ONLINE STABILIZATION OF SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks (SNNs), attributed to the binary, event-driven nature of spikes, possess heightened biological plausibility and enhanced energy efficiency on neuromorphic hardware compared to analog neural networks (ANNs). Mainstream SNN training schemes apply backpropagation-through-time (BPTT) with surrogate gradients to replace the non-differentiable spike emitting process during backpropagation. While achieving competitive performance, the requirement for storing intermediate information at all time-steps incurs higher memory consumption and fails to fulfill the online property crucial to biological brains. Our work focuses on online training techniques, aiming for memory efficiency while preserving biological plausibility. The limitation of not having access to future information in early time steps in online training has constrained previous efforts to incorporate advantageous modules such as batch normalization. To address this problem, we propose Online Spiking Renormalization (OSR) to ensure consistent parameters between testing and training, and Online Threshold Stabilizer (OTS) to stabilize neuron firing rates across time steps. Furthermore, we design a novel online approach to compute the variable mean and variance over time for OSR. Experiments conducted on various datasets demonstrate the proposed method's superior performance among SNN online training algorithms.

1 INTRODUCTION

Regarded as the third generation of neural networks, spiking neural networks (SNNs) possess a greater level of biological plausibility (Zenke et al., 2021) than their second generation counterparts – analog neural networks (ANNs) due to the binary and event-driven nature of spikes. The binary nature of spikes in SNNs eliminates the need for multiplication during inference, leading to improved energy efficiency when deployed on neuromorphic hardware (Furber et al., 2014; Merolla et al., 2014; Shen et al., 2016; Davies et al., 2018; Pei et al., 2019). However, the discontinuity of binary spikes also poses challenges in the training of SNNs.

To address the non-differentiable issue associated with the spike emitting process in SNN training, various approaches have been proposed. The mainstream direct training techniques use surrogate gradients to address this problem, which replaces the non-differentiable Heaviside function during the spike firing process with a differentiable surrogate function (Neftci et al., 2019). In addition to this, they just regard SNNs as binary recurrent neural networks (RNNs) and use backpropagation-through-time (BPTT) for SNN training (Bellec et al., 2018; Zenke & Ganguli, 2018; Wu et al., 2018). Although competitive performances are achieved on the CIFAR-10/100 and ImageNet datasets (Deng et al., 2021; Fang et al., 2021) with a relatively short simulation time, these methods require storing intermediate information of all time-steps for gradient backpropagation. An alternative approach to training SNNs is to use the assistance of ANNs. Several works first train ANNs and then convert them to SNNs (Cao et al., 2015; Rueckauer et al., 2017; Han et al., 2020; Bu et al., 2022a; Deng & Gu, 2021; Bu et al., 2022b). However, these methods often require a longer simulation time and result in more fired spikes. The long simulation time will lead to high latency, while more fired spikes will

consume more energy. Overall, these approaches bring about extra expenses either in the training phases or in the testing phases while not satisfying the online property of the learning process in biological brains.

Recently, online training techniques have been developed to save memory costs while maintaining the biologically-plausible online property during the training process. However, the limitation of not having access to future information in the early time steps has constrained previous efforts to incorporate advantageous modules such as batch normalization (BN). In this work, we design a mechanism that bypasses the need for future information while maintaining consistency across time-steps, thereby reducing the overfitting problem associated with BN. Our main contributions can be summarized as follows:

1. We propose Online Spiking Renormalization (OSR), ensuring consistent scale and shift parameters between testing and training. This helps mitigate the difference between the normalization parameters used in standard batch normalization. In addition, we introduce an online approach for computing a variable's all-time mean and variance that dynamically changes over time for OSR.

2. We devise Online Threshold Stabilizer (OTS), aiming at stabilizing neuron firing rates across varying time steps, which also effectively regulates the overall firing rate.

3. We conduct experiments on CIFAR10, CIFAR100, CIFAR10-DVS, DVS-Gesture, and Imagenet datasets and demonstrate that our proposed method achieves state-of-the-art performance among SNN online training algorithms.

2 Related Work

2.1 Online training approaches

Online training allows real-time parameter updates as new data arrives, especially useful for RNNs and SNNs spanning multiple time-steps. This mechanism serves to curtail memory usage, a particularly advantageous feature when dealing with many time-steps.

Existing literature on RNNs has delved into various approaches to online learning. Real-time recurrent learning (RTRL), introduced by Williams & Zipser (1989), propagates partial derivatives of hidden states across parameters throughout time, enabling the computation of gradients in a forward-in-time manner. A number of contemporary research endeavors, exemplified by UORO (Tallec & Ollivier, 2017), KF-RTRL (Mujika et al., 2018), and SnAp (Menick et al., 2020), have explored enhancing the memory and time efficiency of RTRL through tailored approximations for more pragmatic utilization. Another work put forward a proposition to update parameters in an online fashion, utilizing decoupled gradients coupled with regularization at each time-step (Kag & Saligrama, 2021).

In the domain of SNNs, numerous studies have drawn inspiration from online training techniques developed for RNNs. Some of these works adopt the fundamental principles of RTRL and tailor them to streamline the training process for SNNs (Zenke & Ganguli, 2018; Bellec et al., 2020; Bohnstingl et al., 2022). Yin et al. (2022) directly applied the approach proposed by Kag & Saligrama (2021) to train SNNs. Zenke & Ganguli (2018) connected the online learning rule for leaky integrate-and-fire (LIF) neurons with the nonlinear Hebbian three-factor rule, and Kaiser et al. (2020) extended the neuron model to a double-exponential spike-response model. Xiao et al. (2022) successfully extended online training methodologies to accommodate large-scale tasks such as the ImageNet classification. However, all these works did not consider incorporating network modules like batch normalization to enhance the network performance. As a result, they suffer from a performance disadvantage compared to their BPTT counterparts.

2.2 NORMALIZATION MECHANISMS

Normalization mechanisms are commonly used in neural networks to stabilize network training, which speeds up convergence and enhances network performance. Typical normalization techniques include batch normalization (BN) (Ioffe & Szegedy, 2015), instance normalization (IN) (Ulyanov et al., 2016), group normalization (GN) (Wu & He, 2018), and layer normalization (LN) (Ba et al., 2016). A subsequent work, batch renormalization (Ioffe, 2017), improves BN by eliminating the difference between the batch mean and variance between the training and testing phases.

In spiking neural networks, researchers have also tried to incorporate normalization techniques to enhance SNN performance. For instance, Kim & Panda (2021) proposed BNTT to regulate firing rates by utilizing separate BN parameters at different time steps. Zheng et al. (2021) proposed threshold-dependent batch normalization (tdBN), which extends the scope of BN to the additional temporal dimension and takes into account the impact of threshold on firing rates. TEBN (Duan et al., 2022) combined elements from both of these approaches by applying BN across the spatial-temporal dimension, while utilizing separate scale and shift parameters at different time steps. Although most works apply normalization to the input current, some studies explore normalization for other variables. For example, PSP-BN (Ikegawa et al., 2022) used unique statistics, specifically the second raw moment, as the denominator for normalization, which can be inserted right after the spiking functions. This approach leads to a higher complexity of BN parameters and the potential risk of breaking the temporal coherence of information. Among the aforementioned works, the most successful ones (Duan et al., 2022; Zheng et al., 2021) used information from all time-steps for BN. However, these methods cannot be directly applied to online learning.

