Supplementary Material for *Temporal Effective Batch Normalization in Spiking Neural Networks*

A Theoretical Proof

Here we provide the theoretical proof for **Theorem 1** in the main text.

Proof. Here we consider the gradient w.r.t the time-dependent input signal $\boldsymbol{x}_i[t]$ where t is the current time-step and i is the neuron index. The loss $\mathcal{L}$ is defined in the network with TEBN, and the loss $\tilde{\mathcal{L}}$ is defined in the network without TEBN but with the same architecture. Besides, $\hat{\boldsymbol{x}}\left[t\right]=\hat{\boldsymbol{\gamma}}\left[t\right]\frac{\boldsymbol{x}[t]-\boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2+\epsilon}}+\hat{\boldsymbol{\beta}}\left[t\right]$.

When the surrogate function is defined on the non-differentiable spiking function, the gradients are thus feasible. The gradients satisfy the following condition according to the fact of gradient through Batch Normalization [9]:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{x}_i\left[t\right]}=\frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]}{\boldsymbol{\sigma}_i}\left[\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}-\mathbf{1}\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\right)-\boldsymbol{h}_i\left[t\right]\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\boldsymbol{h}_i\left[t\right]\right)\right] \tag{S1}$$

where $\boldsymbol{h}\left[t\right]=\frac{\boldsymbol{x}[t]-\boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2+\epsilon}}$ is the normed current at time-step t. First, we point out two important facts that help the analysis:

$$\mathbb{E}\left(\boldsymbol{h}_i\left[t\right]\right)=0, \tag{S2}$$

$$\mathbb{E}\left(\boldsymbol{h}_i\left[t\right]^2\right)=1, \tag{S3}$$

$$\left\|\boldsymbol{h}_i\left[t\right]\right\|^2=k=k\mathbb{E}\left(\boldsymbol{h}_i\left[t\right]^2\right), \tag{S4}$$

where k is the mini-batch size. We first partition the items in the right square bracket of Eq. S1 and let the expression in the square bracket be ϕ:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{x}_i\left[t\right]}&=\frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]}{\boldsymbol{\sigma}_i}\left[\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}-1\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\right)\right)-\boldsymbol{h}_i\left[t\right]\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\boldsymbol{h}_i\left[t\right]\right)\right]\\&=\frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]}{\boldsymbol{\sigma}_i}\phi.\end{aligned} \tag{S5}$$

When squaring ϕ, we have:

$$\begin{aligned}\phi^2=&\left\|\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}-1\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\right)\right\|^2+\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\boldsymbol{h}_i\left[t\right]\right)^2k\\&-2k\mathbb{E}\left(\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}-1\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\right)\right)\boldsymbol{h}_i\left[t\right]\right)\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\boldsymbol{h}_i\left[t\right]\right),\end{aligned} \tag{S6}$$

where

$$\mathbb{E}\left(\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}-1\cdot\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\right)\right)\boldsymbol{h}_i\left[t\right]\right)=\mathbb{E}\left(\frac{\partial\mathcal{L}}{\partial\hat{\boldsymbol{x}}_i[t]}\boldsymbol{h}_i\left[t\right]\right). \tag{S7}$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Thus, it satisfies that,

$$\begin{aligned}\phi^2 &= \left\| \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} - 1 \cdot \mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right) \right\|^2 + \mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \boldsymbol{h}_i\left[t\right] \right)^2 k - 2k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \boldsymbol{h}_i\left[t\right] \right)^2 \\ &= \left\| \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} - 1 \cdot \mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right) \right\|^2 - k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \boldsymbol{h}_i\left[t\right] \right)^2 \\ &= \left\| \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right\|^2 - 2k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right)^2 + k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right)^2 - k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \boldsymbol{h}_i\left[t\right] \right)^2 \\ &= \left\| \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right\|^2 - k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right)^2 - k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \boldsymbol{h}_i\left[t\right] \right)^2 . \end{aligned} \tag{S8}$$

So we insert ϕ into the squared Eq. S5 and it gives:

$$\left\| \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_i\left[t\right]} \right\|^2 = \frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]^2}{{\boldsymbol{\sigma}_i}^2} \left(\left\| \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right\|^2 - k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right)^2 - k\mathbb{E}\left(\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \boldsymbol{h}_i\left[t\right] \right)^2 \right). \tag{S9}$$

