# Optimized Potential Initialization for Low-latency Spiking Neural Networks

**Tong Bu**[1,2], **Jianhao Ding**[2], **Zhaofei Yu**[1,2*], **Tiejun Huang**[1,2]

[1]Institute for Artificial Intelligence, Peking University
[2]Department of Computer Science and Technology, Peking University
putong30@pku.edu.cn, djh01998@stu.pku.edu.cn, {yuzf12,tjhuang}@pku.edu.cn

## Abstract

Spiking Neural Networks (SNNs) have been attached great importance due to the distinctive properties of low power consumption, biological plausibility, and adversarial robustness. The most effective way to train deep SNNs is through ANN-to-SNN conversion, which have yielded the best performance in deep network structure and large-scale datasets. However, there is a trade-off between accuracy and latency. In order to achieve high precision as original ANNs, a long simulation time is needed to match the firing rate of a spiking neuron with the activation value of an analog neuron, which impedes the practical application of SNN. In this paper, we aim to achieve high-performance converted SNNs with extremely low latency (fewer than 32 time-steps). We start by theoretically analyzing ANN-to-SNN conversion and show that scaling the thresholds does play a similar role as weight normalization. Instead of introducing constraints that facilitate ANN-to-SNN conversion at the cost of model capacity, we applied a more direct way by optimizing the initial membrane potential to reduce the conversion loss in each layer. Besides, we demonstrate that optimal initialization of membrane potentials can implement expected error-free ANN-to-SNN conversion. We evaluate our algorithm on the CIFAR-10, CIFAR-100 and ImageNet datasets and achieve state-of-the-art accuracy, using fewer time-steps. For example, we reach top-1 accuracy of 93.38% on CIFAR-10 with 16 time-steps. Moreover, our method can be applied to other ANN-SNN conversion methodologies and remarkably promote performance when the time-steps is small.

## Introduction

Spiking Neural Networks (SNNs), as the third generation of Artificial Neural Networks (ANNs) (Maass 1997), have attracted great attention in recent years. Unlike traditional ANNs transmitting information at each propagation cycle, SNNs deliver information through spikes only when the membrane potential reaches the threshold (Gerstner and Kistler 2002). Due to the event-driven calculation, sparse activation, and multiplication-free characteristics (Roy, Jaiswal, and Panda 2019), SNNs have greater energy efficiency than ANNs on neuromorphic chips (Schemmel et al. 2010; Furber et al. 2012; Merolla et al. 2014;
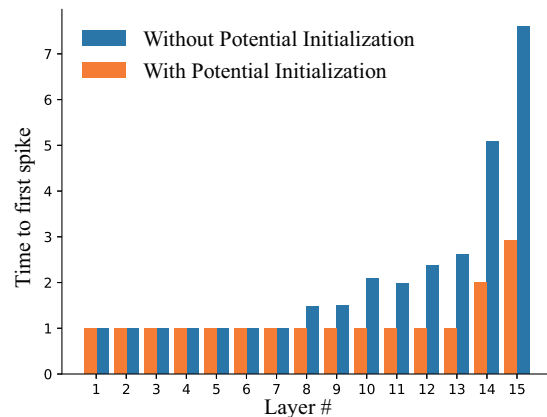
---

Figure 1: Comparison of the propagation delay of converted SNN from VGG-16 with/without membrane potential initialization on the CIFAR-10 dataset. The converted SNN without potential initialization suffers a much longer propagation delay than that with potential initialization.

Davies et al. 2018; Pei et al. 2019). In addition, SNNs have inherent adversarial robustness. The adversarial accuracy of SNNs under gradient-based attacks is higher than ANNs with the same structure (Sharmin et al. 2020). Nevertheless, the use of SNNs is still limited as it remains challenging to train high-performance SNNs.

Generally, there are two main approaches to train a multilayer SNN: (1) gradient-based optimization and (2) ANN-to-SNN conversion. The gradient-based optimization takes the idea of ANNs and computes the gradient through backpropagation (Lee, Delbruck, and Pfeiffer 2016; Lee et al. 2020). Although the surrogate gradient methods have been proposed to mitigate the non-differentiable problem of the threshold-triggered firing of SNNs (Shrestha and Orchard 2018; Wu et al. 2018; Neftci, Mostafa, and Zenke 2019), it is still limited to shallow SNNs as the gradient becomes much unstable when the layer goes deeper (Zheng et al. 2021). Besides, the gradient-based optimization method requires more GPU computing than ANN training.

Unlike the gradient-based optimization method, ANN-to-

SNN conversion builds the relationship between activation of analog neurons and dynamics of spiking neurons, and then maps the parameters of a well-trained ANN to an SNN with low accuracy loss (Cao, Chen, and Khosla 2015; Diehl et al. 2015; Rueckauer et al. 2017; Han, Srinivasan, and Roy 2020). Thus high-performance SNNs can be obtained without additional training. ANN-to-SNN conversion requires nearly the same GPU computing and time as ANN training, and has yielded the best performance in deep network structure and large-scale datasets (Deng and Gu 2021). Despite these advantages, there has been a trade-off between accuracy and latency. In order to achieve high precision as original ANNs, a long simulation time is needed to match the firing rate of a spiking neuron with the activation value of an analog neuron, which impedes the practical application of SNN.

In this paper, we make a step towards high-performance converted SNNs with extremely low latency (fewer than 32 time-steps). Instead of introducing constraints that facilitate ANN-to-SNN conversion at the cost of model capacity, we show that the initialization of membrane potentials, which are typically chosen to be zero for all neurons, can be optimized to alleviate the trade-off between accuracy and latency. Although zero initialization of membrane potentials can make it easier to relate activation of analog neurons to dynamics of spiking neurons, it also comes with inevitable long latency problems. As illustrated in Fig. 1, we find that without proper initialization, the neurons in converted SNN take a long time to fire the first spike, and thus the network is "inactive" in the first few time-steps. Based on this, we analyze ANN-to-SNN conversion theoretically and prove that the expectation of square conversion error reaches the minimum value when the initial membrane potential is half of the firing threshold. Meanwhile, the expectation of conversion error reaches zero. By setting an optimal initial value in converted SNN, we find a considerable decrease in inference time and a remarkably increased accuracy in low inference time.

The main contributions of this paper can be summarized as follows:

- We theoretically analyze ANN-to-SNN conversion and show that scaling the thresholds does play a similar role as weight normalization, which can help to explain why threshold balancing can reduce the conversion loss and improve the inference latency.

- We prove that the initialization of membrane potentials, which are typically chosen to be zero for all neurons, can be optimized to implement expected error-free ANN-to-SNN conversion.

- We demonstrate the effectiveness of the proposed method in deep network architectures on the CIFAR-10, CIFAR-100 and ImageNet datasets. The proposed method achieves state-of-the-art accuracy on nearly all tested datasets and network structures, using fewer time-steps.