3 PRELIMINARIES

3.1 LEAKY INTEGRATE AND FIRE NEURON

Spiking neurons are the basic building blocks of SNNs, with the LIF neuron model being the most commonly used (Gerstner et al., 2014). The dynamic of the LIF neuron before firing can be described by:

$$\tau \frac{du(t)}{dt} = -(u(t) - u_{\text{rest}}) + I(t), \tag{1}$$

where $u(t)$ is the membrane potential of the neuron at time t, $I(t)$ is the input current received by the neuron, τ is the membrane time constant, and u_{rest} is the resting potential. When membrane potential $u(t)$ reaches a certain threshold θ, the neuron will emit a spike, and $u(t)$ will be suddenly reset to a value u_{reset}. In practice, we often use a discrete form of Eq. 1, which can be represented as:

$$\boldsymbol{u}^l[t] = (1 - \frac{1}{\tau^l})\boldsymbol{u}^l[t-0.5] + \boldsymbol{W}^l \boldsymbol{s}^{l-1}[t], \tag{2}$$

$$\boldsymbol{s}^l[t] = \Theta(\boldsymbol{u}^l[t] - \theta), \tag{3}$$

$$\boldsymbol{u}^l[t+0.5] = \boldsymbol{u}^l[t] \odot (1 - \boldsymbol{s}^l[t]). \tag{4}$$

Here, we use a tensor form that $\boldsymbol{u}^l$, $\boldsymbol{W}^l$, and $\boldsymbol{s}^l$ denote the membrane potential tensor, weight matrix between layers $l-1$ and l, output spike tensor of layer l, respectively. Among them, $\boldsymbol{u}^l[t]$ is the membrane potential after decay and adding input but before the reset, and $\boldsymbol{u}^l[t+\frac{1}{2}]$ is the membrane potential after reset. Θ is a Heaviside step function. The element of $\boldsymbol{s}^l[t]$ equals 1 if the neuron fires and 0 otherwise.

4 METHODS

Our holistic method is illustrated in Figure 1. In the following parts, we first briefly introduce the forward and backward propagation processes of our algorithm and then elaborate on the modules we add to the network.

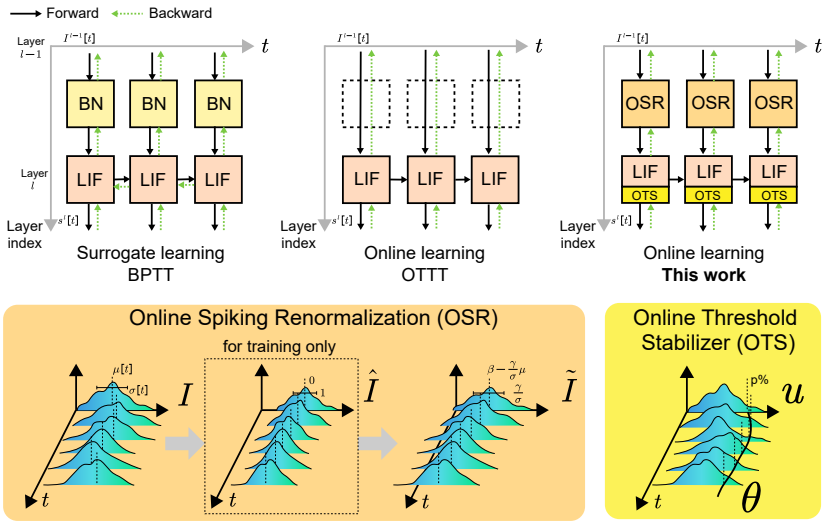


Figure 1: Illustration of online stabilization techniques for SNN. Our method uses online spiking renormalization to improve the generalization and an online threshold stabilizer to regulate the firing rate within the framework of online SNN training, which requires less memory usage than BPTT training. OTTT (Xiao et al., 2022) adopts normalization-free networks and thus has no BN modules.

These modules include online calculation of all-time mean and variance, online spiking renormalization (OSR), and online threshold stabilizer (OTS).

4.1 Forward and Backward Propagation

In the forward stage, our method uses the LIF formulas (Eqs. 2-4). The OSR replaces $s^{l-1}[t]\boldsymbol{W}^l$ with $renorm(s^{l-1}[t]\boldsymbol{W}^l)$ in Eq. 2, and the OTS changes θ in Eq. 3 over time-steps.

In the backward stage, we select the TET loss (Deng et al., 2021) as our loss function since the loss function needs to provide feedback at each time-step:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}\left((1-\epsilon)\sum_{i=1}^{n} y_i \log o_i[t] + \epsilon \sum_{i=1}^{n}(o_i[t]-\phi(y_i))^2\right), \tag{5}$$

where T is the total simulation time, y_i denotes whether the label is equal to i, and $\phi(y_i)$ is the target value of y_i set in MSE loss. $o_i[t]$ is the spikes of output neuron i at time t in the output layer, and ϵ is a small value to introduce MSE loss as a regularization term as proposed by Deng et al. (2021).

For the online gradient propagation, we remove the propagation path of neuron membrane potential decay and reset (from the next time step to the last time step) for membrane potential (as shown in Figure 1). Then the gradients received by membrane potential and weights become:

$$\frac{\partial \mathcal{L}}{\partial u_y^l[t]} = \frac{\partial \mathcal{L}_t}{\partial s_y^l[t]}\frac{\partial s_y^l[t]}{\partial u_y^l[t]}, \tag{6}$$

$$\frac{\partial \mathcal{L}}{\partial w_{xy}^l} = \sum_{t=1}^{T}\frac{\partial \mathcal{L}_t}{\partial s_y^l[t]}\frac{\partial s_y^l[t]}{\partial u_y^l[t]}\frac{\partial u_y^l[t]}{\partial w_{xy}^l}. \tag{7}$$

Note that Eq. 7 just sums up the derivative of $\mathcal{L}_t$ to w^l_{xy} at all time-steps, so there is no backward or forward temporal dependency both in Eqs. 6 and 7)

4.2 Incorporating Batch Normalization into Online Algorithms

In Zheng et al. (2021); Kim & Panda (2021); Duan et al. (2022), it is shown that applying BN once across all time-steps, rather than separately on each time-step, yields superior performance. However, in online training, it is impractical as we need normalization before having information on all time-steps. As per Duan et al. (2022), using mean and variance across all time-steps is crucial for reducing temporal covariate shift and enhancing performance.

Online Calculation of All-time Mean and Variance. As a result, we want to calculate the mean and variance of the input current (denoted as $I_x[t]$ for the x-th dimension at time t) among all time-steps (denoted as μ and σ^2) in training for the *running_mean* and *running_var* (denoted as $\hat{\mu}$ and $\hat{\sigma^2}$, the detailed usages of them are illustrated in the next paragraph, Online Spiking Renormalization). To keep the memory cost low, we have to calculate them from the mean and variance of each time-step: $\mu[1], \cdots, \mu[T]$ and $\sigma^2[1], \cdots, \sigma^2[T]$. Their relationship can be described as the following equation:

$$\mu[t] = \frac{1}{m}\sum_{x=1}^{m} I_x[t], \qquad \sigma^2[t] = \frac{1}{m}\sum_{x=1}^{m}(I_x[t] - \mu[t])^2, \tag{8}$$

$$\mu = \frac{1}{mT}\sum_{t=1}^{T}\sum_{x=1}^{m} I_x[t] = \frac{1}{T}\sum_{t=1}^{T}\mu[t], \tag{9}$$

$$\sigma^2 = \frac{1}{mT}\sum_{t=1}^{T}\sum_{x=1}^{m}(I_x[t] - \mu)^2 = \frac{1}{T}\sum_{t=1}^{T}\sigma^2[t] + \frac{1}{T}\sum_{t=1}^{T}\mu[t]^2 - \mu^2. \tag{10}$$

Hence, we can initialize μ and σ^2 as 0 for each batch, add $\frac{1}{T}\mu[t]$ to μ and add $\frac{1}{T}(\sigma^2[t] + \mu[t]^2)$ at each time step, and subtract μ^2 from σ^2 at the last time step.