As the expected values in the Eq. S9 are all squared and $\frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} = \frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{x}_i[t]}$, we have:

$$\left\| \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}_i\left[t\right]} \right\|^2 \leq \frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]^2}{{\boldsymbol{\sigma}_i}^2} \left\| \frac{\partial \mathcal{L}}{\partial \hat{\boldsymbol{x}}_i[t]} \right\|^2 = \frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]^2}{{\boldsymbol{\sigma}_i}^2} \left\| \frac{\partial \tilde{\mathcal{L}}}{\partial \boldsymbol{x}_i[t]} \right\|^2 . \tag{S10}$$

Thus,

$$\left\| \nabla_{\boldsymbol{x}_i[t]} \mathcal{L} \right\| \leqslant \frac{\hat{\boldsymbol{\gamma}}_i\left[t\right]}{\boldsymbol{\sigma}_i} \left\| \nabla_{\boldsymbol{x}_i[t]} \tilde{\mathcal{L}} \right\| . \tag{S11}$$

□

The proof of **Theorem 2** in the main text is given below:

Proof. In the case where TEBN scales the presynaptic inputs of the l^{th} layer by $p^l[t]$ at time-step t, we denote ${\bar{\boldsymbol{x}}_i}^{l-1}\left[t\right]$ as current without scaling. The neuron model in layer l can be expressed as:

$$\boldsymbol{u}^l\left[t\right] = \tau \boldsymbol{u}^l\left[t-1\right]\left(1 - \boldsymbol{o}^l\left[t-1\right]\right) + p^l\left[t\right] \bar{\boldsymbol{x}}^{l-1}\left[t\right], \tag{S12}$$

$$\boldsymbol{o}^l\left[t-1\right] = H\left(\boldsymbol{u}^l\left[t-1\right] - \theta\right), \tag{S13}$$

$$\bar{\boldsymbol{x}}^{l-1}\left[t\right] = \boldsymbol{\gamma} \boldsymbol{h}^{l-1}\left[t\right] + \boldsymbol{\beta}. \tag{S14}$$

According to the former expressions, it follows that:

$$\frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{x}}_i^{l-1}\left[t\right]} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^l\left[t\right]} \frac{\partial \boldsymbol{u}_i^l\left[t\right]}{\partial \bar{\boldsymbol{x}}_i^{l-1}\left[t\right]} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^l\left[t\right]} p^l\left[t\right], \tag{S15}$$

and similarly,

$$\frac{\partial \mathcal{L}}{\partial \bar{\boldsymbol{x}}_i^{l-1}\left[t-1\right]} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^l\left[t-1\right]} p^l\left[t-1\right]. \tag{S16}$$

The relation of the gradient of $\boldsymbol{u}_i^l$ between adjacent time-steps goes like this:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^l\left[t-1\right]} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^l\left[t\right]} \frac{\partial \boldsymbol{u}_i^l\left[t\right]}{\partial \boldsymbol{u}_i^l\left[t-1\right]} + \frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_i^l\left[t-1\right]} \frac{\partial \boldsymbol{o}_i^l\left[t-1\right]}{\partial \boldsymbol{u}_i^l\left[t-1\right]} \\ &= \tau \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i^l\left[t\right]} \left[\left(1 - \boldsymbol{o}_i^l\left[t-1\right]\right) - \boldsymbol{u}_i^l\left[t-1\right] \hbar\left(\boldsymbol{u}_i^l\left[t-1\right]\right)\right] \\ &\quad + \frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_i^l\left[t-1\right]} \hbar\left(\boldsymbol{u}_i^l\left[t-1\right]\right), \end{aligned} \tag{S17}$$

where $\hbar(\cdot)$ is the surrogate function of the Heaviside function. By merging the three equations (Eq. S15,S16,S17), we have:

$$\nabla_{\bar{\boldsymbol{x}}_i^{l-1}[t-1]}\mathcal{L} = \frac{p^l[t-1]}{p^l[t]}\tau\left[\left(1-\boldsymbol{o}_i^l[t-1]\right)-\boldsymbol{u}_i^l[t-1]\,\hbar\left(\boldsymbol{u}_i^l[t-1]\right)\right]\nabla_{\bar{\boldsymbol{x}}_i^{l-1}[t]}\mathcal{L} + p^l[t-1]\,\nabla_{\boldsymbol{o}_i^l[t-1]}\mathcal{L}\cdot\hbar\left(\boldsymbol{u}_i^l[t-1]\right). \tag{S18}$$