- We show that our method can be applied to other ANN-SNN conversion methodologies and remarkably promote performance when the time-steps is small.

## Related Work

**Gradient-based optimization** The gradient-based optimization methods directly compute the gradient through backpropagation, which can be divided into two different categories (Kim, Kim, and Kim 2020): (1) activation-based methods and (2) timing-based methods. The activation-based methods unfold the SNNs into discrete time-steps and compute the gradient with backpropagation through time (BPTT), which borrow the idea from training recurrent neural networks in ANNs (Lee, Delbruck, and Pfeiffer 2016; Lee et al. 2020). As the gradient of the activation with respect to the membrane potential is non-differentiable, the surrogate gradient is often used (Shrestha and Orchard 2018; Wu et al. 2018; Neftci, Mostafa, and Zenke 2019; Chen et al. 2021; Fang et al. 2021a,b). However, there is a lack of rigorous theoretical analysis of the surrogate gradient (Zenke and Vogels 2021; Zenke et al. 2021). When the layer of SNNs becomes deeper (>50 layers), the gradient becomes much unstable, and the networks suffer the degradation problem (Zheng et al. 2021). The timing-based methods utilize some approximation methods to estimate the gradient of timings of firing spikes with respect to the membrane potential at the spike timing, which can significantly improves runtime efficiency of BP training. However, they are usually limited to shallow networks (<10 layers) (Mostafa 2017; Kheradpisheh and Masquelier 2020; Zhang and Li 2020; Zhou et al. 2021; Wu et al. 2021).

**ANN-to-SNN conversion** The ANN-to-SNN conversion is first proposed by Cao et al. (2015), which trains an ANN with ReLU activations and then converts the ANN to an SNN by replacing the activations with spiking neurons. By properly mapping the parameters in ANN to SNN, deep SNNs can gain comparable performance as deep ANNs. Further methods have been proposed to analyze conversion loss and improve the overall performance of converted SNNs, such as weight normalization and threshold balancing (Diehl et al. 2015; Rueckauer et al. 2016; Sengupta et al. 2019). A soft reset mechanism is applied to IF neurons in previous work (Rueckauer et al. 2016; Han, Srinivasan, and Roy 2020), to avoid information loss when neurons are reset. These works can achieve loss-less conversion with long inference time-steps (Kim et al. 2020), but still suffer from severe accuracy loss with relatively small time-steps. In recent works, most studies focus on accelerating the inference with converted SNN. Stockl and Maass (2021) propose new spiking neurons to better relate ANNs to SNN. Han and Roy (2020) use a time-based encoding scheme to speed up inference. RMP (Han, Srinivasan, and Roy 2020), RNL (Ding et al. 2021) and TCL (Ho and Chang 2020) try to alleviate the trade-off between accuracy and latency by adjusting the threshold dynamically. Ding et al. (2021) propose an optimal fit curve to quantify the fit between ANNs' activations and SNNs' firing rates and demonstrate that the inference time can be reduced by optimizing the upper bound of the fit curve. Hwang et al. (2021) proposed a layer-wisely searching algorithm and performed adequate experiments to explore the best initial value of membrane potential. Deng et al. (2020) and Li et al. (2021) propose a new method to shift weight, bias and membrane potential in each layer, making

relatively low-latency in converted SNNs. Different from the above methods, we directly optimize the initial membrane potential to increase performance at low inference time.

## Methods

In this section, we first introduce the neuron models for ANNs and SNNs, then we derive the mathematical framework for ANN-to-SNN conversion. Based on this, we show that the initial membrane potential is essential to ANN-to-SNN conversion, and derive the optimal initialization to achieve expected error-free conversion.

### ANNs and SNNs

The fundamental idea behind ANN-to-SNN conversion is to build the relationship between the activation value of an analog neuron and the firing rate of a spiking neuron. Based on this relation, we can map the weights of trained ANNs to SNNs. Thus high-performance SNNs can be obtained without additional training (Cao, Chen, and Khosla 2015). To be specific, for an ANN, the ReLU activation function of analog neurons in layer $l$ ($l = 1, 2, ..., L$) can be described as:

$$\boldsymbol{a}^l = \max(W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l, 0), \qquad (1)$$

where vector $\boldsymbol{a}^l$ denotes the output activation values of all neurons in layer $l$, $W^l$ is the weight matrix between neurons in layer $l-1$ and neurons in layer $l$, and $\boldsymbol{b}^l$ refers to the bias of the neurons in layer $l$.

For SNNs, we consider the Integrate-and-Fire (IF) model, which is commonly used in the previous works (Cao, Chen, and Khosla 2015; Diehl et al. 2015; Han, Srinivasan, and Roy 2020). In the IF model, if the spiking neurons in layer $l$ receive input $\boldsymbol{x}^{l-1}(t)$ at time $t$, the temporal membrane potential $\boldsymbol{v}^l_{temp}(t)$ can be formulated as the addition of its membrane potential $\boldsymbol{v}^l(t-1)$ at time $t-1$ and the summation of weighted input:

$$\boldsymbol{v}^l_{temp}(t) = \boldsymbol{v}^l(t-1) + W^l \boldsymbol{x}^{l-1}(t) + \boldsymbol{b}^l, \qquad (2)$$

where $\boldsymbol{x}^{l-1}(t)$ denotes the unweighted postsynaptic potentials from presynaptic neurons in layer $l-1$ at time $t$, $W^l$ is the synaptic weights, and $\boldsymbol{b}^l$ is the bias potential of spiking neurons in layer $l$. When any element $\boldsymbol{v}^l_{temp,i}(t)$ of $\boldsymbol{v}^l_{temp}(t)$ exceeds the firing threshold $V^l_{th}$ at layer $l$, the neuron will elicit a spike with unweighted postsynaptic potential $V^l_{th}$:

$$\boldsymbol{s}^l_i(t) = \begin{cases} 1, & \text{if } \boldsymbol{v}^l_{temp,i}(t) > V^l_{th} \\ 0, & \text{otherwise} \end{cases}, \qquad (3)$$

$$\boldsymbol{x}^l(t) = \boldsymbol{s}^l(t) V^l_{th}. \qquad (4)$$

Here $\boldsymbol{s}^l_i(t)$ is the $i$-th element of $\boldsymbol{s}^l(t)$, which denotes the output spike at time $t$ and equals 1 if there is a spike and 0 otherwise. After firing a spike, the membrane potential $\boldsymbol{v}^l(t)$ at the next time-step $t$ will go back to a reset value. Two approaches are commonly used to reset the potential: "reset-to-zero" and "reset-by-subtraction". As there exists obvious information loss in "reset-to-zero"(Rueckauer et al. 2017; Han and Roy 2020), we adopt "reset-by-subtraction" mechanism

in this paper. Specifically, after the firing, the membrane potential is reduced by an amount equal to the firing threshold $V^l_{th}$. Thus the membrane potential updates according to:

$$\boldsymbol{v}^l(t) = \boldsymbol{v}^l(t-1) + W^l \boldsymbol{x}^{l-1}(t) + \boldsymbol{b}^l - \boldsymbol{s}^l(t) V^l_{th}. \qquad (5)$$

