TAB: TEMPORAL ACCUMULATED BATCH NORMALIZATION IN SPIKING NEURAL NETWORKS

Haiyan Jiang[1] **Vincent Zoonekynd**[2] **Giulia De Masi**[3,4] **Bin Gu**[1,6*] **Huan Xiong**[1,5*]
[1]Mohamed bin Zayed University of Artificial Intelligence, UAE
[2]Abu Dhabi Investment Authority, UAE [3]Technology Innovation Institute, UAE
[4]Sant'Anna School of Advanced Studies, Italy [5]Harbin Institute of Technology, China
[6]School of Artificial Intelligence, Jilin University, China
haiyan.jiang@mbzuai.ac.ae vincent.zoonekynd@adia.ae
giuliademasi@gmail.com {bin.gu, huan.xiong}@mbzuai.ac.ae

ABSTRACT

Spiking Neural Networks (SNNs) are attracting growing interest for their energy-efficient computing when implemented on neuromorphic hardware. However, directly training SNNs, even adopting batch normalization (BN), is highly challenging due to their non-differentiable activation function and the temporally delayed accumulation of outputs over time. For SNN training, this temporal accumulation gives rise to Temporal Covariate Shifts (TCS) along the temporal dimension, a phenomenon that would become increasingly pronounced with layer-wise computations across multiple layers and multiple time-steps. In this paper, we introduce TAB (Temporal Accumulated Batch Normalization), a novel SNN batch normalization method that addresses the temporal covariate shift issue by aligning with neuron dynamics (specifically the accumulated membrane potential) and utilizing temporal accumulated statistics for data normalization. Within its framework, TAB effectively encapsulates the historical temporal dependencies that underlie the membrane potential accumulation process, thereby establishing a natural connection between neuron dynamics and TAB batch normalization. Experimental results on CIFAR-10, CIFAR-100, and DVS-CIFAR10 show that our TAB method outperforms other state-of-the-art methods.

1 INTRODUCTION

Spiking Neural Networks (SNNs) are known to be biologically inspired artificial neural networks (ANNs) and have recently attracted great research interest (Chowdhury et al., 2022; Ding et al., 2022). The attraction of SNNs lies in their ability to deliver energy-efficient and fast-inference computations when implemented on neuromorphic hardware such as Loihi (Davies et al., 2018) and TrueNorth (Akopyan et al., 2015; DeBole et al., 2019). These advantages arise from the fact that SNNs utilize spikes to transmit information between layers, whereby the networks circumvent multiplication during inference (Roy et al., 2019). However, the discrete and non-differentiable nature of the binary firing functions makes it difficult to directly train deep SNNs. ANN-to-SNN conversion (Diehl et al., 2015; Bu et al., 2022; Jiang et al., 2023) and directly training with surrogate gradients back-propagation (Neftci et al., 2019; Deng et al., 2022; 2023) are two typical solutions.

Batch Normalization (BN) has found extensive use in ANNs and has seen tremendous success in boosting their performance by reducing the internal covariate shift (ICS) and flattening the loss landscape (Ioffe & Szegedy, 2015; Santurkar et al., 2018). In ANNs, ICS refers to changes in the distribution of layer inputs caused by updates of preceding layers, while in SNNs, the *Temporal Covariate Shift (TCS) phenomenon* (Duan et al., 2022) has been identified due to updates of preceding layers and prior time-steps, which transpires along the additional temporal dimension. Within SNNs, synaptic currents are sequentially fed into spiking neurons, with spike-triggered asynchronous currents accumulating in the membrane potential. Whenever this accumulated membrane potential exceeds a threshold, a spike is generated. This *temporal dependency* on membrane accumulation

*Corresponding authors. Codes are available at https://github.com/HaiyanJiang/SNN-TAB.

has the potential to amplify the internal covariate shift across the temporal domain. The intertwining of this *temporal dependency with the TCS phenomenon*, presents a significant challenge in direct training of SNNs especially for the integration of BN techniques into SNNs.

When it comes to BN techniques for SNNs, only a few methods have been proposed. These methods either normalize data jointly by aggregating data across the temporal dimension or perform independent normalization at each discrete time-step. For example, Kim & Panda (2021) conducts independent batch normalization separately at each time-step. However, this approach uses separate sets of mean, variance, and scale and shift parameters at each time-step, failing to account for the temporal dependencies of the input spikes. While Zheng et al. (2021) merges the data along the time dimension and utilizes shared batch statistics across all time-steps for normalization. Nonetheless, introducing such overall statistics may limit the flexibility to capture varying temporal characteristics at different time-steps. On the other hand, Duan et al. (2022) attempts to tackle the TCS issue by assigning different weights to each time-step, while still utilizing shared batch statistics across all time-steps for normalization. Although these methods improve upon the performance of the SNN models, they do not significantly address the alignment with the neuron dynamics, i.e., the membrane accumulation dependency, or provide a potential to do so.

In this paper, we propose TAB (Temporal Accumulated Batch Normalization) as a solution to effectively address these challenges by closely aligning with the neuron dynamics, specifically the accumulated membrane potential, and providing more accurate batch statistics. This alignment establishes a natural connection between neuronal dynamics and batch normalization in SNNs. Neuron dynamics refer to the changes in the membrane potential of a neuron over time as it integrates input signals and generates spikes. Here, "aligning with neuron dynamics" means that TAB is tailored to mimic or capture neurons' behavior as closely as possible, normalizing data in line with the temporal dependencies and information accumulation within neurons. This alignment ensures that TAB's normalization process corresponds well with how neurons naturally operate in SNNs, thus leading to improved performance by addressing the temporal covariate shift problem.