Calculate L_2 norm of the both side of Eq. S18, and the following inequality holds:

$$\left\|\nabla_{\bar{\boldsymbol{x}}_i^{l-1}[t-1]}\mathcal{L}\right\| \le \frac{p^l[t-1]}{p^l[t]}\tau\left\|1-\boldsymbol{o}_i^l[t-1]-\boldsymbol{u}_i^l[t-1]\,\hbar\left(\boldsymbol{u}_i^l[t-1]\right)\right\|\left\|\nabla_{\bar{\boldsymbol{x}}_i^{l-1}[t]}\mathcal{L}\right\| + p^l[t-1]\left\|\nabla_{\boldsymbol{o}_i^l[t-1]}\mathcal{L}\cdot\hbar\left(\boldsymbol{u}_i^l[t-1]\right)\right\|. \tag{S19}$$

We examine the term $\left\|1-\boldsymbol{o}_i^l[t-1]-\boldsymbol{u}_i^l[t-1]\,\hbar\left(\boldsymbol{u}_i^l[t-1]\right)\right\|$: $\boldsymbol{o}_i^l[t]$, $\boldsymbol{u}_i^l[t]$, and $\hbar$ all have constraints:

$$0 \le 1-\boldsymbol{o}^l[t-1] \le 1, \tag{S20}$$

$$0 \le \boldsymbol{u}^l[t-1] \le \theta, \tag{S21}$$

$$0 \le \hbar(\boldsymbol{u}_i^l[t-1]) \le \hbar_{\max}. \tag{S22}$$

Thus,

$$\left\|1-\boldsymbol{o}_i^l[t-1]-\boldsymbol{u}_i^l[t-1]\,\hbar\left(\boldsymbol{u}_i^l[t-1]\right)\right\| \le \left\|1-\boldsymbol{o}_i^l[t-1]\right\|+\left\|\boldsymbol{u}_i^l[t-1]\,\hbar\left(\boldsymbol{u}_i^l[t-1]\right)\right\| \le \sqrt{k}\left(1+\theta\hbar_{\max}\right). \tag{S23}$$

where k is the mini-batch size. Hence, we have:

$$\left\|\nabla_{\bar{\boldsymbol{x}}_i^{l-1}[t-1]}\mathcal{L}\right\| \le \frac{p[t-1]}{p[t]}\tau\sqrt{k}\left(1+\theta\hbar_{\max}\right)\left\|\nabla_{\bar{\boldsymbol{x}}_i^{l-1}[t]}\mathcal{L}\right\| + p[t-1]\,\hbar_{\max}\left\|\nabla_{\boldsymbol{o}_i^l[t-1]}\mathcal{L}\right\|. \tag{S24}$$

□

B Analysis on Weight of Presynaptic Inputs

To investigate the necessity of our method, we modify the presynaptic inputs of the first layer in a trained 9-layer CNN model using default BN with 3 time-steps trained on IFAR-10 and analyze the test accuracy. By utilizing rescaling factors between 0.4 and 1.8 with a step of 0.2, we presented the test results in Fig. S1. The standard BN method can be thought of as TEBN with $p^1[t]=1, t=0,1,2$. However, as shown in the figure, the best test accuracy occurs when $p^1[0]=0.6, p^1[1]=1.2, p^1[2]=1.0$. Besides, in many cases, the accuracy is higher than the result with $p^1[t]=1, t=0,1,2$. Our results show that there is a searchable space to find better weights for BN at different time-steps. The performance of the BN method without scaling at different time-steps is bottlenecked. Using proper scaling can continue to improve model performance.

C Distribution of Presynaptic Inputs with Different BN Methods

Using a 9-layer CNN with 3 time-steps on CIFAR-10, the temporal distributions of presynaptic inputs at each layer after training with BNTT, tdBN, and TEBN are shown in Fig. S2. We fix the two axes to obtain a more clear comparison.

D Spiking Response Model Neuron Experiments

We added the experiment of SRM on the CIFAR-10 dataset to validate the generalization of TEBN. For the SRM model, we used standard double-exponential PSP kernels with a brief finite rise and exponential decay, of the form $\epsilon(t)=\frac{\tau_m}{\tau_m-\tau_s}(e^{-\frac{t}{\tau_m}}-e^{-\frac{t}{\tau_s}})$. As shown in Tab. S1, after training for 100 epochs, the SRM-based network gets slighter worse accuracy than the LIF-based network with the same structure, which implies that our method can be generalized to different neuron models.