Online Spiking Renormalization (OSR). In the testing phase, the BN will give the input current at all time-steps the same linear transformation: subtracting $\hat{\mu}$ (*running_mean*) followed by dividing $\hat{\sigma^2}$ (*running_var*). However, in the training phase, information at different time-steps will go through different linear transformations in the forward stage under the vanilla BN. To eliminate this difference between the training phase and the testing phase, we use the same transformation in the forward stage for the training stage as the testing stage. Specifically, we apply the transformation

$$\tilde{\boldsymbol{I}}[t] = \gamma \cdot \frac{\boldsymbol{I}[t] - \hat{\mu}}{\sqrt{\hat{\sigma^2} + \epsilon}} + \beta \tag{11}$$

during the forward step in training, where $\boldsymbol{I}[t]$ is the input current to be normalized.

The next question is: How to compute gradients in the backward stage if we use this forward transformation? The $\hat{\mu}$ and $\hat{\sigma^2}$ are not the real mean and variance of a batch, and actually they do not come from the current batch data. As a result, if no additional mechanisms are involved, this 'standardization' just plays the role of linear transformation instead of real normalization. Our solution for this is to first go through a real normalization and then apply a linear transform to the normalized tensor. To be specific, we first normalize $\boldsymbol{I}[t]$ to $\hat{\boldsymbol{I}}[t] = \frac{\boldsymbol{I}[t]-\mu[t]}{\sqrt{\sigma^2[t]+\epsilon}}$ and then linearly transform it twice to $\tilde{\boldsymbol{I}}[t] = \gamma \cdot \frac{\boldsymbol{I}[t]-\hat{\mu}}{\sqrt{\hat{\sigma^2}+\epsilon}} + \beta$:

$$\tilde{\boldsymbol{I}}[t] = \gamma \cdot \frac{\boldsymbol{I}[t] - \hat{\mu}}{\sqrt{\hat{\sigma^2} + \epsilon}} + \beta = \gamma \cdot \left(\hat{\boldsymbol{I}}[t] \cdot \frac{\sqrt{\sigma^2[t] + \epsilon}}{\sqrt{\hat{\sigma^2} + \epsilon}} + \frac{\mu[t] - \hat{\mu}}{\sqrt{\hat{\sigma^2} + \epsilon}}\right) + \beta. \tag{12}$$

Eq. 12 denotes the normalization followed by a linear transformation. The gradients for $\boldsymbol{I}[t], \gamma, \beta$ are:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{I}[t]} = \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{I}}[t]} \cdot \frac{\partial \hat{\boldsymbol{I}}[t]}{\partial \boldsymbol{I}[t]} \cdot \gamma \cdot \frac{\sqrt{\sigma^2[t]+\epsilon}}{\sqrt{\hat{\sigma^2}+\epsilon}}, \tag{13}$$

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_x \frac{\partial \mathcal{L}}{\partial \tilde{I}_x[t]} \left(\hat{I}_x[t] \cdot \frac{\sqrt{\sigma^2[t]+\epsilon}}{\sqrt{\hat{\sigma^2}+\epsilon}} + \frac{\mu[t]-\hat{\mu}}{\sqrt{\hat{\sigma^2}+\epsilon}} \right), \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = \sum_x \frac{\partial \mathcal{L}}{\partial \tilde{I}_x[t]}. \tag{15}$$

Online Threshold Stabilizer (OTS). To enhance the stability of mean and variance in the OSR process during training, we introduce the OTS mechanism. The variable subject to normalization is the input current of neurons, and our objective is to ensure the mean and variance of it remain stable across all time-steps. This raises the crucial question: When should we intervene to stabilize the mean and variance of input currents?

The mean and variance of the input current in a layer are significantly influenced by the output spikes from the preceding layer, making it essential to stabilize the firing rate of each layer. The firing rate is determined by the proportion of membrane potential surpassing the firing threshold within discrete time-steps. Consequently, we can adjust either the membrane potential or the firing threshold to regulate the firing rate. Between these options, regulating the firing threshold stands out as a judicious choice: it leaves the neuronal dynamics unchanged and only impacts backward propagation by altering the values of the surrogate function.

Specifically, we assume the membrane potential of neurons in one layer at time t follows a normal distribution $N(\mu_{\text{mem}}[t], \sigma^2_{\text{mem}}[t])$ (where we denote $\theta[t]$, $\mu_{\text{mem}}[t]$, and $\sigma_{\text{mem}}[t]$ are the threshold, mean of membrane potential, and variance of membrane potential at time t), then the firing rate of this layer at time t is

$$1 - \Phi^{-1}\left(\frac{\theta[t] - \mu_{\text{mem}}[t]}{\sigma_{\text{mem}}[t]} \right), \tag{16}$$

where $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{y^2}{2}} dy$ is the cumulative distribution function of normal distribution. To keep this ratio constant among time-steps, we need to keep the quantile $\frac{\theta[t]-\mu_{\text{mem}}[t]}{\sigma_{\text{mem}}[t]}$ constant. Under this control, the adjusted threshold at time t, $\theta[t] = \mu_{\text{mem}}[t] + \sigma_{\text{mem}}[t] \cdot \frac{\theta[1]-\mu_{\text{mem}}[1]}{\sigma_{\text{mem}}[1]}$.

4.3 Theoretical Analysis

In this section, we discuss how our online threshold stabilizer (OTS) helps stabilize online spiking renormalization (OSR). The process involves three stages: adjusting the firing threshold of layer $l-1$; corresponding adjustment of the firing rate of layer $l-1$; and regulating the mean and variance before normalization in layer l to ensure stability. This involves two crucial aspects: adjusting the threshold for a stable firing rate, which subsequently stabilizes the mean and variance. For the step from threshold to firing rate, existing research has shown that when a LIF neuron receives constant input with Gaussian noise, the membrane potential will have a Gaussian distribution (Hohn & Burkitt, 2001). This implies the reasonableness of our Gaussian distribution assumption of membrane potential, further supporting the process of OTS. For the step from firing rate to mean and variance, studying the property of the all-time sample variance σ^2 is a good choice: In Eq. 10, σ^2 can be split into two parts: The first part is $\frac{1}{T}\sum_{t=1}^{T}\sigma^2[t]$, which stands for the average variance inside each time-step. The second part is $\frac{1}{T}\sum_{t=1}^{T}\mu[t]^2 - \mu^2$, which is the variance of the mean at each time-step (variance of $\mu[1], \cdots, \mu[T]$). To stabilize the whole training process, we want the mean among different time-steps to vary as little as possible. In other words, we want the variance of the mean among time-steps to be low. However, for variance inside each time-step, we do not need it to be low.

