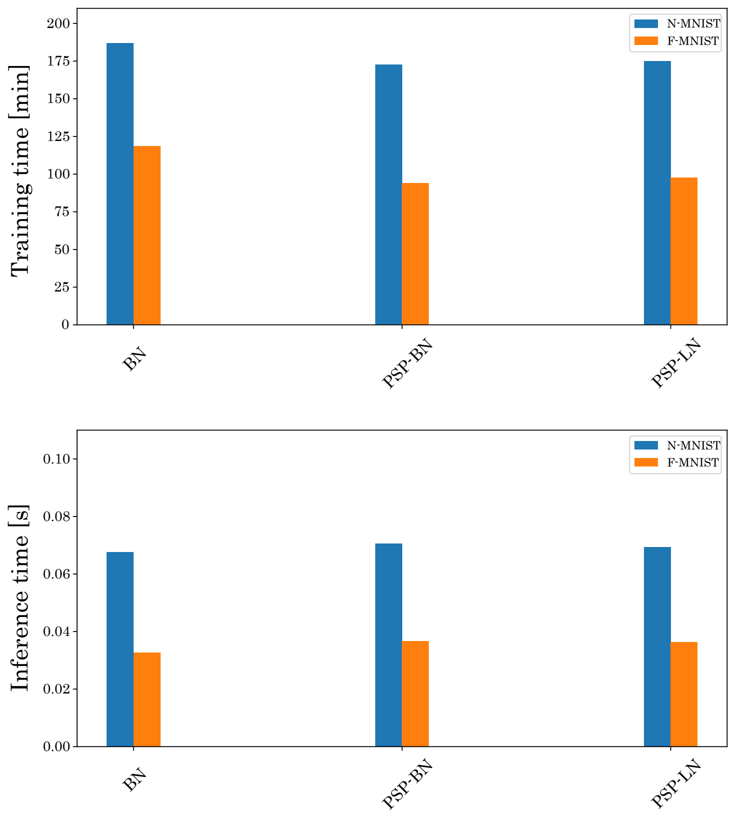

Figure 10. Comparison of the training and inference time by using BN, PSP-BN and PSP-LN. The top is the training time and the bottom is the inference time.

5.3. Performance Evaluation of Deep SNNs by Residual Modules

Finally, we evaluate the performance of SNNs using the residual blocks. Table 3 shows the performance of SNNs using the pre-activation and post-activation residual blocks. As shown in this table, the accuracy is substantially improved using the pre-activation residual blocks. This result shows that the post-activation employed in previous studies without refractory period [5,11,33] is unsuitable for SNNs with a refractory period. Thus, while ensuring the biological plausibility, due to the refractory period, we can obtain deep SNNs beyond 100 layers using our normalizations and pre-activation residual blocks.

Table 3. Performance comparison using post-activation and pre-activation residual blocks. We use ResNet-106 on N-MNIST and F-MNIST datasets, respectively.

Meshod	Dataset	Network Architecture	Acc. (%)
PSP-BN	N-MNIST	Post-activation ResNet-106	10.0
PSP-BN	N-MNIST	Pre-activation ResNet-106	75.4
PSP-LN	N-MNIST	Post-activation ResNet-106	10.0
PSP-LN	N-MNIST	Pre-activation ResNet-106	86.8
PSP-BN	F-MNIST	Post-activation ResNet-106	10.0
PSP-BN	F-MNIST	Pre-activation ResNet-106	81.6
PSP-LN	F-MNIST	Post-activation ResNet-106	10.0
PSP-LN	F-MNIST	Pre-activation ResNet-106	82.1

6. Discussion and Conclusions

In this study, we proposed an appropriate normalization method for SNN. The proposed normalization removes the subtraction term from the standard normalization and uses the second raw moment as the denominator. Our normalized models outperformed other normalized models based on existing normalization such as BN, BNTT, and tdBN by inserting this simple normalization before the convolutional layer. Furthermore, our proposed model with pre-activation residual blocks can train with more than 100 layers without any other special techniques dedicated to SNNs.