### Theory for ANN-SNN conversion

In order to relate the firing rate of SNNs to the activation value of ANNs, here we accumulate Eq. (5) from time 1 to $T$, divide it by $TV^l_{th}$, and get:

$$\frac{\boldsymbol{v}^l(T)}{TV^l_{th}} - \frac{\boldsymbol{v}^l(0)}{TV^l_{th}} = \frac{W^l \sum_{t=1}^T \boldsymbol{x}^{l-1}(t)}{TV^l_{th}} + \frac{\boldsymbol{b}^l}{V^l_{th}} - \frac{\sum_{t=1}^T \boldsymbol{s}^l(t)}{T}. \qquad (6)$$

We use $\boldsymbol{r}^l(T) = \frac{\sum_{t=1}^T \boldsymbol{s}^l(t)}{T}$ to denote the firing rates of spiking neurons in layer $l$ during the period from time 0 to $T$, and substitute Eq. (4) into Eq. (6) to eliminate $\boldsymbol{x}^{l-1}(t)$, we have:

$$\boldsymbol{r}^l(T) = \frac{W^l \boldsymbol{r}^{l-1}(T) V^{l-1}_{th} + \boldsymbol{b}^l}{V^l_{th}} - \frac{\boldsymbol{v}^l(T)}{TV^l_{th}} + \frac{\boldsymbol{v}^l(0)}{TV^l_{th}}. \qquad (7)$$

Note that Eq. (7) is the core equation of ANN-SNN conversion. It describes the relationship of the firing rates of neurons in adjacent layers of an SNN, and can be related to the forwarding process of an ANN (Eq. (1)). To see this, we make the following assumption: The inference time (latency) $T$ is large enough so that $\frac{\boldsymbol{v}^l(T)}{T} \approx 0$ and $\frac{\boldsymbol{v}^l(0)}{T} \approx 0$. Hence Eq. (7) can be simplified as:

$$\boldsymbol{r}^l(T) = \frac{W^l \boldsymbol{r}^{l-1}(T) V^{l-1}_{th} + \boldsymbol{b}^l}{V^l_{th}} \qquad (8)$$

$$= \max \left( \frac{W^l \boldsymbol{r}^{l-1}(T) V^{l-1}_{th} + \boldsymbol{b}^l}{V^l_{th}}, 0 \right).$$

The last equality holds as the firing rate $\boldsymbol{r}^l(T)$ is strictly restricted in $[0, 1]$. By contrast, the ReLU activation values $\boldsymbol{a}^l$ of ANNs in Eq. (1) only need to satisfy $\boldsymbol{a}^l \geqslant 0$. In fact, the activation values $\boldsymbol{a}^l$ of ANNs have an upper bound for countable limited dataset. Thus we can perform normalization for all activation values in Eq. (1). Specifically, assuming that $\boldsymbol{z}^l = \frac{\boldsymbol{a}^l}{\max\{\boldsymbol{a}^l\}}$, where $\max\{\boldsymbol{a}^l\}$ denotes the maximum value of $\boldsymbol{a}^l$, we can rewritten Eq. (1) as:

$$\boldsymbol{z}^l = \max \left( \frac{W^l \boldsymbol{z}^{l-1} \max\{\boldsymbol{a}^{l-1}\} + \boldsymbol{b}^l}{\max\{\boldsymbol{a}^l\}}, 0 \right). \qquad (9)$$

By comparing Eq. (8) and Eq. (9), we can find an ANN can be converted to an SNN by copying both weights and biases, and setting the firing threshold $V^l_{th}$ equal to the upper bound $\max\{\boldsymbol{a}^l\}$ of the ReLU activation of analog neurons. Our result can help to explain the previous finding that scaling the firing thresholds can reduce the conversion loss and improve the inference latency (Han and Roy 2020). Actually, scaling the thresholds does play a similar role as the weight normalization technique (Diehl et al. 2015; Rueckauer et al. 2017) used in ANN-to-SNN conversion.

From another perspective, we can directly relate the post-synaptic potentials of spiking neurons in adjacent layers to the forwarding process of an ANN (Eq. (1)). If we use $\hat{\boldsymbol{a}}^l(T) = \boldsymbol{r}^l(T)V_{th}^l = \frac{\sum_{t=1}^{T} V_{th}^l \boldsymbol{s}^l(t)}{T} = \frac{\sum_{t=1}^{T} \boldsymbol{x}^l(t)}{T}$ to denote the average postsynaptic potentials from presynaptic neurons in layer $l-1$ during the period from time $0$ to $T$ and substitute it to Eq. (7), we have:

$$\hat{\boldsymbol{a}}^l(T) = W^l \hat{\boldsymbol{a}}^{l-1}(T) + \boldsymbol{b}^l - \frac{\boldsymbol{v}^l(T)}{T} + \frac{\boldsymbol{v}^l(0)}{T} \quad (10)$$
$$= \max\left( W^l \hat{\boldsymbol{a}}^{l-1}(T) + \boldsymbol{b}^l - \frac{\boldsymbol{v}^l(T)}{T} + \frac{\boldsymbol{v}^l(0)}{T}, 0 \right).$$

By comparing Eq. (1) and Eq. (10), we get the same conclusion that if the inference time (latency) $T$ is large enough, an ANN can be converted to an SNN by coping both the weights and the biases. Note that although $V_{th}^l$ is not included in Eq. (10), it also should equal the maximum value of $\boldsymbol{a}^l$ due to $\hat{\boldsymbol{a}}^l(T) = \boldsymbol{r}^l(T)V_{th}^l$.

## Optimal initialization of membrane potentials

The exact equivalence between the forwarding process of an ANN and the firing rates (or postsynaptic potentials) of adjacent layers of an SNN discussed above depends on the assumed condition that the time $T$ is large enough so that $\frac{\boldsymbol{v}^l(T)}{T} \approx 0$ and $\frac{\boldsymbol{v}^l(0)}{T} \approx 0$. It incurs a long simulation time for SNNs when applied to complicated datasets. Moreover, under low latency constraints, there exists an intrinsic difference between $\boldsymbol{r}^l(T)$ and $\boldsymbol{z}^l$, which will transfer layer by layer, resulting in considerable accuracy degradation for converted SNNs. In this subsection, we will analyze the impact of membrane potential initialization and show that optimal initialization can implement expected error-free ANN-to-SNN conversion.