Denote $p[t]$ to be the firing probability of each neuron at time-step t and gross firing rate $p = \frac{1}{T}\sum_{i=1}^{T} p[t]$. To proceed with the theoretical derivation, we must establish the following assumptions:

Assumption 4.1. Assume all entries of $\boldsymbol{s}^{l-1}[t]$ (of size $B \cdot C_{in}$) and $\boldsymbol{W}^l$ (of size $C_{in} \cdot C_{out}$) are independent for $1 \leq t \leq T$, all $s_i^{l-1}[t]$ obey i.i.d Bernoulli$(p[t])$ distribution, and all w_{ji}^l obey any i.i.d distribution.

Under the above assumptions, we have the following conclusions (note we only discuss the expectation of the target variables since both the sample mean μ and the sample variance σ^2 are estimated statistics):

Theorem 4.2. *When Assumption 4.1 holds and the gross firing rate p holds constant, then the expectation of sample variance of $\mu[t]$ among time-steps $\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mu[t]^2 - \mu^2\right]$ increases when the variance of firing rate among time-steps $\frac{1}{T}\sum_{t=1}^{T} p[t]^2 - p^2$ increases.*

Theorem 4.3. *When Assumption 4.1 holds and the gross firing rate p keeps constant, then the expectation of variance within time-steps $\mathbb{E}[\frac{1}{T}\sum_{t=1}^{T}\sigma^2[t]]$ keeps constant.*

These results indicate that given the gross firing rate (p) constant, reducing the firing probability ($p[t]$) among time-steps will reduce the variance of the mean ($\mu[t]$) among time-steps (Theorem. 4.2) but will not affect the variance inside time-steps ($\sum \sigma^2[t]$) (Theorem. 4.3). Thus, a steady firing rate helps stabilize the sample mean, which further indicates that our OTS mechanism helps our OSR mechanism. Related experimental results are shown in the ablation study in experiments (Table 1).

5 Experiments

To show the effectiveness of our proposed method, we conduct experiments on CIFAR10, CIFAR100 (Krizhevsky et al., 2009), DVS-Gesture (Amir et al., 2017), CIFAR10-DVS (Li et al., 2017), and Imagenet (Deng et al., 2009) datasets to evaluate the performance of our method. The model we choose is consistent with OTTT (Xiao et al., 2022) to conduct a fair comparison. All experiments are run on Nvidia RTX 4090 GPUs with Pytorch 2.0. The implementation details are provided in the Appendix.

5.1 Comparison with other works

Here we compare our approach with previous SNN training methods. We select the BPTT-based algorithms tdBN (Zheng et al., 2021), SEW (Fang et al., 2021), TET (Deng et al., 2021), TEBN (Duan et al., 2022), and an online algorithm OTTT (Xiao et al., 2022). The results have shown that our algorithm performs well on all datasets. For the CIFAR10 dataset, we have achieved similar accuracy with TET and outperformed tdBN and OTTT. For the CIFAR100 dataset, we have outperformed TET and OTTT. For the DVS-Gesture dataset, we have outperformed all listed methods, including tdBN and OTTT. For the CIFAR10-DVS dataset, we have outperformed tdBN and OTTT. For the Imagenet dataset, we have outperformed tdBN and OTTT. Note that the network that OTTT uses (NF-Resnet-34) adds membrane potential in the shortcut connection, which enhances its overall performance over SEW-Resnet-34 and Resnet-34. We test our method for the same architecture (the last line) and achieve better performance with fewer time-steps. Note that among the online algorithms, we have outperformed OTTT on all datasets with fewer time-steps ($T = 4$ vs $T = 6$). In addition, although the overall performance of state-of-the-art BPTT-based algorithms outperforms the online ones in Table 1, they require more memory, especially when the number of total time steps is large (the detailed information is provided in Section 5.2).

Ablation experiments. We also conduct ablation experiments on the Imagenet dataset with SEW-Resnet-34. The ablation involves two of our proposed modules: online spiking renormalization (OSR) and online threshold stabilizer (OTS). When both OTS and OSR are applied, the network's performance has improved by over 3%. However, individually adding these modules (especially OSR) worsens the network performance

Table 1: Performance comparison on CIFAR-10/100, DVS-Gesture, CIFAR10-DVS, and Imagenet

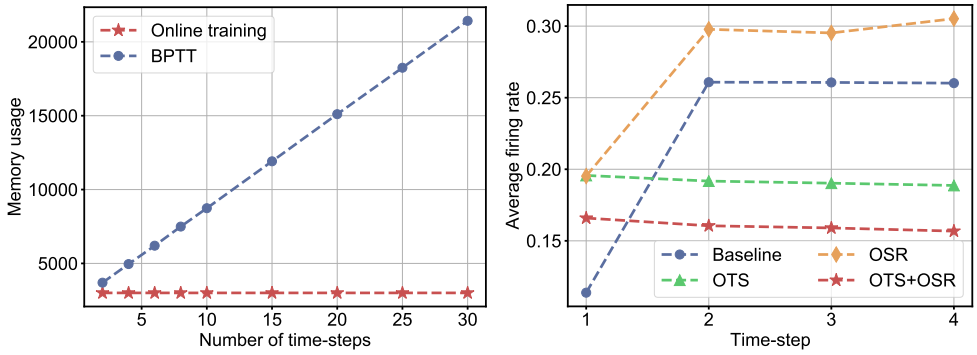
Dataset	Model	Online or not	Architecture	Time steps	Accuracy
CIFAR10	tdBN (Zheng et al., 2021)	✗	Resnet-19	4	92.92%
	TET (Deng et al., 2021)	✗	Resnet-19	4	94.44%
	TEBN (Duan et al., 2022)	✗	Resnet-19	4	95.58%
	OTTT (Xiao et al., 2022)	✔	VGGSNN	6	93.58%
	Ours	✔	VGGSNN	4	94.35%
CIFAR100	TET (Deng et al., 2021)	✗	Resnet-19	4	74.47%
	TEBN (Duan et al., 2022)	✗	Resnet-19	4	78.71%
	OTTT (Xiao et al., 2022)	✔	VGGSNN	6	71.11%
	Ours	✔	VGGSNN	4	76.48%
DVS-Gesture	tdBN (Zheng et al., 2021)	✗	Resnet-17	40	96.88%
	OTTT (Xiao et al., 2022)	✔	VGGSNN	20	96.88%
	Ours	✔	VGGSNN	20	97.57%
CIFAR10-DVS	tdBN (Zheng et al., 2021)	✗	Resnet-19	10	67.80%
	TET (Deng et al., 2021)	✗	VGG-11	10	83.17%
	TEBN (Duan et al., 2022)	✗	VGGSNN	10	84.90%
	OTTT (Xiao et al., 2022)	✔	VGGSNN	10	76.30%
	Ours	✔	VGGSNN	10	82.40%
Imagenet	tdBN (Zheng et al., 2021)	✗	Resnet-34	6	63.72%
	SEW (Fang et al., 2021)	✗	SEW-Resnet-34	4	67.04%
	TET (Deng et al., 2021)	✗	SEW-Resnet-34	4	68.00%
	TEBN (Duan et al., 2022)	✗	SEW-Resnet-34	4	68.28%
	OTTT (Xiao et al., 2022)	✔	NF-Resnet-34	6	65.15%
	BN(Vanilla)	✔	SEW-Resnet-34	4	60.48%
	BN(OSR)	✔	SEW-Resnet-34	4	54.97%
	BN(OTS)	✔	SEW-Resnet-34	4	60.31%
	BN(OSR+OTS)(**Ours**)	✔	SEW-Resnet-34	4	64.14%
	BN(OSR+OTS)(**Ours**)	✔	NF-Resnet-34	4	67.54%

on Imagenet, indicating their complementary nature. A reasonable explanation is that adding OTS alone does not perfectly align the mean and variance of BN between the training and testing phases. Conversely, adding OSR alone leads to uneven firing rates among time-steps, causing instability in OSR's linear transform step.

