Temporal-Coded Spiking Neural Networks with Dynamic Firing Threshold: Learning with Event-Driven Backpropagation

Wenjie Wei[1], Malu Zhang[1*], Hong Qu[1], Ammar Belatreche[2], Jian Zhang[3], and Hong Chen[3]

[1]University of Electronic Science and Technology of China,
[2]Northumbria University, [3]Tsinghua University

{maluzhang,hongqu}@uestc.edu.cn,ammar.belatreche@northumbria.ac.uk,hongchen@tsinghua.edu.cn

Abstract

Spiking Neural Networks (SNNs) offer a highly promising computing paradigm due to their biological plausibility, exceptional spatiotemporal information processing capability and low power consumption. As a temporal encoding scheme for SNNs, Time-To-First-Spike (TTFS) encodes information using the timing of a single spike, which allows spiking neurons to transmit information through sparse spike trains and results in lower power consumption and higher computational efficiency compared to traditional rate-based encoding counterparts. However, despite the advantages of the TTFS encoding scheme, the effective and efficient training of TTFS-based deep SNNs remains a significant and open research problem. In this work, we first examine the factors underlying the limitations of applying existing TTFS-based learning algorithms to deep SNNs. Specifically, we investigate issues related to over-sparsity of spikes and the complexity of finding the 'causal set'. We then propose a simple yet efficient dynamic firing threshold (DFT) mechanism for spiking neurons to address these issues. Building upon the proposed DFT mechanism, we further introduce a novel direct training algorithm for TTFS-based deep SNNs, called DTA-TTFS. This method utilizes event-driven processing and spike timing to enable efficient learning of deep SNNs. The proposed training method was validated on the image classification task and experimental results clearly demonstrate that our proposed method achieves state-of-the-art accuracy in comparison to existing TTFS-based learning algorithms, while maintaining high levels of sparsity and energy efficiency on neuromorphic inference accelerator.

*Corresponding author

1. Introduction

Deep learning approaches have demonstrated remarkable performance in various intelligent applications [18, 27, 24, 23] and innovative simulations [62, 61]. However, these achievements come at a significant cost of energy consumption, severely limiting the deployment of artificial neural networks (ANNs) on resource-limited platforms [43, 9, 3, 50]. Brain-inspired spiking neural networks (SNNs) have emerged as a promising and energy-efficient alternative to ANNs [22, 31, 57]. SNNs have garnered attention for their distinctive features of sparse-asynchronous spikes and event-driven computing, which have played a pivotal role in driving the development of neuromorphic computing platforms such as ANP-I [54], TrueNorth [1], Loihi [8], and Tianjic [39].

Despite the inherent energy efficiency of SNNs, their extensive application to real-world practical scenarios has been limited by the absence of efficient and scalable training algorithms [52, 35, 30, 14, 20]. Traditional training techniques for ANNs, such as error backpropagation (BP) and standard GPU-accelerated training packages, cannot be leveraged directly by SNNs due to the complex temporal dynamics and the non-differentiability of spiking neurons' activity [33, 6, 56]. To address these challenges, various solutions have been proposed, broadly categorized into three distinct groups.

The first category of solutions proposed to address the lack of scalable training algorithms for SNNs is ANN-to-SNN conversion method [40, 41, 49, 45]. These methods involve training an ANN first and then converting it to its SNN counterpart. The resulting SNN may achieve comparable performance to pre-trained ANNs through manual and careful setting of the SNN parameters such as the firing threshold [28] and the chosen spiking neuron model [44]. However, such conversion often results in precision loss due to approximation and requires longer time steps for lossless

conversion [19]. Moreover, ANN-to-SNN conversion methods often encode the activation values of artificial neurons using the firing rate of spiking neurons [11, 4, 5], which fails to fully leverage the timing information of spikes. Furthermore, the runtime computation on neuromorphic hardware can often be highly energy-intensive.

The second category of the proposed SNN training solutions is RNN-like method, commonly known as surrogate gradient learning [47, 48]. In these surrogate gradient-based SNNs, information is conveyed through spikes during feedforward computation, while during backpropagation, gradient information is calculated by treating the neurons' membrane potential as a differentiable signal at the current time step [42, 34]. To address the non-differentiability of spikes in the direct training of SNNs, surrogate functions are employed to approximate the derivative of the spike generation function during the backward process [53, 29]. However, while these methods have successfully tackled the challenges associated with the non-differentiability of spiking neurons, they necessitate updating synaptic weights at each time step, regardless of whether there is a spike emission or not [47, 15]. Consequently, this results in substantial energy and memory demands as large intermediate activation values necessitate storage to facilitate gradient computation.

Unlike the above-mentioned methods, the third category of learning algorithms in SNNs is spike-driven method [2, 33, 58, 60, 38, 6], which operate in a strictly event-driven manner during both forward and backward computations. SpikeProp [2] and its various extensions [33, 6, 25] are typical examples of this category. However, these algorithms remain limited to shallow network structures and relatively simple classification tasks. To overcome this limitation, Zhang *et al*. [58] and Zhou *et al*. [60] extended the spike-driven algorithm to more complex network structures. Despite achieving competitive accuracy on large datasets [60], the identification of the 'causal set'[1] in this approach requires substantial computational resources. To improve the training efficiency, Park *et al*. [38] proposed a surrogate deep neural network (DNN) model that is subsequently converted into a TTFS-based SNN. However, this conversion method is prone to conversion errors, potentially impairing the temporal learning capability of SNNs. Recently, several multi-spike-based spike-driven algorithms have been developed [59, 65]. While Zhang *et al*. [59] and Zhu *et al*. [65] achieve competitive performance on CIFAR utilizing the rate coding scheme, we focus on developing an energy-efficient TTFS-based spike-driven algorithm.