2 Background

2.1 Related Work

SNN Learning Methods. Many works have recently emerged and focused on the supervised training of SNNs (Wu et al., 2021a; Zhou et al., 2021; Meng et al., 2022; Xiao et al., 2021). These SNN learning methods can be mainly categorized into two classes: ANN-to-SNN conversion (Diehl et al., 2015; Deng & Gu, 2021; Ding et al., 2021; Han et al., 2020; Li et al., 2021a; Bu et al., 2022; Hao et al., 2023; Lv et al., 2023) and end-to-end training with back-propagation (Fang et al., 2021; Zhang & Li, 2020; Deng et al., 2022; Xiao et al., 2022; Guo et al., 2022; Meng et al., 2023). ANN-to-SNN conversion takes a pre-trained ANN and converts it into an SNN by preserving the weights and replacing the ReLU activation function with a spiking activation function. This approach can be efficient in obtaining an SNN since the ANN has already been trained and the weights can be directly copied to the SNN. However, the resulting performance of the converted SNN may not be as good as that of the original source ANN. It usually requires a large number of time-steps for the converted SNN to achieve performance comparable to the source ANN. Direct end-to-end training usually employs the surrogate gradients (Wu et al., 2018; 2019; Neftci et al., 2019; Zheng et al., 2021; Eshraghian et al., 2021) method to overcome the non-differentiable nature of the binary spiking function to directly train SNNs from scratch. This method can yield comparable performance to that of traditional ANNs with a few time-steps.

BN Method in ANNs. Batch normalization methods have significantly contributed to the success of ANNs by boosting their learning and inference performance (Ioffe & Szegedy, 2015; Xiong et al., 2020; Bjorck et al., 2018). BN is a technique used to stabilize the distribution (over a mini-batch) of inputs to each network layer during training. This is achieved by introducing additional BN layers which set the first two moments (mean and variance) of the activation distribution to zero and one. Then, the batch-normalized inputs are scaled and shifted using learnable/trainable parameters to preserve model expressiveness. This normalization is performed before the non-linearity is applied. The BN layer can be formulated as,

$$\mathrm{BN}(x_i) = \gamma \hat{x}_i + \beta \,, \quad \hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \,, \quad i = 1, \cdots, b \,.$$

The mini-batch mean μ and variance σ^2 are computed by $\mu = \frac{1}{b}\sum_{i=1}^{b} x_i$ and $\sigma^2 = \frac{1}{b}\sum_{i=1}^{b}(x_i - \mu)^2$.

BN Method in SNNs. Due to the additional temporal dimension, several recent studies have proposed modifications to batch normalization to fit the training of SNNs. The threshold-dependent Batch Normalization (tdBN) method (Zheng et al., 2021) is introduced to alleviate the gradient vanishing or explosion during training SNNs. The tdBN utilizes shared BN statistics and parameters (as the conventional BN) by merging the data along the temporal dimension. Similar to tdBN, the TEBN method (Duan et al., 2022) employs shared BN statistics by merging the data along the temporal dimension, then scales using different weights to capture temporal dynamics. Different from them, BNTT (Kim & Panda, 2021) uses separate BN statistics and parameters at each time-step t independently, however, it ignores the temporal dependencies of the input spikes. Differently, our TAB method leverages the accumulated pre-synaptic inputs in the temporal domain, which is in alignment with the membrane potential accumulation in the LIF model.

2.2 Spiking Neuron Dynamics and Neuron Model

SNNs use binary spike trains to transmit information between layers. Each neuron maintains its membrane potential dynamics $\boldsymbol{u}_i(t)$ over time, "integrates" the received input with a leakage (much like an RC circuit), and fires a spike if the accumulated membrane potential value exceeds a threshold. We adopt the widely used leaky-integrate-and-fire (LIF) model. Neuron dynamics refer to the changes in the membrane potential of a neuron over time as it integrates input signals and generates spikes, which can be formulated as a first-order differential equation (ODE),

$$\text{LIF Neuron Dynamics:} \quad \tau \frac{d\boldsymbol{u}_i(t)}{dt} = -\boldsymbol{u}_i(t) + R\boldsymbol{I}_i(t), \quad \boldsymbol{u}_i(t) < V_{th}, \tag{1}$$

where $\boldsymbol{I}_i(t)$ is the injected input current to the i-th neuron at time t, $\boldsymbol{u}_i(t)$ is the membrane potential of the i-th neuron at time t in the current layer, V_{th} is the membrane threshold, and τ denotes the membrane time constant, and R denotes the resistor. For numerical simulations of LIF neurons, we consider a discrete version of the neuron dynamics. Similar to Wu & He (2018), the membrane potential $\boldsymbol{u}_i[t]$ of the i-th neuron at time-step (discrete) t is represented as:

$$\boldsymbol{u}_i[t] = \lambda \boldsymbol{u}_i[t-1] + \sum_{j \in \mathrm{pre}(i)} W_{ij}\boldsymbol{o}_j[t] \,. \tag{2}$$