5.2 Qualitative results

A. Memory Usage: We compare the training memory usage between online algorithms and BPTT algorithms here. We test the case where $T = 2, 4, 6, 8, 10, 15, 20, 25, 30$ on the CIFAR10 dataset with VGGSNN architecture and a batch size of 128. The memory usage statistics are plotted in Figure 2 (a). We can see that our method maintains a constant memory requirement irrespective of time-steps, whereas BPTT approaches scale memory usage linearly with the number of time-steps. In addition, even when the number of time-steps is as low as 2, the memory cost of our algorithm is still lower than that of its BPTT counterpart.

B. Firing Rate Statistics: We compare the firing rate (probability) statistics among different configurations of our proposed modules. We test these statistics on Imagenet, using the SEW-Resnet-34 architecture with total time-steps $T = 4$. The gross firing rate statistics are listed in Table 2 and the per-time-step firing rates



(a) (b)

Figure 2: (a) Comparison of memory usage between our method and BPTT. BPTT incurs memory costs linearly proportional to time-steps, whereas our approach maintains constant memory usage regardless of time-steps. (b) Firing rate statistics of different configurations. From the figures, we know that the online threshold stabilizer indeed stabilizes the firing rate among time-steps.

Table 2: Gross firing rate

Configuration	OTTT (Xiao et al., 2022)	Vanilla BN	OSR	OTS	OSR+OTS
Gross firing rate	24%	22.39%	27.34%	19.16%	16.06%

are plotted in Figure 2 (b). Results have shown that OTS successfully decreases the gross firing rate, which meets our expectations since it raises the thresholds in the latter time-steps. The effect of OSR on firing rates is more interesting: When OTS is not added, it increases the gross firing rate. However, it decreases the total firing rate when OTS is added. For per-time-step firing rates, when the OTS mechanism is not added, the neurons fire far fewer spikes in the first time-step compared with later time-steps, while the firing rate is relatively stable from the second time-step to the last time-step. Besides, our OTS mechanism has not perfectly eliminated the firing rate variation among time-steps. Actually, it slightly over-lifts the firing rate of the first time-step, which might be caused by the firing rate distribution difference among time-steps.

6 Conclusion and Future Work

In this paper, we investigate online training for spiking neural networks, aiming to reduce training memory costs. We integrate essential batch normalization into the online training process by introducing online spiking renormalization and online threshold stabilizers to enhance training stability. Experiments on diverse datasets demonstrate the effectiveness of our proposed modules, showcasing the superior performance of our holistic approach among SNN online training algorithms. However, our approach currently falls short of BPTT in performance, primarily due to the absence of inner-layer and inter-layer reverse-in-time dependencies during backpropagation. Addressing the inner-layer dependency might involve incorporating eligibility traces, but effectively managing the significant inter-layer dependency in online learning remains a challenge. Moreover, achieving biologically plausible learning necessitates local (Journé et al., 2022) and event-driven (Zhu et al., 2022) properties in addition to online behavior—areas we haven't extensively explored in this work. These shortcomings present promising avenues for future research and deeper investigation.

REFERENCES

Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7243–7252, 2017.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in Neural Information Processing Systems*, 31:795–805, 2018.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11(1):3625, December 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-17236-y.

Thomas Bohnstingl, Stanislaw Wozniak, Angeliki Pantazi, and Evangelos Eleftheriou. Online Spatio-Temporal Learning in Deep Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2022.3153985.

Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *In Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11–20, 2022a.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *International Conference on Learning Representations*, 2022b.

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

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. ISSN 0272-1732. doi: 10.1109/MM.2018.112130359.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. In *International Conference on Learning Representations*, 2021.

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*, 2021.

Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 34377–34390. Curran Associates, Inc., 2022.

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*, 34:21056–21069, 2021.

Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13558–13567, 2020.

Nicolas Hohn and Anthony N Burkitt. Shot noise in the leaky integrate-and-fire neuron. *Physical Review E*, 63(3):031902, 2001.

Shin-ichi Ikegawa, Ryuji Saiin, Yoshihide Sawada, and Naotake Natori. Rethinking the role of normalization and residual blocks for spiking neural networks, March 2022.

Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. *Advances in neural information processing systems*, 30, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Adrien Journé, Hector Garcia Rodriguez, Qinghai Guo, and Timoleon Moraitis. Hebbian deep learning without feedback. In *The Eleventh International Conference on Learning Representations*, 2022.

Anil Kag and Venkatesh Saligrama. Training recurrent neural networks via forward propagation through time. In *International Conference on Machine Learning*, pp. 5189–5200. PMLR, 2021.

Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

Youngeun Kim and Priyadarshini Panda. Revisiting Batch Normalization for Training Low-Latency Deep Spiking Neural Networks From Scratch. *Frontiers in Neuroscience*, 15:773954, December 2021. ISSN 1662-453X. doi: 10.3389/fnins.2021.773954.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

Yuhang Li, Youngeun Kim, Hyoungseob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. In *European Conference on Computer Vision*, pp. 631–649. Springer, 2022.

Jacob Menick, Erich Elsen, Utku Evci, Simon Osindero, Karen Simonyan, and Alex Graves. Practical real time recurrent learning with a sparse approximation. In *International conference on learning representations*, 2020.

Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random kronecker factors. *Advances in Neural Information Processing Systems*, 31, 2018.

Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019. ISSN 1053-5888, 1558-0792. doi: 10.1109/MSP.2019.2931595.

Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572 (7767):106–111, 2019.

Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

Juncheng Shen, De Ma, Zonghua Gu, Ming Zhang, Xiaolei Zhu, Xiaoqiang Xu, Qi Xu, Yangjing Shen, and Gang Pan. Darwin: a neuromorphic hardware co-processor based on spiking neural networks. *Science China Information Sciences*, 59(2):1–5, 2016.

Corentin Tallec and Yann Ollivier. Unbiased Online Recurrent Optimization, May 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, June 1989. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco.1989.1.2.270.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3–19, 2018.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online Training Through Time for Spiking Neural Networks, October 2022.

Bojian Yin, Federico Corradi, and Sander M. Bohte. Accurate online training of dynamical spiking neural networks through Forward Propagation Through Time, November 2022.

Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, June 2018. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco_a_01086.

Friedemann Zenke, Sander M Bohté, Claudia Clopath, Iulia M Comşa, Julian Göltz, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O Neftci, Mihai A Petrovici, et al. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4):571–575, 2021.

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*, volume 35, pp. 11062–11070, 2021.

Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. *Advances in Neural Information Processing Systems*, 2022.

A THEORETICAL DERIVATION

To make it easy to follow, We rewrite Assumption. 4.1 as follows:

Assumption A.1. Assume all entries of $\boldsymbol{s}^{l-1}[t]$ (of size $B \cdot C_{in}$) and $\boldsymbol{W}^l$ (of size $C_{in} \cdot C_{out}$) are independent for $1 \leq t \leq T$, all $s_i^{l-1}[t]$ obey i.i.d Bernoulli($p[t]$) distribution, and all w_{ji}^l obey any i.i.d distribution.