Besides the type of normalization, some papers pointed out that tuning hyperparameters τ_s and λ is essential for high accuracy [41,42]. Investigating this aspect, we found that PSP-BN is sensitive to λ, whereas PSP-LN is robust to τ_s and λ (Figures 11 and 12). These results imply that the effectiveness of tuning hyperparameters depends on X. We will conduct more detailed analysis in this regard in the future. In addition, we will also analyze the effect on other datasets and networks. Furthermore, we aim to extend postsynaptic normalization based on tdBN to develop robust normalization techniques for the thresholds in spiking neurons.

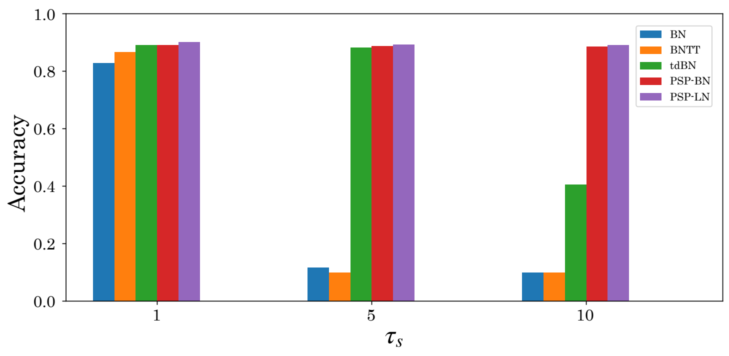

Figure 11. Accuracy comparison of F-MNIST dataset with respect to changing hyperparameter τ_s $(= \{1, 5, 10\})$.

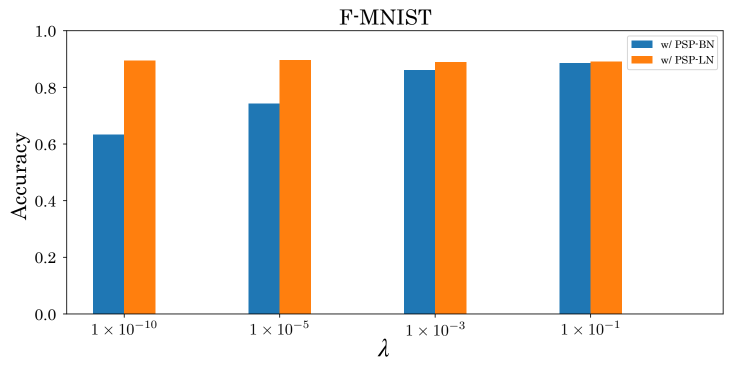

Figure 12. Accuracy comparison of F-MNIST with respect to changing λ $(= \{10^{-10}, 10^{-5}, 10^{-3}, 10^{-1}\})$.