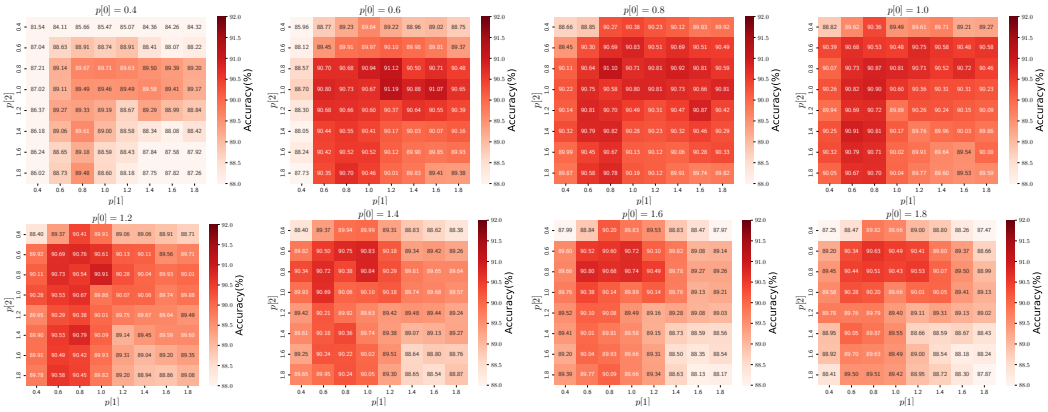

Figure S1: Accuracy heatmap of a 9-layer CNN scaling with different $p[t]$s.

Table S1: Performance on CIFAR-10 dataset with SRM model.

Neuron	Architecture	Accuracy(%)
LIF(τ=0.25)	7-layer CNN	92.57
SRM(τ_s=2,τ_m=4)	7-layer CNN	91.69

E Speech Commands Experiments

In fact, no motion is required for classification tasks. We have added the experiment on the Speech Commands dataset to compare our method with other normalization methods. We apply the 4-layer CNN proposed in [2] and train the network for 50 epochs with 102 time-steps. As reported in Tab. S2, our method achieves better performance than other normalization methods, which implies that our method works for the time-dependent task.

Table S2: Comparison with other normalization methods on Speech Commands dataset.

Model	Accuracy(%)
LN	92.33
BNTT	94.23
Without BN	94.60
BN	95.12
tdBN	95.39
TEBN	**95.47**

F Federated Learning

BNTT like methods can be used for learning in diverse scenarios [5, 11, 6], we believe that our method also can be applied to more diverse learning scenarios. Here we add the experiment of federated learning [11] to validate the generalization of the proposed TEBN. We compare our method with BNTT [7]. For a fair comparison, we use the same VGG-9 [7] structure and only replace BNTT by TEBN. The results are shown in Tab. S3, we obtained final testing accuracy of 85.81%(v.s. 76.44% of BNTT) with 10 clients in total and 2 participating clients.