According to Eq. (10), we can rewrite the relationship of the postsynaptic potentials (or firing rates) of spiking neurons in adjacent layers in a new way:

$$\hat{\boldsymbol{a}}^l(T) = \max\left( \frac{V_{th}^l}{T} \left\lfloor \frac{TW^l \hat{\boldsymbol{a}}^{l-1}(T) + T\boldsymbol{b}^l + \boldsymbol{v}^l(0)}{V_{th}^l} \right\rfloor, 0 \right). \quad (11)$$

$$\boldsymbol{r}^l(T) = \max\left( \frac{1}{T} \left\lfloor \frac{TW^l \boldsymbol{r}^{l-1}(T) V_{th}^{l-1} + T\boldsymbol{b}^l + \boldsymbol{v}^l(0)}{V_{th}^l} \right\rfloor, 0 \right). \quad (12)$$

Here $TW^l \hat{\boldsymbol{a}}^{l-1}(T) + T\boldsymbol{b}^l + \boldsymbol{v}^l(0)$ represents the accumulated potential from time $0$ to $T$. $\lfloor x \rfloor$ denotes the floor function, which calculates the maximum integer that is smaller or equal to $x$. Eq. (12) holds as $\hat{\boldsymbol{a}}^l(T) = \boldsymbol{r}^l(T)V_{th}^l$. By comparing Eq. (11) and Eq. (1), Eq. (12) and Eq. (9), we can find there exist inherent quantization errors between ANNs and SNNs due to the discrete characteristic of the firing rate. Here we propose a simple and direct way that optimize the initialization of membrane potentials to reduce the error and improve the conversion.

To be specific, we suppose that the ReLU activation $\boldsymbol{a}^{l-1}$ in layer $l-1$ of an ANN is the same as the post-synaptic potentials $\hat{\boldsymbol{a}}^{l-1}(T)$ in layer $l-1$ of an SNN,

---

Algorithm 1: Overall algorithm of ANN to SNN conversion.
**Input**: An ANN $f_{\text{ANN}}(\boldsymbol{x}; W, \boldsymbol{b})$; A Dataset $D$
**Parameter**: ANN parameters $W, \boldsymbol{b}$, Trainable clipping upper-bound $\boldsymbol{\theta}$
**Output**: $f_{\text{SNN}}$
1: **for** $l = 1$ to $L$ **do**
2:     Replace ReLU$(\boldsymbol{x})$ by clip$(\boldsymbol{x}; 0, \theta^l)$
3:     **if** is MaxPooling **then**
4:         Replace MaxPooling by AvgPooling
5:     **end if**
6: **end for**
7: **for** $e = 1$ to $epochs$ **do**
8:     **for** length of Dataset $D$ **do**
9:         Sample minibatch $(\boldsymbol{x}^0, \boldsymbol{y}^0)$ from $D$
10:         **for** $l = 1$ to $L$ **do**
11:             $\boldsymbol{x}^l = $ clip$(W^l \boldsymbol{x}^{l-1} + \boldsymbol{b}^l, 0, \theta^l)$
12:         **end for**
13:         Loss $= \mathscr{L}(\boldsymbol{x}^L; \boldsymbol{y})$
14:         Update $W, \boldsymbol{b}, \boldsymbol{\theta}$ via stochastic gradient descent
15:     **end for**
16: **end for**
17: **for** $l = 1$ to $L$ **do**
18:     $f_{\text{SNN}}.W^l = f_{\text{ANN}}.W^l$
19:     $f_{\text{SNN}}.\boldsymbol{b}^l = f_{\text{ANN}}.\boldsymbol{b}^l$
20:     $f_{\text{SNN}}.V_{th}^l = f_{\text{ANN}}.\theta^l$
21:     $f_{\text{SNN}}.\boldsymbol{v}^l(0) = f_{\text{SNN}}.V_{th}^l/2$
22: **end for**
23: **return** $f_{\text{SNN}}$

---

that is $\boldsymbol{a}^{l-1} = \hat{\boldsymbol{a}}^{l-1}(T)$, $W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l = W^l \hat{\boldsymbol{a}}^{l-1}(T) + \boldsymbol{b}^l$, and then compare the outputs of ANN and SNN in layer $l$. For the convenience of representation, we use $f(\boldsymbol{z}) = \max(\boldsymbol{z}, 0)$ to denote the activation function $\boldsymbol{a}^l = \max(W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l, 0)$ of ANNs, that is $\boldsymbol{a}^l = f(\boldsymbol{z}) = \max(\boldsymbol{z}, 0)$ and $\boldsymbol{z} = W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l$. Besides, we use $f'(\boldsymbol{z}) = \max\left( \frac{V_{th}^l}{T} \left\lfloor \frac{T\boldsymbol{z} + \boldsymbol{v}^l(0)}{V_{th}^l} \right\rfloor, 0 \right)$ to denote the activation function $\hat{\boldsymbol{a}}^l(T) = \max\left( \frac{V_{th}^l}{T} \left\lfloor \frac{TW^l \hat{\boldsymbol{a}}^{l-1}(T) + T\boldsymbol{b}^l + \boldsymbol{v}^l(0)}{V_{th}^l} \right\rfloor, 0 \right)$ of SNN, that is $\hat{\boldsymbol{a}}^l(T) = f'(\boldsymbol{z}) = \max\left( \frac{V_{th}^l}{T} \left\lfloor \frac{T\boldsymbol{z} + \boldsymbol{v}^l(0)}{V_{th}^l} \right\rfloor, 0 \right)$ and $\boldsymbol{z} = W^l \hat{\boldsymbol{a}}^{l-1}(T) + \boldsymbol{b}^l$.

The expected squared difference between $\boldsymbol{a}^l$ and $\hat{\boldsymbol{a}}^l(T)$ can be defined as:

$$E_{\boldsymbol{z}} \left\| f(\boldsymbol{z}) - f'(\boldsymbol{z}) \right\|_2^2 = E_{\boldsymbol{z}} \left\| \boldsymbol{z} - \frac{V_{th}^l}{T} \left\lfloor \frac{T\boldsymbol{z} + \boldsymbol{v}^l(0)}{V_{th}^l} \right\rfloor \right\|_2^2. \quad (13)$$

Note that as the threshold is set to be the max activation value of ANN, $\boldsymbol{z} = W^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l = \boldsymbol{a}^l$ should always fall into interval $[0, V_{th}^l]$. If we assume that $\boldsymbol{z}_i$ is uniformly distributed in every small interval $[m_t, m_{t+1}]$ with the probability density function $p_i^t$ $(t = 0, 1, ..., T-1, T)$, where $\boldsymbol{z}_i$ and $\boldsymbol{v}_i^l(0)$ denote the $i$-th element in $\boldsymbol{z}$ and $\boldsymbol{v}^l(0)$, respectively, $m_0 = 0, m_{T+1} = V_{th}^l, m_t = \frac{tV_{th}^l - \boldsymbol{v}_i^l(0)}{T}$ for $t = 1, 2, ..., T$, and $p_i^0 = p_i^T$, then we can obtain the optimal initialization of membrane potentials. We have the following Theorem.