We adopt a simple current model $R\boldsymbol{I}_i[t] = \sum_{j\in\mathrm{pre}(i)} W_{ij}\boldsymbol{o}_j[t]$, with R absorbed in weights W_{ij}. Here, $\boldsymbol{o}_i[t]$ denotes the binary spike of neuron i at time-step $[t]$, taking a value of 1 when a spike occurs and 0 otherwise. The index j refers to pre-synaptic neurons. The membrane potential $\boldsymbol{u}_i[t]$ increases with the summation of input spikes from all the pre-synaptic neurons $\mathrm{pre(i)}$ connecting the current i-th neuron through synaptic weight W_{ij}. It also decreases with a leak factor λ ($0 < \lambda \leqslant 1$), where λ and the time constant τ are related by $\lambda = e^{-\frac{\Delta t}{\tau}}$. The discrete LIF model degenerates to the IF model when $\lambda = 1$, therefore in the following, we only use the LIF model with $0 < \lambda \leqslant 1$. When the neuron's membrane potential $\boldsymbol{u}_i[t]$ exceeds the threshold V_{th}, the neuron will fire a spike with $\boldsymbol{o}_i[t] = 1$ and then reset the membrane potential to 0. By combining the sub-threshold dynamics Eq. (2) and hard reset mechanism, the whole iterative LIF model can be formulated by:

$$\text{Discrete LIF Neuron Model:} \quad \boldsymbol{u}_i[t] = \lambda \boldsymbol{u}_i[t-1](1-\boldsymbol{o}_i[t-1]) + \sum_{j\in\mathrm{pre}(i)} W_{ij}\boldsymbol{o}_j[t] \,, \tag{3}$$

$$\boldsymbol{o}_i[t] = H(\boldsymbol{u}_i[t] - V_{th}) \,, \tag{4}$$

where $H(x)$ is the Heaviside step function, i.e., the non-differentiable spiking activation function. $H(x) = 1$ if $x > 0$ and $H(x) = 0$ otherwise.

3 Proposed TAB Method

In this section, we will present our TAB method. We begin by introducing the Temporal Dependencies and Temporal Covariate Shift in SNNs which motivate our method. Following this, we introduce our TAB method, which addresses these challenges. Finally, we establish a theoretical connection between the neural dynamics and the TAB method by deriving the closed-form solution of LIF dynamics ODE.

3.1 Motivation: Temporal Dependencies and Temporal Covariate Shift

Temporal dependencies in SNNs arise naturally from the sequential nature of spike events, where synaptic currents (also known as spike trains) are sequentially fed into spiking neurons, playing a pivotal role in capturing the dynamic evolution of input spikes over time. These networks model the dynamics of biological neurons through ODEs and utilize spikes to transmit information (Eshraghian et al., 2021). In SNNs, each neuron maintains a membrane potential, continuously 'integrating' and accumulating received spikes over time. It emits a spike only when its accumulated membrane potential exceeds a threshold, remaining inactive otherwise in the current time-step (Li et al., 2021a). This process highlights the intrinsic influence of temporal dynamics on the temporally delayed accumulation of the membrane potential. We refer to this accumulation dependency over the time dimension as temporal dependencies.

In SNNs, a phenomenon known as Temporal Covariate Shift (TCS) has been identified (Duan et al., 2022), which represents ICS (Internal Covariate Shift) (Ioffe & Szegedy, 2015) across the additional temporal dimension, and it refers to changes in the distribution of layer inputs caused by updates of preceding layers, and prior time-steps. Within the framework of SNNs, synaptic currents are sequentially fed into spiking neurons, and spike-triggered asynchronous currents are accumulated into the membrane potential which will trigger a spike when it exceeds the membrane threshold. This temporal dependency on membrane potential accumulation intensifies the internal covariate shift along the temporal domain. This temporal dependency, together with the TCS phenomenon, presents a significant challenge when integrating BN techniques into SNNs.

Our motivation comes along these lines, how to perform batch normalization in training of SNNs, but keeping in mind the temporal dependency of the data, as well as the temporal covariate shift. A simple, yet elegant, method that aligns closely with this underlying neuron dynamics comes with Temporal Accumulated Batch normalization (TAB). Generally speaking, our TAB method addresses the temporal covariate shift issue by aligning with the inherent temporal dependencies in SNNs. Fig. 1 illustrates the temporal dependencies and neuron dynamics and showcases the involvement of our proposed TAB method.

Neuronal dynamics refers to the change in membrane potential over time as a neuron integrates input signals and generates spikes. This temporal accumulation of the membrane potential in SNNs enables neurons to process input data by taking into account both past and current time-steps (with no access to future information beyond t), and the TAB method aligns closely with this underlying neuron dynamics and alleviates the TCS issue.

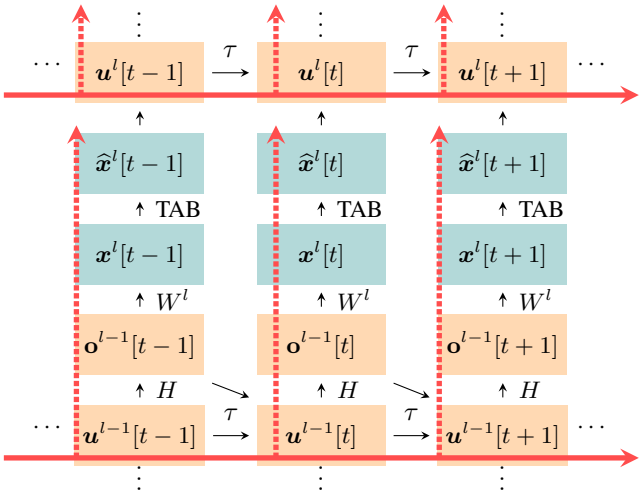

Figure 1: The temporal dependencies and neuron dynamics in SNNs, specifically the temporal dependency associated with the accumulation of membrane potential in the discrete LIF model. The black arrows represent the temporally delayed accumulation over time, while the red arrows indicate the information flow along the spatial layers (vertical axis) and the temporal domain (horizontal axis).