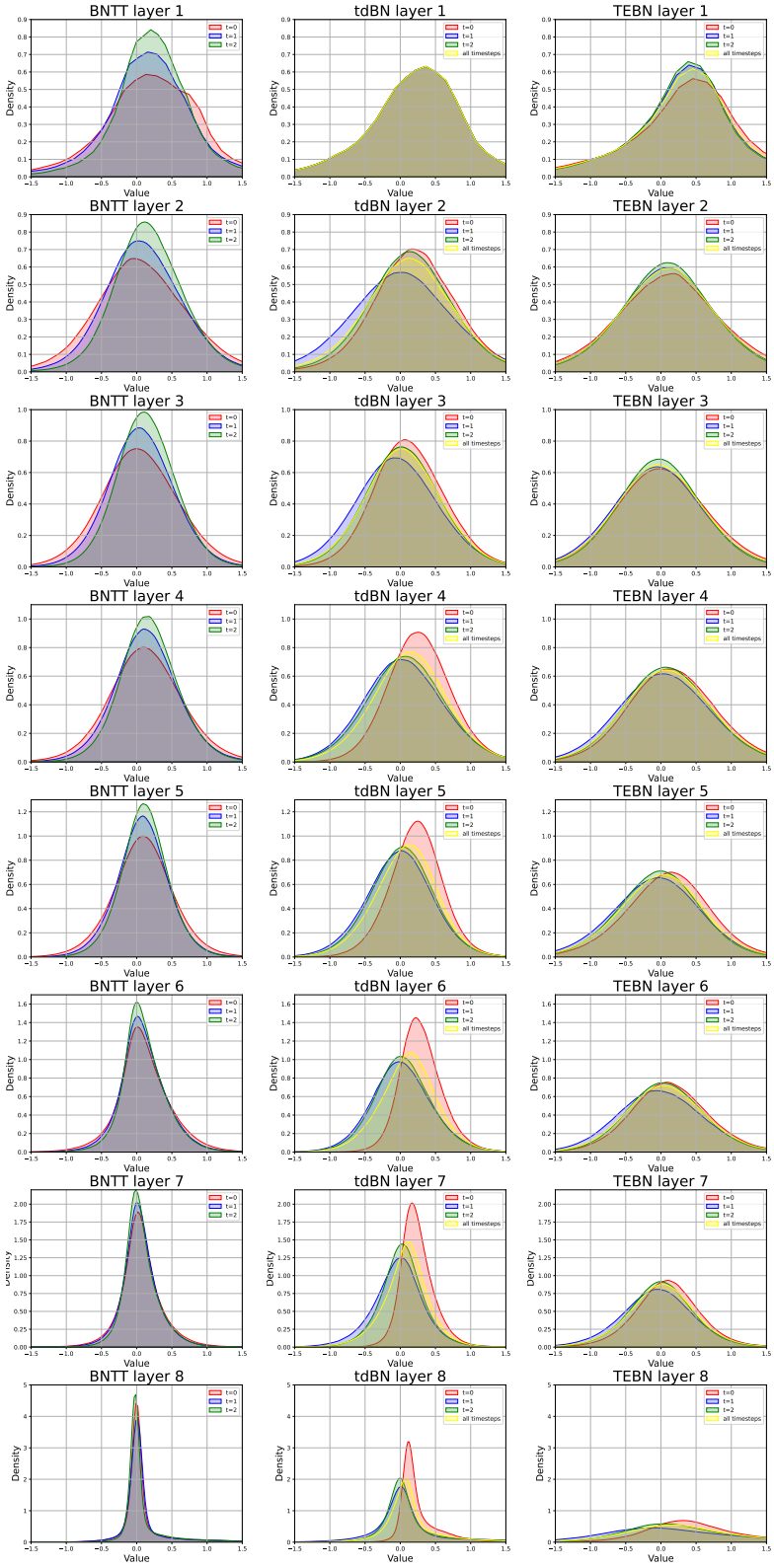

Figure S2: Distributions of presynaptic inputs with different BN methods

Table S3: Comparison on the task of federated learning

Dataset	Model	Methods	Architecture	Time-steps	Accuracy(%)
CIFAR-10	BNTT	Surrogate Gradient	VGG-9	20	76.44
CIFAR-10	**TEBN**	Surrogate Gradient	VGG-9	4	**85.81**

G Experiments Details

G.1 Operating Environments

We train models on the PyTorch platform. Our ResNet-34 and SEW ResNet-34 experiments are conducted on 4 NVIDIA Tesla A100 GPUs, ResNet-19 experiments are conducted on a NVIDIA Tesla V100, and other experiments are conducted on a NVIDIA GTX3080 GPU.

G.2 Network Architectures

We use 9-layer CNN (64C3-128C3S2-256C3-256C3S2-512C3-512C3S2-512C3-512C3S2-10/100FC), ResNet-19 [14], ResNet-34 [14] and SEW ResNet-34 [3] network architecture to conduct experiments. In order to enhance generalization, we add Dropout [10] layers before fully-connected layers. Furthermore, we use a voting layer after the last fully-connected layer in DVS-CIFAR10 experiments as proposed in [4, 12]. In comparison experiments with existing BN methods, we reproduce the same architectures as BNTT [7]: VGG-9 (64C3-64C3-AP2-128C3-128C3-AP2-256C3-256C3-256C3-AP2-1024FC-10FC) on CIFAR-10, VGG-11 (64C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-AP2-512C3-512C3-AP2-4096FC-100FC) on CIFAR-100 and 7-layer CNN (64C3-AP2-128C3-128C3-AP2-256C3-256C3-AP2-1024FC-10FC) on DVS-CIFAR10. Moreover, in order to compare with other state-of-the-art methods, we also reproduce 7-layer CNN[13] (128C3-256C3-AP2-512C3-AP2-1024C3-512C3-1024FC-10FC) on CIFAR-10, 6-layer CNN[4] (128C3-AP2-128C3-AP2-128C3-AP2-128C3-AP2-512FC-10FC) and VGGSNN[1] (64C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-AP2-512C3-512C3-AP2-10FC) on DVS-CIFAR10.