(a) VGG-16 on CIFAR-10    (b) ResNet-20 on CIFAR-10

Zero    Optimal    Uniform    Gaussian

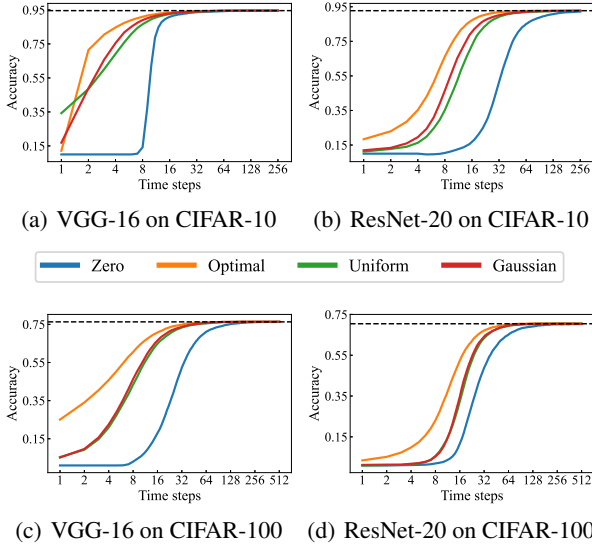(c) VGG-16 on CIFAR-100    (d) ResNet-20 on CIFAR-100

Figure 2: Comparison of different membrane potential initialization strategies with VGG-16/ResNet-20 network structures on CIFAR-10/CIFAR-100 datasets. The dotted line represents the accuracy of source ANN.

**Theorem 1.** *The expectation of square conversion error (Eq. (13)) reaches the minimum value when the initial value $\boldsymbol{v}^l(0)$ is $\boldsymbol{V}_{th}^l/2$, meanwhile the expectation of conversion error reaches 0, that is:*

$$\arg\min_{\boldsymbol{v}^l(0)} E_{\boldsymbol{z}} \left\| f(\boldsymbol{z}) - f'(\boldsymbol{z}) \right\|_2^2 = \frac{\boldsymbol{V}_{th}^l}{2}, \qquad (14)$$

$$E_{\boldsymbol{z}} \left( f(\boldsymbol{z}) - f'(\boldsymbol{z}) \right) \Big|_{\boldsymbol{v}^l(0)=\frac{\boldsymbol{V}_{th}^l}{2}} = \boldsymbol{0}. \qquad (15)$$

Here $\boldsymbol{V}_{th}^l$ is the vector of $V_{th}^l$. The detailed proof is in Supplementary Material. Theorem 1 implies that when $\boldsymbol{v}^l(0) = \boldsymbol{V}_{th}^l/2$, not only the expectation of square conversion error is minimized, but the expectation of error will be zero as well. Thus optimal initialization of membrane potential can implement expected error-free ANN-to-SNN conversion.

## Experiments

### Implementation details

We evaluate the performance of our methods for classification tasks on CIFAR-10, CIFAR-100 and ImageNet datasets. For comparison, we utilize VGG-16, ResNet-20 and ResNet-18 network structures as previous work. The proposed ANN-to-SNN conversion algorithm is given in Algorithm 1. For the source ANN, we replace all max-pooling layers with average-pooling layers. In addition, similar to (Ho and Chang 2020), we add trainable clipping layers to the source ANNs, enabling a better set of the firing thresholds of converted SNNs. For the SNN, we copy both weights and biases from source ANN, and set the firing threshold $V_{th}^l$ ($l = 1, 2, ..., L$) equal to the upper bound of the activation of analog neurons. Besides, the initial membrane potential



(a) VGG-16 on CIFAR-10    (b) ResNet-20 on CIFAR-10

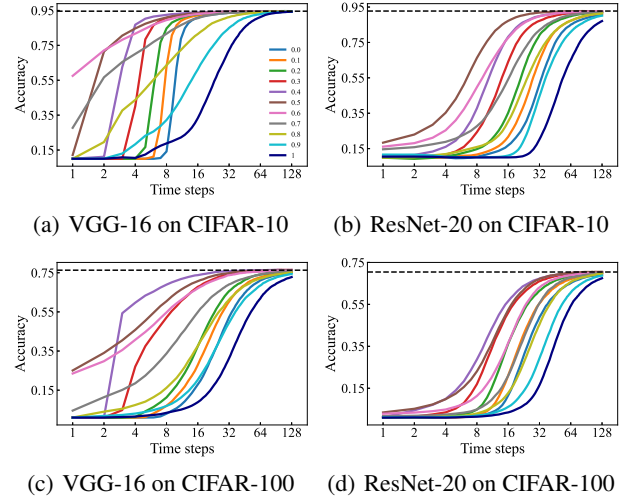(c) VGG-16 on CIFAR-100    (d) ResNet-20 on CIFAR-100

Figure 3: Comparison of different constant initial membrane potentials with VGG-16/ResNet-20 network structures on CIFAR-10/CIFAR-100 datasets. The dotted line represents the accuracy of source ANN.

$\boldsymbol{v}^l(0)$ of all spiking neurons in layer $l$ is set to the same optimal value $V_{th}^l/2$. The details of the pre-processing, parameter configuration, and training are as follows.

**Pre-processing.** We randomly crop and resize the images of CIFAR-10 and CIFAR-100 datasets into shape $32 \times 32$ after padding 4, and then conduct random horizontal flip to avoid over-fitting. Besides, we use Cutout (DeVries and Taylor 2017) with the recommended parameters. Specifically, the hole and length are 1 and 16 for CIFAR-10, and 1 and 8 for CIFAR-100. The AutoAugment (Cubuk et al. 2019) policy is also applied for both datasets. Finally, we apply data normalization on all datasets to ensure that the mean value of all input values is 0 and the standard deviation is 1. For ImageNet datasets, we randomly crop and resize the image into $224 \times 224$. We also apply CollorJitter and Label Smooth (Szegedy et al. 2016) during training. Similar to CIFAR datasets, we normalize all input data to ensure that the mean value is 0 and the standard deviation is 1.

**Hyper-Parameters.** When training ANNs, we use the Stochastic Gradient Descent optimizer (Bottou 2012) with a momentum parameter of 0.9 and a cosine decay scheduler (Loshchilov and Hutter 2017) to adjust the learning rate. The initial learning rates for CIFAR-10 and CIFAR-100 are 0.1 and 0.02, respectively. Each model is trained for 300 epochs. For ImageNet dataset, the initial learning rates is set to 0.1 and the total epoch is set to 120. The L2-regularization coefficient of the weights and biases is set to $5 \times 10^{-4}$ for CIFAR datasets and $1 \times 10^{-4}$ for ImageNet. The weight decays of the upper bound parameter $\boldsymbol{\theta}$ are $1 \times 10^{-3}$ for VGG-16 on CIFAR-10, $5 \times 10^{-4}$ for ResNet-18/20 on CIFAR-10, VGG-16/ ResNet-18/20 on CIFAR-100, and $1 \times 10^{-4}$ for VGG-16 on ImageNet.