For simplicity, we omit the superscript l and $l-1$ in the following derivation. Before proving the theorems, we derive the mean and variance of variable $I_{bi}[t]$ in Lemma. A.2 and Lemma. A.3:

Lemma A.2. *When Assumption 4.1 holds, all $I_{bi}[t]$ will share the identical distribution, and $\mathbb{E}[I_{bi}[t]] = C_{in}p[t]\mathbb{E}[w_{ji}]$, $\mathbb{VAR}[I_{bi}[t]] = C_{in}(p[t]\mathbb{VAR}[w_{ji}] + (p[t] - p[t]^2)\mathbb{E}^2[w_{ji}])$.*

Proof. Since $I_{bi}[t] = \sum_{j=1}^{C_{in}} s_{bj}[t]w_{ji}$ are all sum of products of independent variables with identical distributions ($s_{bj}[t]$ and w_{ji}), they share the identical distribution. We calculate the mean and variance of $I_{bi}[t]$ as follows:

$$\mathbb{E}[I_{bi}[t]] = \sum_{j=1}^{C_{in}} \mathbb{E}(s_{bj}[t])\mathbb{E}(w_{ji}) = C_{in}p[t]\mathbb{E}[w_{ji}] \tag{17}$$

$$\begin{aligned}
\mathbb{E}[I_{bi}^2[t]] &= \mathbb{E}\left[\left(\sum_{j=1}^{C_{in}} s_{bj}[t]w_{ji}\right)^2\right] = \mathbb{E}\left[\sum_{j=1}^{C_{in}} \left(s_{bj}[t]w_{ji}\right)^2\right] + C_{in}(C_{in}-1)p[t]^2\mathbb{E}^2[w_{ji}] \\
&= \sum_{j=1}^{C_{in}} \mathbb{E}[s_{bj}[t]^2]\mathbb{E}[w_{ji}^2] + C_{in}(C_{in}-1)p[t]^2\mathbb{E}^2[w_{ji}] \\
&= C_{in}p[t]\mathbb{E}[w_{ji}^2] + C_{in}(C_{in}-1)p[t]^2\mathbb{E}^2[w_{ji}] \qquad (18) \\
\mathbb{VAR}(I_{bi}[t]) &= \mathbb{E}[I_{bi}^2[t]] - \mathbb{E}^2[I_{bi}[t]] = C_{in}(p[t]\mathbb{E}[w_{ji}^2] - p[t]^2\mathbb{E}^2[w_{ji}]) \\
&= C_{in}(p[t]\mathbb{VAR}[w_{ji}] + (p[t] - p[t]^2)\mathbb{E}^2[w_{ji}]) \qquad (19)
\end{aligned}$$

□

Lemma A.3. *When Assumption 4.1 holds, for all $1 \leq b_1, b_2 \leq B$, $1 \leq i_1, i_2 \leq C_{out}$, and $1 \leq t_1, t_2 \leq T$, $I_{b_1 i_1}[t_1]$ and $I_{b_2 i_2}[t_2]$ are uncorrelated when $(b_1, t_1) \neq (b_2, t_2)$ and $i_1 \neq i_2$. When $(b_1, t_1) = (b_2, t_2)$ and $i_1 \neq i_2$, $\mathbb{COV}(I_{bi_1}[t], I_{bi_2}[t]) = C_{in}\mathbb{E}^2[w_{ji}](p[t] - p[t]^2)$; When $(b_1, t_1) \neq (b_2, t_2)$ and $i_1 = i_2$, $\mathbb{COV}(I_{b_1 i}[t_1], I_{b_2 i}[t_2]) = C_{in}p[t_1]p[t_2]\mathbb{VAR}[w_{ji}]$.*

Proof. Since $I_{bi}[t] = \sum_{j=1}^{C_{in}} s_{bj}[t]w_{ji}$,

$$\mathbb{COV}(I_{b_1 i_1}[t_1], I_{b_2 i_2}[t_2]) = \mathbb{COV}(\sum_{j=1}^{C_{in}} s_{b_1 j}[t_1]w_{ji_1}, \sum_{j=1}^{C_{in}} s_{b_2 j}[t_2]w_{ji_2}) \tag{20}$$

When $(b_1, t_1) \neq (b_2, t_2)$ and $i_1 \neq i_2$, the lemma is trivial since the entries in the summation are all uncorrelated. For the case when $(b_1, t_1) = (b_2, t_2)$, we have:

$$\begin{aligned}
\mathbb{COV}(I_{bi_1}[t], I_{bi_2}[t]) &= \mathbb{E}\left[\left(\sum_{j=1}^{C_{in}} s_{bj}[t] w_{ji_1}\right)\left(\sum_{j=1}^{C_{in}} s_{bj}[t] w_{ji_2})\right)\right] - \mathbb{E}[I_{bi_1}[t]]\mathbb{E}[I_{bi_2}[t]] \\
&= \sum_{j=1}^{C_{in}} \Big(\mathbb{E}(s_{bj}[t]^2 w_{ji_1} w_{ji_2}) - \mathbb{E}(s_{bj}[t] w_{ji_1})\mathbb{E}(s_{bj}[t] w_{ji_2})\Big) \\
&= \sum_{j=1}^{C_{in}} \Big(p[t]\mathbb{E}^2[w_{ji_1}] - p[t]^2\mathbb{E}^2[w_{ji_1}]\Big) \\
&= C_{in}\mathbb{E}^2[w_{ji}](p[t] - p[t]^2)
\end{aligned} \tag{21}$$

For the case when $i_1 = i_2$, we have:

$$\begin{aligned}
\mathbb{COV}(I_{b_1 i}[t_1], I_{b_2 i}[t_2]) &= \mathbb{E}\left[\left(\sum_{j=1}^{C_{in}} s_{b_1 j}[t_1] w_{ji}\right)\left(\sum_{j=1}^{C_{in}} s_{b_2 j}[t_2] w_{ji})\right)\right] - \mathbb{E}[I_{b_1 i}[t_1]]\mathbb{E}[I_{b_2 i}[t_2]] \\
&= \sum_{j=1}^{C_{in}} \Big(\mathbb{E}(s_{b_1 j}[t_1] s_{b_2 j}[t_2] w_{ji}^2) - \mathbb{E}(s_{b_1 j}[t_1] w_{ji})\mathbb{E}(s_{b_2 j}[t_2] w_{ji})\Big) \\
&= \sum_{j=1}^{C_{in}} \Big(p[t]^2\mathbb{E}[w_{ji}^2] - p[t_1]p[t_2]\mathbb{E}^2[w_{ji}]\Big) \\
&= C_{in} p[t_1] p[t_2] \mathbb{VAR}[w_{ji}]
\end{aligned} \tag{22}$$

□

After getting the mean, variance, and covariance of $I_{bi}[t]$, we can prove the following theorems by calculating the coefficient before the variance of $p[t]$:

Theorem A.4. *When Assumption 4.1 holds and the gross firing rate p holds constant, then the expectation of sample variance of $\mu[t]$ among time-steps $\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mu[t]^2 - \mu^2\right]$ increases when the variance of firing rate among time-steps $\frac{1}{T}\sum_{t=1}^{T} p[t]^2 - p^2$ increases.*

Proof. Here we omit the subscript b (the batch dimension) for $I_{bi}[t]$. First we calculate the expectation of $\mu[t]$ and μ:

$$\mathbb{E}[\mu[t]] = \mathbb{E}[I_{bi}[t]] = C_{in} p[t] \mathbb{E}[w_{ji}] \tag{23}$$

$$\mathbb{E}[\mu] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\mu[t]] = C_{in} p \mathbb{E}[w_{ji}] \tag{24}$$