G.3 Data Pre-processing

We apply data normalization to the CIFAR-10 and CIFAR-100 datasets. In addition, we apply random cropping and horizontal flipping for data augmentation. In comparison experiments with state-of-the-art methods, we add cutout operations as TET [1] and Dspike [8] did on ResNet.

Our DVS-CIFAR10 is loaded from the Spikingjelly framework [2], and we separate the dataset into a training set and a test set with 9000 and 1000 sequences, respectively. We reduce the resolution from 128×128 to 48×48. In comparison experiments with state-of-the-art methods, we additionally apply random cropping and horizontal flipping for data augmentation.

As for ImageNet datasets, We crop the images to 224×224 and using the standard augmentation for the training data.

G.4 Training Settings and Hyperparameters

To allow for effective BP training, the firing functionality of the neurons on the last layer L is removed, so that the output neurons only output the temporal average of the presynaptic inputs of all time-steps $(m_1, m_2, ..., m_n)$ where n is the number of classes. The output is then fed into a Softmax layer to form the cross-entropy loss with the true labels $(y_1, y_2, ..., y_n)$:

$$\mathcal{L} = -\sum_{i=1}^{n} y_i \log\left(\frac{e^{m_i}}{\sum_{j=1}^{n} e^{m_j}}\right). \tag{S25}$$

We choose the triangle-shaped surrogate gradients, which can be described as:

$$\frac{\partial \boldsymbol{o}_i[t]}{\partial \boldsymbol{u}_i[t]} = \hbar(\boldsymbol{u}_i[t]) = \max\left(0, 1 - |\boldsymbol{u}_i[t] - \theta|\right). \tag{S26}$$

Other hyperparameters can be found in Tab S4.

Table S4: Hyperparameters optimization for training

Dataset	Optimizer	Scheduler	Epoch	Learning Rate	Batchsize
CIFAR-10	SGD	CosineAnnealingLR(T=200)	200	0.02	64
CIFAR-100	SGD	CosineAnnealingLR(T=200)	200	0.02	64
DVS-CIFAR10	SGD	CosineAnnealingLR(T=200)	200	0.1	32
ImageNet	AdamW	CosineAnnealingLR(T=120)	120	0.01	64

H Limitations and Societal Impact

Although achieving better performance than existing approaches, our method has only been tested on SNNs with surrogate gradients. Since timing-based backpropagation SNNs do not require unfolding the network over time-steps, which brings better asynchronism, we aim to utilize our method in timing-based SNNs in future research. As for societal impacts, our research focuses on the training of high-performance SNNs and thus does not have obvious negative societal impacts. Considering the reduction in energy consumption brought about by the widespread use of SNNs, we believe that SNNs will become indispensable software in edge computing.

References

[1] 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 (ICLR)*, 2021.

[2] Wei Fang, Yanqi Chen, Jianhao Ding, Ding Chen, Zhaofei Yu, Huihui Zhou, Yonghong Tian, and other contributors. Spikingjelly. `https://github.com/fangwei123456/spikingjelly`.

[3] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:21056–21069, 2021.

[4] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothee Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 2661–2671, 2021.

[5] Youngeun Kim, Joshua Chough, and Priyadarshini Panda. Beyond classification: directly training spiking neural networks for semantic segmentation. *arXiv preprint arXiv:2110.07742*, 2021.

[6] Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, and Priyadarshini Panda. Neural architecture search for spiking neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, 2022.

[7] Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 15:773954–773954, 2021.

[8] 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 (NeurIPS)*, 34:23426–23439, 2021.

[9] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2488–2498, 2018.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[11] Yeshwanth Venkatesha, Youngeun Kim, Leandros Tassiulas, and Priyadarshini Panda. Federated learning with spiking neural networks. *IEEE Transactions on Signal Processing*, 69:6183–6194, 2021.

[12] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1311–1318, 2019.

[13] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:12022–12033, 2020.

[14] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence(AAAI)*, pages 11062–11070, 2021.