**Training details.** When evaluating our converted SNN, we use constant input of the test images. In Fig. 4, we train

| Method | ANN Acc. | T=8 | T=16 | T=32 | T=64 | T=128 | T=256 | T≥512 |
|---|---|---|---|---|---|---|---|---|
| **VGG-16 (Simonyan and Zisserman 2014) on CIFAR-10** | | | | | | | | |
| Robust Norm (Rueckauer et al. 2017) [1] | 92.82 | - | 10.11 | 43.03 | 81.52 | 90.80 | 92.75 | 92.75 |
| Spike Norm (Sengupta et al. 2019) | 91.70 | - | - | - | - | - | - | 91.55 |
| Hybrid Train (Rathi et al. 2020) | 92.81 | - | - | - | - | 91.13 | - | 92.48 |
| RMP (Han, Srinivasan, and Roy 2020) | 93.63 | - | - | 60.30 | 90.35 | 92.41 | 93.04 | 93.63 |
| TSC (Han and Roy 2020) | 93.63 | - | - | - | 92.79 | 93.27 | 93.45 | 93.63 |
| Opt. (Deng and Gu 2021) | 95.72 | - | - | 76.24 | 90.64 | 94.11 | 95.33 | 95.73 |
| RNL (Ding et al. 2021) | 92.82 | - | 57.90 | 85.40 | 91.15 | 92.51 | 92.95 | 92.95 |
| Calibration (Li et al. 2021) | 95.72 | - | - | 93.71 | 95.14 | 95.65 | 95.79 | 95.79 |
| **Ours** | 94.57 | 90.96 | 93.38 | 94.20 | 94.45 | 94.50 | 94.49 | 94.55 |
| **ResNet-20 (He et al. 2016) on CIFAR-10** | | | | | | | | |
| Spike-Norm (Sengupta et al. 2019) | 89.10 | - | - | - | - | - | - | 87.46 |
| Hybrid Train (Rathi et al. 2020) | 93.15 | - | - | - | - | - | 92.22 | 92.94 |
| RMP (Han, Srinivasan, and Roy 2020) | 91.47 | - | - | - | - | 87.60 | 89.37 | 91.36 |
| TSC (Han and Roy 2020) | 91.47 | - | - | - | 69.38 | 88.57 | 90.10 | 91.42 |
| **Ours** | 92.74 | 66.24 | 87.22 | 91.88 | 92.57 | 92.73 | 92.76 | 92.75 |
| **ResNet-18 (He et al. 2016) on CIFAR-10** | | | | | | | | |
| Opt. (Deng and Gu 2021) [2] | 95.46 | - | - | 84.06 | 92.48 | 94.68 | 95.30 | 94.42 |
| Calibration (Li et al. 2021)[2] | 95.46 | - | - | 94.78 | 95.30 | 95.42 | 95.41 | 95.45 |
| **Ours** | 96.04 | 75.44 | 90.43 | 94.82 | 95.92 | 96.08 | 96.06 | 96.06 |

[1] Our implementation of Robust Norm.
[2] Instead of utilizing the standard ResNet-18 or ResNet-20, they add two more layers to standard ResNet-18.

Table 1: Performance comparison between the proposed method and previous work on CIFAR-10 dataset.

ResNet-20 networks on the CIFAR-10 dataset for RMP and RNL, respectively. The RMP model is reproduced according to the paper (Han, Srinivasan, and Roy 2020), and the performance is slighter high than the authors' report. The RNL model (Ding et al. 2021) is tested with the codes on GitHub provided by the authors. All experiments are implemented with PyTorch on a NVIDIA Tesla V100 GPU.

## The effect of membrane potential initialization

We first evaluate the effectiveness of the proposed membrane potential initialization. We train an ANN and convert it to four SNNs with different initial membrane potentials. Fig. 2 illustrates how the accuracy of converted SNN changes with respect to latency. The blue curve denotes zero initialization, namely without initialization. The orange, green, and red curves denote optimal initialization, random initialization from a uniform distribution, and random initialization from a Gaussian distribution, respectively. One can find that the performance of converted SNNs with non-zero initialization (orange, green and red curves) is much better than that of the converted SNN with zero initialization (blue curve), and the converted SNN with opti-

mal initialization achieves the best performance. Moreover, the SNN with zero initialization cannot work if the latency is fewer than 10 time-steps. This phenomenon could be explained as follows. As illustrated in Fig. 1, without initialization, the neurons in converted SNN take a long time to fire the first spikes, and thus the network is "inactive" in the first few time-steps. When the latency is large enough (>256 time-steps), we can find that all these methods can get the same accuracy as source ANN (dotted line).

Then we compare different constant initial membrane potentials, ranging from 0 to 1. The results are shown in Fig. 3. Overall, a larger time-steps will bring more apparent performance improvement, and the performance of all converted SNNs approach the performance of source ANN with enough time-steps. Furthermore, we can find that the converted SNNs with non-zero initialization are much better than the converted SNN with zero initialization. Note that in this experiment, all thresholds of spiking neurons are resized to 1 by scaling the weights and biases. Thus the theoretically optimal initial membrane potential is 0.5. The converted SNN from VGG-16/ResNet-20 with an initial membrane potential of 0.5 achieves optimal or near-optimal per-

| Method | ANN Acc. | T=8 | T=16 | T=32 | T=64 | T=128 | T=256 | T≥512 |
|---|---|---|---|---|---|---|---|---|
| **VGG-16 (Simonyan and Zisserman 2014) on CIFAR-100** | | | | | | | | |
| Spike-Norm (Sengupta et al. 2019) | 71.22 | - | - | - | - | - | - | 70.77 |
| RMP (Han, Srinivasan, and Roy 2020) | 71.22 | - | - | - | - | 63.76 | 68.34 | 70.93 |
| TSC (Han and Roy 2020) | 71.22 | - | - | - | - | 69.86 | 70.65 | 70.97 |
| Opt. (Deng and Gu 2021) | 77.89 | - | - | 7.64 | 21.84 | 55.04 | 73.54 | 77.71 |
| Calibration (Li et al. 2021) | 77.89 | - | - | 73.55 | 76.64 | 77.40 | 77.68 | 77.87 |
| **Ours** | 76.31 | 60.49 | 70.72 | 74.82 | 75.97 | 76.25 | 76.29 | 76.31 |
| **ResNet-20 (He et al. 2016) on CIFAR-100** | | | | | | | | |
| Spike-Norm (Sengupta et al. 2019) | 69.72 | - | - | - | - | - | - | 64.09 |
| RMP (Han, Srinivasan, and Roy 2020) | 68.72 | - | - | 27.64 | 46.91 | 57.69 | 64.06 | 67.82 |
| TSC (Han and Roy 2020) | 68.72 | - | - | - | - | 58.42 | 65.27 | 68.18 |
| **Ours** | 70.43 | 23.09 | 52.34 | 67.18 | 69.96 | 70.51 | 70.59 | 70.53 |
| **ResNet-18 (He et al. 2016) on CIFAR-100** | | | | | | | | |
| Opt. (Deng and Gu 2021) [1] | 77.16 | - | - | 51.27 | 70.12 | 75.81 | 77.22 | 77.19 |
| Calibration (Li et al. 2021) [1] | 77.16 | - | - | 76.32 | 77.29 | 77.73 | 77.63 | 77.25 |
| **Ours** | 79.36 | 57.70 | 72.85 | 77.86 | 78.98 | 79.20 | 79.26 | 79.28 |