Author Contributions: S.-i.I. wrote the manuscript, R.S. wrote the code, Y.S. and N.N. reviewed the work and contributed via discussions. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.* **1997**, *10*, 1659–1671. [CrossRef]
2. Akopyan, F.; Sawada, J.; Cassidy, A.; Alvarez-Icaza, R.; Arthur, J.; Merolla, P.; Imam, N.; Nakamura, Y.; Datta, P.; Nam, G.-J.; et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 1537–1557. [CrossRef]
3. Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* **2018**, *38*, 82–99. [CrossRef]
4. Maguire, L.P.; McGinnity, T.M.; Glackin, B.; Ghani, A.; Belatreche, A.; Harkin, J. Challenges for large-scale implementations of spiking neural networks on FPGAs. *Neurocomputing* **2007**, *71*, 13–29. [CrossRef]
5. Lee, C.; Sarwar, S.S.; Panda, P.; Srinivasan, G.; Roy, K. Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci.* **2020**, *14*, 119. [CrossRef]
6. Shrestha, S.B.; Orchard, G. Slayer: Spike layer error reassignment in time. *arXiv* **2018**, arXiv:1810.08646.
7. Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* **2018**, *12*, 331. [CrossRef]
8. Zenke, F.; Ganguli, S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput.* **2018**, *30*, 1514–1541. [CrossRef]
9. Zhang, W.; Li, P. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12022–12033.
10. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
11. Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; Tian, Y. Deep Residual Learning in Spiking Neural Networks. *arXiv* **2021**, arXiv:2102.04159.
12. Orchard, G.; Jayawant, A.; Cohen, G.K.; Thakor, N. Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* **2015**, *9*, 437. [CrossRef]
13. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
14. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity mappings in deep residual networks. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; Springer: Cham, Switzerland, 2016; pp. 630–645.
15. Lapique, L. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *J. Physiol. Pathol.* **1907**, *9*, 620–635.
16. Stein, R.B. A theoretical analysis of neuronal variability. *Biophys. J.* **1965**, *5*, 173–194. [CrossRef]
17. Izhikevich, E.M. Simple model of spiking neurons. *IEEE Trans. Neural Netw.* **2003**, *14*, 1569–1572. [CrossRef]
18. Hodgkin, A.L.; Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **1952**, *117*, 500–544. [CrossRef]
19. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press: Cambridge, UK, 2002.
20. Rall, W. Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic input. *J. Neurophysiol.* **1967**, *30*, 1138–1168. [CrossRef]
21. Comsa, I.M.; Potempa, K.; Versari, L.; Fischbacher, T.; Gesmundo, A.; Alakuijala, J. Temporal coding in spiking neural networks with alpha synaptic function. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 8529–8533.
22. Diehl, P.U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.-C.; Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8.
23. Li, Y.; Deng, S.; Dong, X.; Gong, R.; Gu, S. A Free Lunch From ANN: Towards Efficient, Accurate Spiking Neural Networks Calibration. In Proceedings of the International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 6316–6325.
24. Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; Liu, S.-C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* **2017**, *11*, 682. [CrossRef] [PubMed]
25. Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; Roy, K. Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* **2019**, *13*, 95. [CrossRef]
26. Diehl, P.U.; Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* **2015**, *9*, 99. [CrossRef] [PubMed]
27. Ioffe, S.; Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 448–456.
28. Santurkar, S.; Tsipras, D.; Ilyas, A.; Mądry, A. How does batch normalization help optimization? In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 2488–2498.
29. Shen, Y.; Wang, J.; Navlakha, S. A correspondence between normalization strategies in artificial and biological neural networks. *Neural Comput.* **2021**, *33*, 3179–3203. [CrossRef] [PubMed]

30. Ba, J.L.; Kiros, J.R.; Hinton, G.E. Layer normalization. *arXiv* **2016**, arXiv:1607.06450.
31. Ulyanov, D.; Vedaldi, A.; Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv* **2016**, arXiv:1607.08022.
32. Wu, Y.; He, K. Group normalization. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
33. Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; Li, G. Going Deeper With Directly-Trained Larger Spiking Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11062–11070.
34. Kim, Y.; Panda, P. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Front. Neurosci.* **2021**, *15*, 773954. [CrossRef]
35. Ledinauskas, E.; Ruseckas, J.; Juršėnas, A.; Buračas, G. Training deep spiking neural networks. *arXiv* **2020**, arXiv:2006.04436.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
37. Aertsen, A.; Braitenberg, V. (Eds.) *Brain Theory: Biological Basis and Computational Principles*; Elsevier: Amsterdam, The Netherlands, 1996.
38. Frankenhaeuser, B.; Vallbo, Å.B. Accommodation in myelinated nerve fibres of Xenopus laevis as computed on the basis of voltage clamp data. *Acta Physiol. Scand.* **1965**, *63*, 1–20. [CrossRef]
39. Schlue, W.R.; Richter, D.W.; Mauritz, K.H.; Nacimiento, A.C. Responses of cat spinal motoneuron somata and axons to linearly rising currents. *J. Neurophysiol.* **1974**, *37*, 303–309. [CrossRef]
40. Stafstrom, C.E.; Schwindt, P.C.; Flatman, J.A.; Crill, W.E. Properties of subthreshold response and action potential recorded in layer V neurons from cat sensorimotor cortex in vitro. *J. Neurophysiol.* **1984**, *52*, 244–263. [CrossRef]
41. Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 2661–2671.
42. Li, Y.; Guo, Y.; Zhang, S.; Deng, S.; Hai, Y.; Gu, S. Differentiable Spike: Rethinking Gradient-Descent for Training Spiking Neural Networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12022–12033.