3.2 Temporal Accumulated Batch Normalization (TAB)

To address the temporal covariate shift issue and to model the temporal distributions in SNNs, our TAB method aligns with the inherent temporal dependencies by utilizing the temporal accumulated batch statistics $(\mu_{1:t}, \sigma^2_{1:t})$ over an expanding window $[1, t]$. To achieve this, we establish the relationship between the expectations and variances across accumulated time-steps $(\mu_{1:t}, \sigma^2_{1:t})$ and those of the

single time-step $(\mu[t], \sigma^2[t])$, as follows:

$$\mu_{1:t} = \frac{1}{t}\sum_{s=1}^{t}\mu[s]\ , \quad \sigma^2_{1:t} = \frac{1}{t}\sum_{s=1}^{t}\sigma^2[s]\ . \tag{5}$$

Our proposed TAB method utilizes Temporal Accumulated Statistics $(\mu_{1:t}, \sigma^2_{1:t})$ for data normalization, and then assigns different learnable weights $\omega[t] > 0$ to each time-step to distinguish their effect on the final result. The TAB method is given by

$$\hat{x}_i[t] = \text{TAB}(x_i[t]) = \omega[t]\left(\gamma[t]\frac{x_i[t]-\mu_{1:t}}{\sqrt{\sigma^2_{1:t}+\epsilon}} + \beta[t]\right) = \hat{\gamma}[t]\frac{x_i[t]-\mu_{1:t}}{\sqrt{\sigma^2_{1:t}+\epsilon}} + \hat{\beta}[t]\ ,\ \omega[t] > 0\ . \tag{6}$$

Given the pre-synaptic inputs $\boldsymbol{x}^l[t]$ to layer l at time-step t, the spiking neuron with TAB is as follows,

$$\boldsymbol{x}^l[t] = \boldsymbol{W}^l\boldsymbol{o}^{l-1}[t]\ , \tag{7}$$

$$\boldsymbol{u}^l[t] = \lambda\boldsymbol{u}^l[t-1](1-\boldsymbol{o}^l[t-1]) + \widehat{\boldsymbol{x}}^l[t]\ , \tag{8}$$

$$\text{where} \quad \widehat{\boldsymbol{x}}^l[t] = \text{TAB}(\boldsymbol{x}^l[t]) = \widehat{\boldsymbol{\gamma}}[t]\frac{\boldsymbol{x}^l[t]-\boldsymbol{\mu}_{1:t}}{\sqrt{\boldsymbol{\sigma}^2_{1:t}+\epsilon}} + \widehat{\boldsymbol{\beta}}[t]\ . \tag{9}$$

Here $\boldsymbol{u}^l[t]$ and $\boldsymbol{o}^l[t]$ denote the membrane potential and binary spike outputs *of all neurons* in l-th layer at time-step t, and $\boldsymbol{W}^l$ denotes the synaptic weights between layer $l-1$ and layer l. We assign different positive weights $\boldsymbol{\omega}^l[t] > \mathbf{0}$ to each time-step which is different from Deng et al. (2022) and $\hat{\boldsymbol{\gamma}}[t] = \boldsymbol{\omega}[t]\boldsymbol{\gamma}[t], \hat{\boldsymbol{\beta}}[t] = \boldsymbol{\omega}[t]\boldsymbol{\beta}[t]$. The weights $\boldsymbol{\omega}[t]$ and parameters $\boldsymbol{\gamma}^l[t], \boldsymbol{\beta}^l[t]$ are learnable, which are trained during the training process. For details, refer to **??** and **??**. Refer to **??** for the learning rules to compute the gradients.

Computation of the temporal accumulated statistics is dynamically performed, in a moving averaging fashion, without the need to store batch data from all previous time-steps. This not only saves memory, but is also an important feature of our novel approach. For the algorithm details of the TAB method, please refer to **??** in the Appendix.

The rationale behind employing this accumulated spatial-temporal information in TAB comes from the sequential processing and temporal dependency characteristics intrinsic to spiking neurons. The TAB method utilizes the accumulated batch statistics $(\mu_{1:t}, \sigma^2_{1:t})$ over an expanding window $[1, t]$. Fig. 2 illustrates an overview of four typical BN methods used in SNNs: default BN (Ioffe & Szegedy, 2015), BNTT (Kim & Panda, 2021), tdBN (Zheng et al., 2021), and TEBN (Duan et al., 2022). A comprehensive overview of statistics and parameters used by these methods is summarized in **??** in the **??**.

As shown in **??**, BNTT (Kim & Panda, 2021) considers BN statistics at each time-step individually and calculates different BN statistics $(\mu[t], \sigma^2[t])$ and BN parameters $(\gamma[t])$ at each time-step, which ignores the temporal dependencies of the input spikes. In contrast, tdBN (Zheng et al., 2021) computes the same overall BN statistics $(\mu_{1:T}, \sigma^2_{1:T})$ and BN parameters (γ, β) across all time-steps, but overlooking the temporal differences. Similarly, TEBN (Duan et al., 2022) employs the same overall BN statistics $(\mu_{1:T}, \sigma^2_{1:T})$ as tdBN, but introduces distinct weight parameters $p[t]$ at each time-step to capture time-specific variations. However, both tdBN and TEBN, computing BN statistics over T time-steps, *implicitly assume* access to data from all T time-steps, that is, even if the current time-step is $t < T$, future information up to time-step T can also be obtained, which is not true for the temporal accumulation of membrane potential nor the neural dynamics. As illustrated in Fig. 2, the input statistics of tdBN and TEBN consider the statistics of all the time-steps and all effective batches, while BNTT considers BN statistics at each time-step. Despite these differences, none of the existing methods have addressed the alignment with the membrane potential accumulation.