[1] Instead of utilizing the standard ResNet-18 or ResNet-20, they add two more layers to standard ResNet-18.

Table 2: Performance comparison between the proposed method and previous work on CIFAR-100 dataset.

formance on CIFAR-10 and CIFAR-100 datasets. In fact, the derivations of optimal initial membrane potential are based on the assumption of piecewise uniform distribution. Thus a small deviation may be expected, as the assumptions cannot be strictly satisfied. To verify it further, we make a concrete analysis of the performance of converted SNN with different initial values. As shown in Fig. S1 in Supplementary Material, the optimal initial value is always between 0.4 and 0.6. The converted SNN with an initial membrane potential of 0.5 achieves optimal or near-optimal performance.

## Comparison with the State-of-the-Art

We compare our method to other state-of-the-art ANN-to-SNN conversion methods on the CIFAR-10 dataset, and the results are listed in Table 1. Our model can achieve nearly loss-less conversion with a small inference time. For VGG-16, the proposed method reaches an accuracy of 94.20% using only 32 time-steps, whereas the methods of Robust Norm, RMP, Opt., RNL and Calibration reach 43.03%, 60.3%, 76.24%, 85.4% and 93.71% at the end of 32 time-steps. Moreover, the proposed method achieves an accuracy of 90.96% using unprecedented 8 time-steps, which is 8 time faster than RMP and Opt. that use 64 time-steps. For ResNet-20, it reaches 91.88% top-1 accuracy with 32 time-steps. Note that the works of Opt. and Calibration add two layers to standard ResNet-18 rather than use the standard ResNet-18 or ResNet-20 structure. For a fair comparison, we add the experiments of converting an SNN from ResNet-18. For the same time-steps, the performance of our method

is much better than Opt., and nearly the same as Calibration, which utilizes advanced pipeline to calibrating the error and adds two more layers. Moreover, we can achieve an accuracy of 75.44% even the time-steps is only 8. These results show that the proposed method outperforms previous work and can implement fast inference.

Next, we test the performance of our method on the CIFAR-100 dataset. Table 2 compares our method with other state-of-the-art methods. For VGG-16, the proposed method reaches an accuracy of 74.82% using only 32 time-steps, whereas the methods of Opt. and Calibration reach 7.64% and 73.55% with the same time-steps. Moreover, for ResNet-20 and ResNet-18, our method can reach 52.34% and 72.85% top-1 accuracies, respectively, with 16 time-steps. These results demonstrate that our methods achieve state-of-the-art accuracy, using fewer time-steps.

Finally, we test our method on the ImageNet dataset with VGG-16 architecture. Table 3 compares the results with other state-of-the art methods. Our proposed method can achieve 64.70% top-1 accuracy using only 32 time-steps and achieve 72.47% top-1 accuracy with only 64 time-steps. All these results demonstrate that our methods is still effective on very large datasets and can reach state-of-the-art accuracy on all datasets.

## Apply optimal initial potentials to other models

Here we test whether the proposed algorithm can be applied to other ANN-to-SNN conversion models. We consider the RMP (Han, Srinivasan, and Roy 2020) and RNL (Ding

| Method | ANN Acc. | T=8 | T=16 | T=32 | T=64 | T=128 | T=256 | T≥512 |
|---|---|---|---|---|---|---|---|---|
| **VGG-16 on ImageNet** | | | | | | | | |
| Rmp (Han, Srinivasan, and Roy 2020) | 73.49 | - | - | - | - | - | 48.32 | 73.09 |
| TSC (Han and Roy 2020) | 73.49 | - | - | - | - | - | 69.71 | 73.46 |
| Opt. (Deng and Gu 2021) | 75.36 | - | - | 0.114 | 0.118 | 0.122 | 1.81 | 73.88 |
| Calibration(advanced) (Li et al. 2021) | 75.36 | - | - | 63.64 | 70.69 | 73.32 | 74.23 | 75.32 |
| **Ours** | 74.85 | 6.25 | 36.02 | 64.70 | 72.47 | 74.24 | 74.62 | 74.69 |

Table 3: Performance comparison between the proposed method and previous work on ImageNet



(a) RMP          (b) RNL
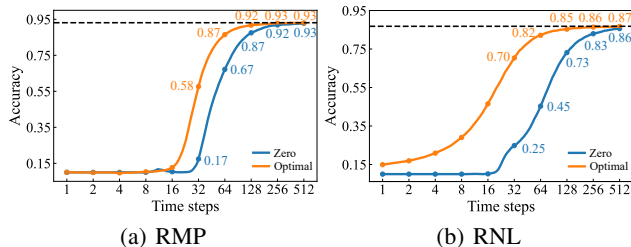
Figure 4: Comparison of the performance of converted SNNs from ResNet-20 with/without optimal initial membrane potential on the CIFAR-10 dataset.

et al. 2021) models. We train a ResNet-20 network on the CIFAR-10 dataset for each model and then convert it to two SNNs with/without optimal initial membrane potential (see Supplementary Material for more details). As illustrated in Fig. 4, one can find that the performance of converted SNN with optimal initial potential (orange curve) is much better than SNN without initialization (blue curve). For RMP model, the converted SNN with optimal initialization outperforms the original SNN by 20% in accuracy (87% vs 67%) using 64 time-steps. For RNL model, the converted SNN with optimal initialization outperforms the original SNN by by 37% in accuracy (82% vs 45%) using 64 time-steps. Moreover, the RNL model with optimal initialization outperforms the original SNN by by 45% in accuracy (70% vs 25%) using 32 time-steps. These results imply that our method is compatible with many ANN-to-SNN conversion methods and can remarkably improve performance when the time-steps is small.