Then we calculate the second moment, including $\mathbb{E}[\mu[t]^2]$ and $\mathbb{E}[\mu[t_1]\mu[t_2]]$:

$$
\begin{aligned}
&\mathbb{E}[\mu[t]^2] = \mathbb{E}^2[\mu[t]] + \mathbb{VAR}[\mu[t]] = C_{in}^2 p[t]^2 \mathbb{E}^2[w_{ji}] + \frac{1}{C_{out}^2}\mathbb{VAR}\left(\sum_{i=1}^{C_{out}} I_i[t]\right) \\
=&C_{in}^2 p[t]^2 \mathbb{E}^2[w_{ji}] + \frac{1}{C_{out}^2}\left(\sum_{i=1}^{C_{out}} \mathbb{VAR}(I_i[t]) + 2\sum_{1\le i_1<i_2\le C_{out}} \mathbb{COV}(I_{i1}[t], I_{i2}[t])\right) \\
=&C_{in}^2 p[t]^2 \mathbb{E}^2[w_{ji}] + \frac{1}{C_{out}^2}\left(C_{out}C_{in}(p[t]\mathbb{E}[w_{ji}^2] - p[t]^2\mathbb{E}^2[w_{ji}]) + C_{out}(C_{out}-1)C_{in}\mathbb{E}^2[w_{ji}](p[t]-p[t]^2)\right) \\
=&C_{in}^2 p[t]^2 \mathbb{E}^2[w_{ji}] + \frac{C_{in}}{C_{out}}\left(p[t]\mathbb{VAR}[w_{ji}] + C_{out}(p[t]-p[t]^2)\mathbb{E}^2[w_{ji}]\right). \qquad (25)
\end{aligned}
$$

$$
\begin{aligned}
&\mathbb{E}[\mu[t_1]\mu[t_2]] = \mathbb{E}[\mu[t_1]]\mathbb{E}[\mu[t_2]] + \mathbb{COV}[\mu[t_1], \mu[t_2]] \\
=&C_{in}^2 p[t_1]p[t_2]\mathbb{E}^2[w_{ji}] + \frac{1}{C_{out}^2}\mathbb{COV}\left(\sum_{i=1}^{C_{out}} I_i[t_1], \sum_{i=1}^{C_{out}} I_i[t_2]\right) \\
=&C_{in}^2 p[t_1]p[t_2]\mathbb{E}^2[w_{ji}] + \frac{1}{C_{out}^2}\sum_{i=1}^{C_{out}} \mathbb{COV}\left(I_i[t_1], I_i[t_2]\right) \\
=&C_{in}^2 p[t_1]p[t_2]\mathbb{E}^2[w_{ji}] + \frac{C_{in}}{C_{out}} p[t_1]p[t_2]\mathbb{VAR}[w_{ji}]. \qquad (26)
\end{aligned}
$$

Finally, we can calculate the target function:

$$
\begin{aligned}
\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mu[t]^2 - \mu^2\right] =&\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\mu[t]^2 - \left(\frac{1}{T}\sum_{t=1}^{T}\mu[t]\right)^2\right] \\
=&\frac{T-1}{T^2}\sum_{t=1}^{T}\mathbb{E}[\mu[t]^2] - \frac{2}{T^2}\sum_{1\le t_1<t_2\le T}\mathbb{E}[\mu[t_1]\mu[t_2]] \\
=&\frac{T-1}{T^2}\sum_{t=1}^{T}\left(C_{in}^2 p[t]^2\mathbb{E}^2[w_{ji}] + \frac{C_{in}}{C_{out}}\left(p[t]\mathbb{VAR}[w_{ji}] + C_{out}(p[t]-p[t]^2)\mathbb{E}^2[w_{ji}]\right)\right) \\
&- \frac{2}{T^2}\sum_{1\le t_1<t_2\le T}\left(C_{in}^2 p[t_1]p[t_2]\mathbb{E}^2[w_{ji}] + \frac{C_{in}}{C_{out}}p[t_1]p[t_2]\mathbb{VAR}[w_{ji}]\right) \\
=&C_{in}^2\mathbb{E}^2[w_{ji}]\left(\frac{1}{T}\sum_{t=1}^{T}p[t]^2 - p^2\right) + C_{in}\mathbb{E}^2[w_{ji}]\left(\frac{T-1}{T}p - \frac{T-1}{T^2}\sum_{t=1}^{T}p[t]^2\right) \\
&+ \frac{C_{in}}{C_{out}}\mathbb{VAR}[w_{ji}]\left(\frac{T-1}{T}p + \frac{1}{T^2}\sum_{t=1}^{T}p[t]^2 - p^2\right) \qquad (27)
\end{aligned}
$$

When p is constant, the variance among time-steps only depends on $\sum_{t=1}^{T} p[t]^2$. In the last equation of Eq. 27, the only thing that can vary is $\sum_{t=1}^{T} p[t]^2$, and the coefficient in front of it is always positive ($C_{in}^2\mathbb{E}^2[w_{ji}] \cdot \frac{1}{T} \ge C_{in}\mathbb{E}^2[w_{ji}] \cdot \frac{1}{T} \ge C_{in}\mathbb{E}^2[w_{ji}] \cdot \frac{T-1}{T^2}$). Therefore, the conclusion holds. □

Theorem A.5. *When Assumption 4.1 holds and the gross firing rate p keeps constant, then the expectation of variance within time-steps $\mathbb{E}[\frac{1}{T}\sum_{t=1}^{T}\sigma^2[t]]$ keeps constant.*

Proof. We first calculate $\mathbb{E}[\sigma^2[t]]$ and then sum them up. The $\mathbb{E}[\sigma^2[t]]$ can be split into calculating $\mathbb{E}[I_i[t]^2]$ and $\mathbb{E}[\mu[t]^2]$, which have been calculated before.

$$\mathbb{E}[\sigma^2[t]] = \mathbb{E}\left[\frac{1}{C_{out}}\sum_{i=1}^{C_{out}} I_i[t]^2 - \mu[t]^2\right] = \frac{1}{C_{out}}\sum_{i=1}^{C_{out}} \mathbb{E}[I_i[t]^2] - \mathbb{E}[\mu[t]^2]$$

$$=C_{in}p[t]\mathbb{E}[w_{ji}^2] + C_{in}(C_{in}-1)p[t]^2\mathbb{E}^2[w_{ji}] - C_{in}^2 p[t]^2\mathbb{E}^2[w_{ji}] - \frac{C_{in}}{C_{out}}\Big(p[t]\mathbb{VAR}[w_{ji}] + C_{out}(p[t]-p[t]^2)\mathbb{E}^2[w_{ji}]\Big)$$

$$=C_{in}p[t]\mathbb{E}[w_{ji}^2] - C_{in}p[t]^2\mathbb{E}^2[w_{ji}] - \frac{C_{in}}{C_{out}}\Big(p[t]\mathbb{VAR}[w_{ji}] + C_{out}(p[t]-p[t]^2)\mathbb{E}^2[w_{ji}]\Big)$$

$$=C_{in}(p[t]\mathbb{VAR}[w_{ji}] + (p[t]-p[t]^2)\mathbb{E}^2[w_{ji}]) - \frac{C_{in}}{C_{out}}\Big(p[t]\mathbb{VAR}[w_{ji}] + C_{out}(p[t]-p[t]^2)\mathbb{E}^2[w_{ji}]\Big)$$

$$=\frac{C_{in}(C_{out}-1)}{C_{out}}p[t]\mathbb{VAR}[w_{ji}] \tag{28}$$

$$\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\sigma^2[t]\right] = \frac{1}{T}\sum_{t=1}^{T}\frac{C_{in}(C_{out}-1)}{C_{out}}p[t]\mathbb{VAR}[w_{ji}] = \frac{C_{in}(C_{out}-1)}{C_{out}}\cdot p\cdot \mathbb{VAR}[w_{ji}] \tag{29}$$