3.3 Theoretical connection between TAB method and the neural dynamics

TAB is tailored to capture the temporal dependencies of neurons as closely as possible by aligning with the neuron dynamics. To explore the theoretical connection between the TAB method and the neural dynamics, we need to delve into the LIF dynamics from the perspective of differential equations. In SNNs, each neuron maintains the dynamics of its membrane potential $U(t)$ over time,

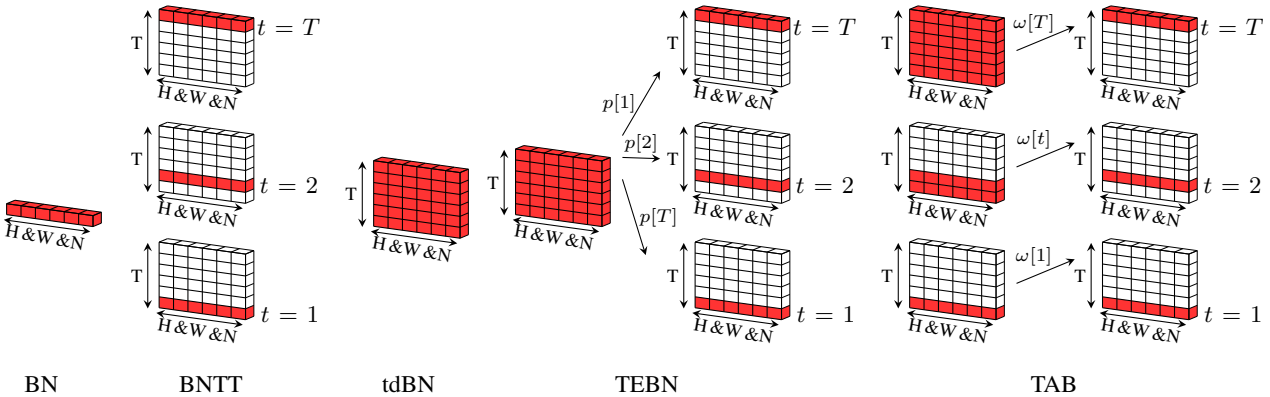

Figure 2: Comparison of different Batch Normalization methods with one given channel. In conventional BN, there is no time dimension. BNTT *independently* normalizes data at each time-step. The tdBN *jointly* normalizes data across all time-steps. TEBN shares a similar approach with tdBN but incorporates per-time-step scaling of the normalized data. In contrast, our TAB normalizes data using temporal accumulated statistics up to time-step t and subsequently applies scaling.

by "integrating" the received input current $I(t)$ with a leakage term until a spike is triggered. This is described as a first-order linear differential equation (ODE),

$$\text{Neuron Dynamics as an ODE:} \quad \tau \frac{dU(t)}{dt} = -U(t) + RI(t), \quad U(t) < V_{th} \,, \tag{10}$$

where $I(t)$ represents the input current injected into the neuron at time t, and it is a function of t (note that $I(t)$ is not a constant value). The closed-form solution of the LIF neuron dynamics (as an ODE) can be derived with analytical and theoretical methods. Additional details are available in **??** and **??**.

Lemma 1. *The analytical closed-form solution for the first-order IVP (Initial Value Problem) of the LIF dynamics ODE is as follows (Gerstner et al., 2014),*

$$U(t) = \exp\left(-\frac{t}{\tau}\right)\left(\int_0^t \frac{R}{\tau} I(s) \exp\left(\frac{s}{\tau}\right) ds + U_0\right) \,. \tag{11}$$

Remark 1. When the neuron initiates at the value U_0 with no further input, i.e., $I(t) = 0$, the closed-form solution of the ODE Eq. (11) shows that the membrane potential $U(t)$ will start at U_0 and exponentially decay with a time constant τ, $U(t) = U_0 \exp\left(-\frac{t}{\tau}\right)$. Consequently, we can determine the membrane potential ratio, often referred to as the leak factor, denoted by λ, as $\lambda = \frac{U(t+\Delta t)}{U(t)} = \frac{U_0 \exp\left(-\frac{t+\Delta t}{\tau}\right)}{U_0 \exp\left(-\frac{t}{\tau}\right)} = \exp\left(-\frac{\Delta t}{\tau}\right)$. This relationship enables us to formulate the discretization scheme as: $U[t+1] = \lambda U[t]$.

This remark provides insights into the behavior of the membrane potential in the absence of input and establishes the discretization principle used for LIF modeling.

Lemma 2. *Through applying integration by parts, we derive another equivalent form of the closed-form solution for the LIF dynamics, denoted as:*

$$U(t) = \underbrace{\overbrace{(U_0 - RI_0)\exp\left(-\frac{t}{\tau}\right)}^{\text{exponential decay term}} + \overbrace{RI(t)}^{\text{input current model}}}_{\text{commonly considered in the discrete LIF model}} \underbrace{- \int_0^t R\exp\left(\frac{s-t}{\tau}\right) dI(s)}_{\text{absent in the discrete LIF model}} \,. \tag{12}$$

With the application of the Riemann–Stieltjes integral, the discretization version of the closed-form solution is represented as:

$$U[t] = \overbrace{\lambda U[t-1]}^{(U_0 - RI_0)\exp\left(-\frac{t}{\tau}\right)} + \underbrace{\overbrace{X[t]}^{WO[t]=RI[t]} - \sum_{i=0}^{n} g_i X[s_i]}_{\textbf{TAB method}} \,. \tag{13}$$

In this formulation Eq. (13), the first exponential decay term, $\lambda U[t-1]$, captures the temporal dependency of the membrane potential from the preceding time-step. The second term, a simple current input model, $RI[t] = WO[t]$, incorporates spikes from the pre-connected neurons at the current time-step $[t]$. Significantly, the third term, representing the temporal accumulated input across all previous time-steps through a weighted sum of the input currents $X[s_i]$ with associated weights g_i, introduces a novel concept. Here $0 = s_0 < \cdots < s_i < \cdots < s_n = t$ denotes a partition of the time interval $[0, t]$ with a finite sequence of numbers. Refer to **??** for the details. Importantly, note that this accumulation mechanism of the inputs is a foundational component of the TAB method, providing a link that connects the TAB method and the neural dynamics.