## Conclusion

In this paper, we theoretically derive the relationship between the forwarding process of of an ANN and the dynamics of an SNN. We demonstrate that optimal initialization of membrane potentials can not only implement expected error-free ANN-to-SNN conversion, but also reduce the time to the first spike of neurons and thus shortening the inference time. Besides, we show that the converted SNN with optimal initial potential outperforms state-of-the-art comparing methods on the CIFAR-10, CIFAR-100 and ImageNet datasets. Moreover, our algorithm is compatible with many ANN-to-SNN conversion methods and can remarkably promote performance in low inference time.

## References

Bottou, L. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, 421–436. Springer.

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

Chen, Y.; Yu, Z.; Fang, W.; Huang, T.; and Tian, Y. 2021. Pruning of Deep Spiking Neural Networks through Gradient Rewiring. In *International Joint Conference on Artificial Intelligence*, 1713–1721.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 113–123.

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1): 82–99.

Deng, L.; Wu, Y.; Hu, X.; Liang, L.; Ding, Y.; Li, G.; Zhao, G.; Li, P.; and Xie, Y. 2020. Rethinking the performance comparison between SNNs and ANNs. *Neural Networks*, 121: 294–307.

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

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

Diehl, P. U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.-C.; and Pfeiffer, M. 2015. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *International Joint Conference on Neural Networks*, 1–8.

Ding, J.; Yu, Z.; Tian, Y.; and Huang, T. 2021. Optimal ANN-SNN conversion for fast and accurate inference in

deep spiking neural networks. In *International Joint Conference on Artificial Intelligence*, 2328–2336.

Fang, W.; Yu, Z.; Chen, Y.; Huang, T.; Masquelier, T.; and Tian, Y. 2021a. Deep residual learning in spiking neural networks. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021b. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2661–2671.

Furber, S. B.; Lester, D. R.; Plana, L. A.; Garside, J. D.; Painkras, E.; Temple, S.; and Brown, A. D. 2012. Overview of the spinnaker system architecture. *IEEE Transactions on Computers*, 62(12): 2454–2467.

Gerstner, W.; and Kistler, W. M. 2002. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge university press.

Han, B.; and Roy, K. 2020. Deep spiking neural network: Energy efficiency through time based coding. In *European Conference on Computer Vision*, 388–404.

Han, B.; Srinivasan, G.; and Roy, K. 2020. RMP-SNN: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 13558–13567.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Ho, N.-D.; and Chang, I.-J. 2020. TCL: an ANN-to-SNN conversion with trainable clipping layers. *arXiv preprint arXiv:2008.04509*.

Hwang, S.; Chang, J.; Oh, M.-H.; Min, K. K.; Jang, T.; Park, K.; Yu, J.; Lee, J.-H.; and Park, B.-G. 2021. Low-latency spiking neural networks using pre-charged membrane potential and delayed evaluation. *Frontiers in Neuroscience*, 15: 135.

Kheradpisheh, S. R.; and Masquelier, T. 2020. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(06): 2050027.

Kim, J.; Kim, K.; and Kim, J.-J. 2020. Unifying activation- and timing-based learning rules for spiking neural networks. In *Advances in Neural Information Processing Systems*, 19534–19544.

Kim, S.; Park, S.; Na, B.; and Yoon, S. 2020. Spiking-YOLO: Spiking neural network for energy-efficient object detection. In *AAAI Conference on Artificial Intelligence*, 11270–11277.

Lee, C.; Sarwar, S. S.; Panda, P.; Srinivasan, G.; and Roy, K. 2020. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14.

Lee, J. H.; Delbruck, T.; and Pfeiffer, M. 2016. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10: 508.

Li, Y.; Deng, S.; Dong, X.; Gong, R.; and Gu, S. 2021. A free lunch from ANN: Towards efficient, accurate spiking neural networks calibration. In *International Conference on Machine Learning*, 6316–6325.

Loshchilov, I.; and Hutter, F. 2017. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Maass, W. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural Networks*, 10(9): 1659–1671.

Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197): 668–673.

Mostafa, H. 2017. Supervised learning based on temporal coding in spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 29(7): 3227–3235.

Neftci, E. O.; Mostafa, H.; and Zenke, F. 2019. 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.

Pei, J.; Deng, L.; Song, S.; Zhao, M.; Zhang, Y.; Wu, S.; Wang, G.; Zou, Z.; Wu, Z.; He, W.; et al. 2019. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 572(7767): 106–111.

Rathi, N.; Srinivasan, G.; Panda, P.; and Roy, K. 2020. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*.

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

Rueckauer, B.; Lungu, I.-A.; Hu, Y.; and Pfeiffer, M. 2016. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*.

Rueckauer, B.; Lungu, I.-A.; Hu, Y.; Pfeiffer, M.; and Liu, S.-C. 2017. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 11: 682.

Schemmel, J.; Brüderle, D.; Grübl, A.; Hock, M.; Meier, K.; and Millner, S. 2010. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *IEEE International Symposium on Circuits and Systems*, 1947–1950. IEEE.

Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; and Roy, K. 2019. Going deeper in spiking neural networks: VGG and residual architectures. *Frontiers in Neuroscience*, 13: 95.

Sharmin, S.; Rathi, N.; Panda, P.; and Roy, K. 2020. Inherent adversarial robustness of deep spiking neural networks: Effects of discrete input encoding and non-linear activations. In *European Conference on Computer Vision*, 399–414.

Shrestha, S. B.; and Orchard, G. 2018. SLAYER: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, 1419–1428.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Stöckl, C.; and Maass, W. 2021. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Machine Intelligence*, 3(3): 230–238.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*.

Wu, J.; Chua, Y.; Zhang, M.; Li, G.; Li, H.; and Tan, K. C. 2021. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.

Wu, Y.; Deng, L.; Li, G.; Zhu, J.; and Shi, L. 2018. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12: 331.

Zenke, F.; Bohté, S. M.; Clopath, C.; Comşa, I. M.; Göltz, J.; Maass, W.; Masquelier, T.; Naud, R.; Neftci, E. O.; Petrovici, M. A.; et al. 2021. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4): 571–575.

Zenke, F.; and Vogels, T. P. 2021. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Computation*, 33(4): 899–925.

Zhang, W.; and Li, P. 2020. Temporal spike sequence learning via backpropagation for deep spiking neural networks. In *Advances in Neural Information Processing Systems*, 12022–12033.

Zheng, H.; Wu, Y.; Deng, L.; Hu, Y.; and Li, G. 2021. Going deeper With directly-trained larger spiking neural networks. In *AAAI Conference on Artificial Intelligence*, 11062–11070.

Zhou, S.; Li, X.; Chen, Y.; Chandrasekaran, S. T.; and Sanyal, A. 2021. Temporal-coded deep spiking neural network with easy training and robust performance. In *AAAI Conference on Artificial Intelligence*, 11143–11151.