As a result, the variance of $p[t]$ will not affect $\mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T}\sigma^2[t]\right]$, which means it keeps constant. □

B Implementation Details

We conduct experiments on CIFAR10, CIFAR100, DVS-Gesture, CIFAR10DVS, and Imagenet datasets. The network structure of VGGSNN we use for the CIFAR10, CIFAR100, DVS-Gesture, CIFAR10-DVS datasets is consistent with OTTT (64C3-128C3-AP2-256C3-256C3-AP2-512C3-512C3-AP2-512C3-512C3-GAP-FC), where 64C3 denotes convolution layer with 3×3 convolution kernel and 64 output channels, AP2 means 2×2 average pooling, GAP means global average pooling, and FC means fully connected layer. For Imagenet classification, we just use standard Resnet-34 architecture.

In all experiments, we use an SGD optimizer with a momentum of 0.9 with a cosine annealing scheduler. The data augmentation we use for each dataset is listed as follows: For CIFAR10 and CIFAR100, we use RandomCrop(4) + Cutout() + RandomHorizontalFlip() + Normalize(); For DVS-Gesture, we use RandomResizedCrop(128, scale=(0.7, 1.0)) + Resize(48) + RandomRotation(20) + RandomTemporalDelete(14) (recall the total time-step is 20, and the random temporal delete drops 6 time-steps (30%)). For CIFAR10-DVS, we use the neuromorphic data augmentation (NDA) which comes from (Li et al., 2022). For Imagenet, we use RandomResizedCrop(224) + RandomHorizontalFlip() + Normalize() during training. During testing, the image is first resized to 256×256 and center-cropped to 224×224 and then normalized. Other hyperparameters we use are provided in Table 3, including total training epochs, batch size, learning rate, weight decay, weight of MSE loss in Eq. 5 ϵ, and dropout rate.

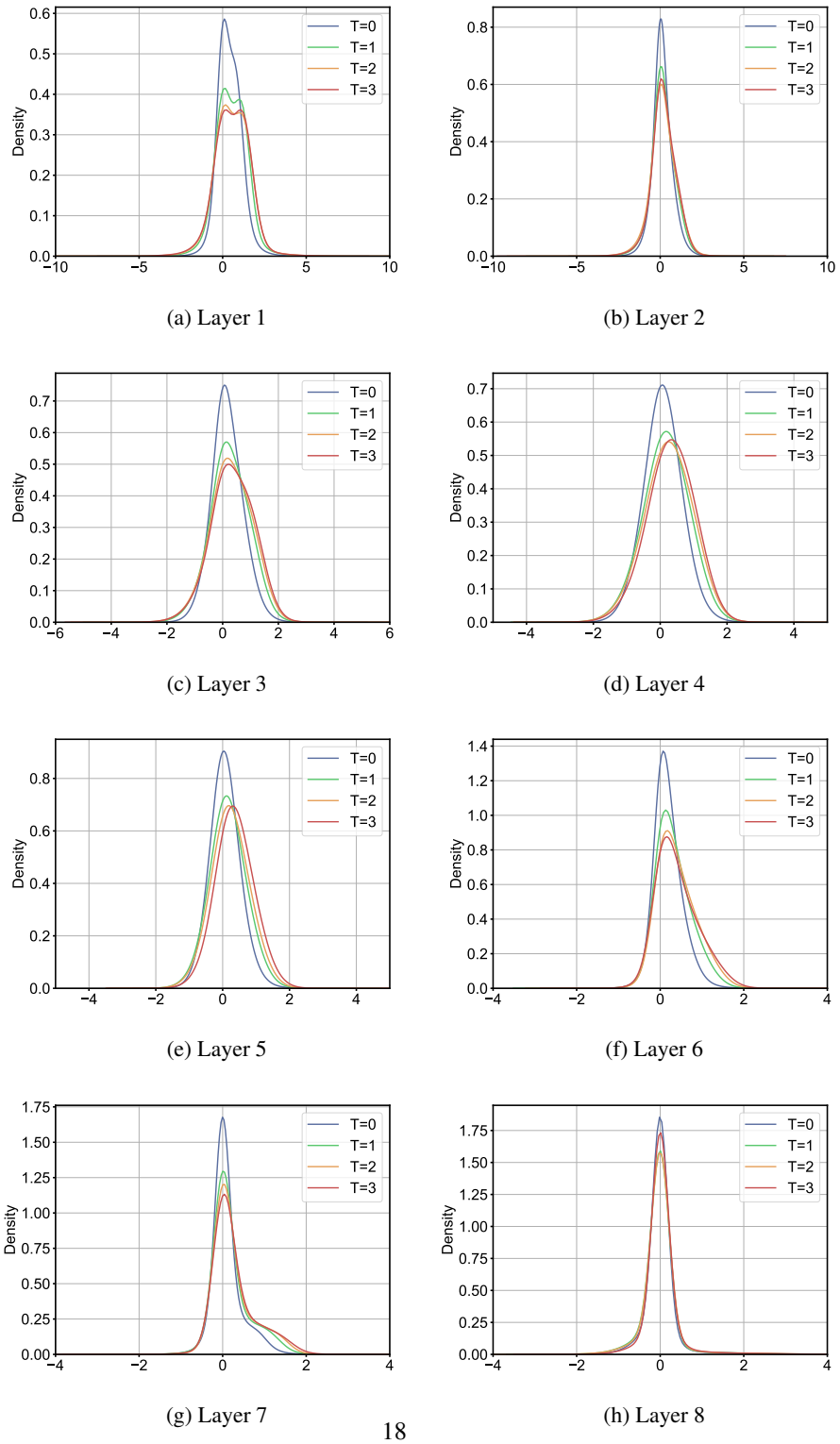
C Membrane Potential Visualization

To see whether the Gaussian assumption in OTS is reasonable, we collect the membrane potential of a VGG network trained with OSR and OTS on the CIFAR-10 dataset and visualize the distribution of membrane

Table 3: Experimental configurations

	CIFAR10	CIFAR100	DVS-Gesture	CIFAR10-DVS	Imagenet
Epochs	300	300	300	300	100
Batch size	128	128	128	128	256
Learning rate	0.1	0.1	0.01	0.1	0.1
Weight decay	5e-4	5e-4	5e-4	5e-4	2e-5
MSE weight ϵ	0.05	0.05	0.001	0.001	0.05
Dropout rate	0	0	0.05	0.1	0

potentials for each layer and each time step. The results are shown in Fig. 3. These distributions display the shape of bell curves, which indicate the similarity between these distributions and Gaussian distributions. Most of the distributions take the mean value around zero. Our proposed algorithm can exploit the adaptation of batch normalization and can cope with the varied distributions during online learning of spiking neural networks.



(a) Layer 1 (b) Layer 2

(c) Layer 3 (d) Layer 4

(e) Layer 5 (f) Layer 6

(g) Layer 7 (h) Layer 8

Figure 3: Visualization of the distributions of membrane potentials for each layer and each time step.