Remark 2. The commonly used discrete LIF model in Eq. (2), as denoted by $U[t] = \lambda U[t-1] + X[t]$, is derived from the first two terms of the discretization version of the closed-form solution Eq. (13). The third term, representing the temporal accumulated input across all previous time-steps, however, is not incorporated into the discrete LIF models typically used in practice.

Remark 3. Note that the recursive application of the discrete LIF model, as denoted by $U[t] = \lambda U[t-1] + X[t]$, yields the temporal evolution of the membrane potential as $U[t] = \lambda^t U[0] + \sum_{s=1}^{t} \lambda^{t-s} X[s]$. This result shows the temporal dependency of the membrane potential accumulation in LIF neuron dynamics.

Recalling the TAB method introduced in Sect. 3.2, our TAB method normalizes data utilizing temporal accumulated batch statistics $(\mu_{1:t}, \sigma^2_{1:t})$ across an expanding window $[1, t]$, where $\mu_{1:t}$ and $\sigma^2_{1:t}$ represent the temporal accumulated information up to time-step $[t]$. The utilization of the temporal accumulated batch statistics aligns well with the accumulation mechanism of the membrane potentialthrough Eq. (13). Consequently, it alleviates the temporal covariate shift issue which refers to the changes in the distribution of layer inputs resulting from updates of preceding layers and prior time-steps. The entire TAB method procedure and membrane updates can be linked through Eq. (13), derived by solving the LIF dynamics ODE. This equation naturally connects TAB batch normalization to neuron dynamics, as evident in Eq. (13).

Upon comparing the commonly used discrete LIF model in Eq. (2) with the discrete closed-form solution in Eq. (13), it shows that the TAB method reintroduces the accumulation term into the normalization procedure. This is achieved by using *temporal accumulated batch statistics* from time-step 1 to t. While the *temporal accumulated batch statistics* employed by the TAB method do not replicate the exact term in Eq. (13), but as an approximation. Thus, there exists no one-to-one functional mapping between the two. The adjustment within TAB method brings the discrete LIF model closer to its analytical closed-form counterpart, thus, TAB can work well in addressing the temporal covariate shift issue. This establishes a natural connection between neuron dynamics and batch normalization.

4 Experiments

In this section, we conduct extensive experiments on large-scale static and neuromorphic datasets, CIFAR-10 (Krizhevsky et al., 2009), CIFAR100 (Krizhevsky et al., 2009), and DVS-CIFAR10 (Li et al., 2017), to verify the effectiveness of our proposed TAB method. We utilize the VGG network architecture and ResNet architecture. Firstly, we perform a comparative analysis of our TAB method with other BN methods in the context of SNNs. Further, we compare our TAB method with other state-of-the-art approaches. For implementation details, refer to **??**.

4.1 Comparison with Other BN Methods

We conduct our evaluation by comparing the performance of the proposed TAB method and other batch normalization methods in the context of SNNs. To ensure fairness in our comparisons, we do not employ advanced data augmentation techniques like cutout (DeVries & Taylor, 2017). Table 1 provides a comprehensive overview of the the test accuracy on both traditional static dataset CIFAR-10, CIFAR-100 and neuromorphic dataset DVS-CIFAR10. On the CIFAR-10 dataset, our TAB method demonstrates remarkable performance improvement, achieving a top-1 accuracy of 94.73% with the ResNet-19 network using only 2 time-steps. Notably, this surpasses the performance of TEBN using 6 time-steps. Furthermore, when using the same network architecture, TAB consistently outperforms other BN methods, even with fewer time-steps T. This pattern holds true for other

Table 1: Comparison between the proposed TAB method and other BN methods in SNNs.

Dataset	Model	Method	Architecture	Time-steps	Accuracy (%)
CIFAR-10	SPIKE-NORM (Sengupta et al., 2019)	ANN-to-SNN	VGG-16	2500	91.55
	NeuNorm (Wu et al., 2019)	Surrogate Gradient	CIFARNet	12	90.53
	BNTT (Kim & Panda, 2021)	Surrogate Gradient	VGG-9	20	90.30
	tdBN (Zheng et al., 2021)	Surrogate Gradient	ResNet-19	6 / 4 / 2	93.16 / 92.92 / 92.34
	TEBN (Duan et al., 2022)	Surrogate Gradient	VGG-9	4	92.81
			ResNet-19	6 / 4 / 2	94.71 / 94.70 / 94.57
	TAB (Ours)	Surrogate Gradient	VGG-9	4	**93.41**
			ResNet-19	6 / 4 / 2	**94.81 / 94.76 / 94.73**
CIFAR-100	SPIKE-NORM (Sengupta et al., 2019)	ANN-to-SNN	VGG-16	2500	70.90
	BNTT (Kim & Panda, 2021)	Surrogate Gradient	VGG-11	50	66.60
	TEBN (Duan et al., 2022)	Surrogate Gradient	VGG-11	4	74.37
	TEBN (Duan et al., 2022)	Surrogate Gradient	ResNet-19	6 / 4 / 2	76.41 / 76.13 / 75.86
	TAB (Ours)	Surrogate Gradient	VGG-11	4	**75.89**
			ResNet-19	6 / 4 / 2	**76.82 / 76.81 / 76.31**
DVS-CIFAR10	NeuNorm (Wu et al., 2019)	Surrogate Gradient	7-layer CNN	40	60.50
	BNTT (Kim & Panda, 2021)	Surrogate Gradient	7-layer CNN	20	63.2
	tdBN (Zheng et al., 2021)	Surrogate Gradient	ResNet-19	10	67.8
	TEBN (Duan et al., 2022)	Surrogate Gradient	7-layer CNN	10	75.10
	TAB (Ours)	Surrogate Gradient	7-layer CNN	4	**76.7**
ImageNet	SlipReLU (Jiang et al., 2023)	ANN-to-SNN	ResNet-34	32	66.61
	tdBN (Zheng et al., 2021)	Surrogate Gradient	ResNet-34	6	63.72
	TEBN (Duan et al., 2022)	Surrogate Gradient	ResNet-34	4	64.29
	TAB (Ours)	Surrogate Gradient	ResNet-34	4	**67.78**
			ResNet-34	2	**65.94**

datasets as well. For instance, on the DVS-CIFAR10 dataset, our TAB method achieves 1.6% better performance (76.7% v.s. 75.10%) while utilizing fewer time-steps (4 v.s. 10) than TEBN. Similarly, on CIFAR-100, our method exhibits a 0.55% increase in accuracy (76.31% v.s. 75.86%) compared to TEBN when both use 2 time-steps. All the accuracy values for other methods reported in the table are drawn from the existing literature.

4.2 Comparison on Large-scale ImageNet dataset

In this section, we investigate the effectiveness of our TAB method on the ImageNet dataset, renowned for its extensive collection of more than 1.25 million training images and $50,000$ test images (Deng et al., 2009). The training set of ImageNet offers $1,280$ training samples for each label, and we apply standard preprocessing and augmentation techniques (He et al., 2016) to the training data. Test data is centered and cropped to dimensions of 224×224. The evaluation employs the ResNet-34 architecture, a widely recognized model. The network is trained using the AdamW optimizer with an initial learning rate of 0.00002 and a weight decay of 0.02. Training occurs on an NVIDIA RTX A6000 with 4 GPUs, each handling a batch size of 24. To ensure unbiased statistics, we follow Zheng et al. (2021) and synchronize batch mean and variance across devices.

The results, presented in Tables Table 1 and **??**, reveal the efficacy of our TAB method. Notably, even with a modest training duration of 80 epochs for $T = 4$, the TAB method exhibits a 3.29% improvement on ResNet-34 over TEBN at $T = 4$ (TAB with 67.78% vs. TEBN 64.29%). Impressively, with only 2 time-steps ($T = 2$), our TAB method achieves an accuracy of 65.94% on ImageNet, showcasing its promising performance.

4.3 Comparison with the State-Of-The-Art Approaches

In this section, we present a comprehensive comparison of our TAB method with other state-of-the-art learning methods for SNNs using CIFAR-10 as the benchmark dataset, as illustrated in Table 2.

On the VGG-11 architecture, our TAB method achieves an impressive accuracy of 94.73% while utilizing 4 time-steps, outperforming all the ANN-to-SNN conversion and hybrid training methods that require more time-steps. Besides, we follow TEBN (Duan et al., 2022) and adopt the cutout augmentation (DeVries & Taylor, 2017) on static datasets denoted by "*" in the table. Compared to other surrogate gradient methods, our TAB method consistently performs better. On ResNet-19, our TAB method achieves an accuracy of 96.09% with 6 time-steps, which is better than Dspike (94.25%), TET (94.5%), TEBN (95.6%) while using the same number of time-steps. Even when using only 2 time-steps $T = 2$, our TAB method on ResNet-19 achieves a higher accuracy than TEBN (Duan et al., 2022) which utilizes 6 time-steps. We contribute this elevated performance to the better representation capability of TAB, achieved by its alignment with the neuron dynamics, thereby bridging the gap between the discrete LIF model and the underlying neuron dynamics and making the two closer. For clarity, all reported accuracy values for other methods in the tables are sourced from the literature. Further experimental results on CIFAR-100 and DVS-CIFAR10 datasets are detailed in **??** from **??**. For a comprehensive comparison with state-of-the-art (SOTA) methods on ImageNet, please consult **??** provided in **??**.

Table 2: Comparison between the proposed TAB and other state-of-the-art approaches on CIFAR-10.

Model	Method	Architecture	Time-steps	Accuracy (%)
RMP (Han et al., 2020)	ANN-to-SNN	ResNet-20	2048	91.36
RTS (Deng & Gu, 2021)	ANN-to-SNN	ResNet-20	128	93.56
QCFS (Bu et al., 2022)	ANN-to-SNN	ResNet-20	16	91.62
PTL (Wu et al., 2021b)	ANN-to-SNN	VGG-11	16	91.24
HC (Rathi et al., 2020)	Hybrid Training	VGG-11	2500	92.94
TC (Zhou et al., 2021)	Time-based Gradient	VGG-16	-	92.68
TSSL-BP (Zhang & Li, 2020)	Time-based Gradient	7-layer CNN	5	91.41
Dspike (Li et al., 2021b)	Surrogate Gradient	ResNet-18*	6 / 4 / 2	94.25 / 93.66 / 93.13
TET (Deng et al., 2022)	Surrogate Gradient	ResNet-19*	6 / 4 / 2	94.50 / 94.44 / 94.16
TEBN (Duan et al., 2022)	Surrogate Gradient	VGG-11	4	93.96
TEBN (Duan et al., 2022)	Surrogate Gradient	ResNet-19*	6 / 4 / 2	95.60 / 95.58 / 95.45
TAB (Ours)	Surrogate Gradient	VGG-11	4	**94.73**
		ResNet-19*	6 / 4 / 2	**96.09 / 95.94 / 95.62**

5 Conclusion

Directly training SNNs is extremely challenging, even when adopting BN techniques to enable more stable training. The presence of the Temporal Covariate Shift (TCS) phenomenon, coupled with the intrinsic temporal dependency of neuron dynamics, further compounds these challenges for directly training SNNs. To tackle this, we have introduced TAB (Temporal Accumulated Batch Normalization), a novel SNN batch normalization approach. TAB closely aligns with the neuron dynamics, normalizing data using temporal accumulated statistics, effectively capturing historical temporal dependencies similar to that of the accumulation process of the membrane potential in the LIF neuron model. Neuron dynamics refer to the changes in the membrane potential of a neuron over time as it integrates input signals and generates spikes. The alignment with the neuron dynamics means that the TAB method is tailored to mimic or capture the behavior of neurons as closely as possible. It aims to normalize the data in a manner that is coherent with the temporal dependencies and accumulation of information that occur within neurons as they process input signals. This alignment ensures that TAB's normalization process corresponds well with the way neurons naturally operate in SNNs, thereby leading to improved training and performance by addressing the temporal covariate shift problem.

Acknowledgements

This work is part of the research project ("Energy-based probing for Spiking Neural Networks", Contract No. TII/ARRC/2073/2021) in collaboration between Technology Innovation Institute (TII, Abu Dhabi) and Mohamed bin Zayed University of Artificial Intelligence (MBZUAI, Abu Dhabi).

REPRODUCIBILITY STATEMENT

The experiments and results presented in this research are reproducible, with all code, data, and detailed methodologies available in the supplementary materials. The codebase has been documented extensively, ensuring clarity and ease of implementation for future researchers. The datasets used in this study, including CIFAR-10, CIFAR-100, and DVS-CIFAR10, are publicly accessible, and we provide precise instructions on data preprocessing and augmentation procedures. Additionally, the hardware and software specifications utilized for conducting experiments are thoroughly documented, enabling researchers to replicate our results under similar computational environments. We are committed to supporting the scientific community's efforts in validating and building upon our work, thus promoting transparency and trustworthiness in the field of spiking neural networks and batch normalization techniques.

ETHICS STATEMENT

This research strictly adheres to ethical standards and guidelines governing scientific inquiry. All experiments involving living subjects or animals were not a part of this study, eliminating any ethical concerns in that regard. In terms of data usage, we employed publicly available datasets, ensuring no breach of privacy or data protection regulations. In terms of research conduct, this study promotes openness and transparency by making all code, data, and methodologies accessible to the wider scientific community. We also acknowledge and properly cite prior work, respecting intellectual property rights and academic integrity. Furthermore, this research focuses on improving the efficiency and effectiveness of spiking neural networks, which could potentially contribute to more energy-efficient AI applications. We are committed to upholding the highest ethical standards in research and encourage responsible and transparent scientific practices within the field.

REFERENCES

Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, et al. Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE transactions on computer-aided design of integrated circuits and systems*, 34(10):1537–1557, 2015.

Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Sayeed Shafayet Chowdhury, Nitin Rathi, and Kaushik Roy. Towards ultra low latency spiking neural networks for vision and sequential tasks using temporal pruning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pp. 709–726. Springer, 2022.

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee Micro*, 38(1):82–99, 2018.

Michael V DeBole, Brian Taba, Arnon Amir, Filipp Akopyan, Alexander Andreopoulos, William P Risk, Jeff Kusnitz, Carlos Ortega Otero, Tapan K Nayak, Rathinakumar Appuswamy, et al. Truenorth: Accelerating from zero to 64 million neurons in 10 years. *Computer*, 52(5):20–29, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 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*, 2022.

Shikuang Deng, Hao Lin, Yuhang Li, and Shi Gu. Surrogate module learning: Reduce the gradient error accumulation in training spiking neural networks. 2023.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Peter U Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. ieee, 2015.

Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2328–2336, 8 2021. doi: 10.24963/ijcai.2021/321. URL https://doi.org/10.24963/ijcai.2021/321. Main Track.

Jianhao Ding, Tong Bu, Zhaofei Yu, Tiejun Huang, and Jian Liu. SNN-RAT: Robustness-enhanced spiking neural network through regularized adversarial training. *Advances in Neural Information Processing Systems*, 35:24780–24793, 2022.

Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. In *Advances in Neural Information Processing Systems*, 2022.

Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *arXiv preprint arXiv:2109.12894*, 2021.

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.

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.

Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. Im-loss: information maximization loss for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:156–166, 2022.

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.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. In *The Eleventh International Conference on Learning Representations*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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.

Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: Towards optimal mapping from activation values to firing rates. 2023.

Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in neuroscience*, pp. 1638, 2021.

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, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, pp. 6316–6325. PMLR, 2021a.

Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in Neural Information Processing Systems*, 34:23426–23439, 2021b.

Changze Lv, Jianhan Xu, and Xiaoqing Zheng. Spiking convolutional neural networks for text classification. In *The Eleventh International Conference on Learning Representations*, 2023.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12444–12453, 2022.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. *arXiv preprint arXiv:2302.14311*, 2023.

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.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*, 2020.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. How does batch normalization help optimization? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, volume 31, pp. 2488–2498, 2018.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 10320–10328, 2021a.

Jibin Wu, Chenglin Xu, Xiao Han, Daquan Zhou, Malu Zhang, Haizhou Li, and Kay Chen Tan. Progressive tandem learning for pattern recognition with deep spiking neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7824–7840, 2021b.

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.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1311–1318, 2019.

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, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in Neural Information Processing Systems*, 34:14516–14528, 2021.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:20717–20730, 2022.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020.

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

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.

Shibo Zhou, Xiaohua Li, Ying Chen, Sanjeev T Chandrasekaran, and Arindam Sanyal. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11143–11151, 2021.