In this paper, we introduce an effective and efficient spike-driven learning algorithm that facilitates the training of high-performance TTFS-based deep SNNs. The main contributions of this work are outlined as follows:

- We provide a comprehensive analysis of the main shortcomings of existing methods in achieving high performance in TTFS-based deep SNNs. Specifically, we examine issues such as the over-sparsity of spikes and the complexity of determining the 'causal set'.
- We propose a simple yet efficient dynamic firing threshold (DFT) mechanism for spiking neurons that can effectively address the aforementioned issues.
- Building upon the proposed DFT, we further introduce a novel direct training algorithm for TTFS-based deep SNNs, which we refer to as DTA-TTFS. In this training method, the timing of a single spike is considered the basic information carrier and the learning process is performed strictly in an event-driven manner.
- The proposed method is validated through extensive experiments on benchmark image classification tasks and achieves state-of-the-art accuracy compared to existing TTFS-based learning algorithms. Furthermore, we demonstrate the ultra-low power capability of the SNN with DTA-TTFS on a recently developed neuromorphic accelerator.

2. Preliminaries and Problem Analysis

In this section, we first briefly review the TTFS coding and the employed spiking neuron model. Subsequently, we provide a comprehensive analysis of the primary shortcomings in existing TTFS-based deep SNN methods.

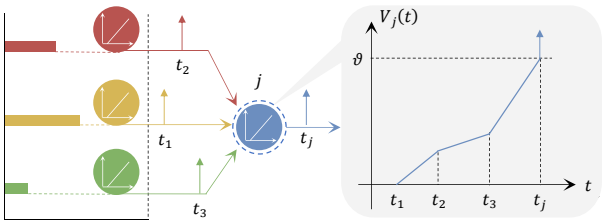
2.1. Preliminaries

TTFS coding scheme: There are two primary coding schemes utilized to represent information in SNNs: rate coding and temporal coding. The rate coding method represents information based on the average firing rate of a spiking neuron. However, this encoding scheme neglects the temporal information carried by spikes, leading to higher spike frequency and increased energy consumption. Conversely, temporal coding leverages the precise timing of spikes to encode information, resulting in lower power consumption and higher computational efficiency compared to the rate-based coding method.

As a temporal coding method, TTFS encodes information by the timing of the first spike. As depicted in Fig. 1, the higher the activation value, the earlier the first spike is emitted. The relationship between the input activation value a_i and the encoded spike time t_i can be expressed as

$$t_i = \left(1 - \frac{a_i}{a_{max}}\right) \times T_w, \tag{1}$$

[1] Assume a postsynaptic neuron fire a first spike at t_{out}, only a subset of input spikes had arrived before t_{out} and contributed to the output spike. This set of spikes, $C = \{i : t_i < t_{out}\}$, is called the causal set in this work.



(a) (b)

Figure 1. (a) TTFS coding scheme. The larger intensity value corresponds to the earlier spike. (b) ReL-PSP spiking neuron model. Neuron j accumulates the linear PSP and fires a spike when the membrane potential crosses the firing threshold ϑ.

where a_{max} is the maximum input activation value, T_w is the length of the permissible spike time window. As the TTFS coding scheme requires only a single spike to convey information, it effectively diminishes the number of spikes and significantly decreases the energy requirements. Consequently, it achieves higher sparsity and energy efficiency, making it a powerful coding scheme that maximizes the energy efficiency of SNNs.

Spiking neuron model: The Rectified Linear Postsynaptic Potential (ReL-PSP) spiking neuron [58] is chosen for our method due to its simplicity and energy efficiency. As shown in Fig. 1(b), the ReL-PSP neuron model utilizes a rectified linear kernel function to shape the neuron's PSP, and its dynamics is defined as

$$V_j^l(t) = \sum_{i=1}^{N} w_{ij}^l K\left(t - t_i^{l-1}\right), \tag{2}$$

where $V_j^l(t)$ is the membrane potential of neuron j in layer l, and w_{ij}^l is the synaptic weight between neuron j and its presynaptic neuron i. N is the number of neurons in the previous layer $l-1$, t_i^{l-1} is the spike time of input neuron i, and $K\left(t - t_i^{l-1}\right)$ is the PSP function that is defined as

$$K\left(t - t_i^{l-1}\right) = \begin{cases} t - t_i^{l-1}, & \text{if} \quad t > t_i^{l-1}, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

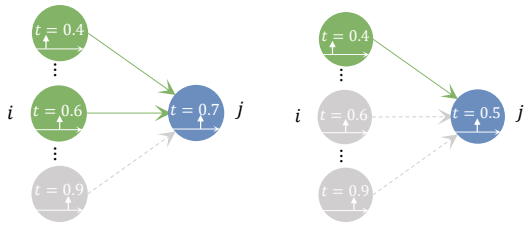
The neuron j emits a spike when $V_j^l(t)$ reaches the firing threshold ϑ. Mathematically, the spike generation function F is defined as

$$t_j^l = F\{t | V_j^l(t) \geq \vartheta, t \geq 0\}. \tag{4}$$

The ReL-PSP neuron has proven its efficacy in tackling the main challenges encountered during the training of deep SNNs, such as the non-differentiability of the spike function, gradient explosion, and the dead neuron issue. Despite the successful performance on the MNIST dataset, its performance is not ideal on deeper network structures and more complex datasets, such as CIFAR-10 and CIFAR-100. In the subsequent section, we will conduct a comprehensive analysis to identify the underlying reasons for this limitation.

2.2. Problem Analysis

Over-sparsity of spikes: Sparse neural networks offer several advantages such as efficient variable-size representation, information decomposition, and energy efficiency. However, over-sparsity may diminish the model's effective capacity and adversely affect its performance [17]. This issue becomes more serious in TTFS-based deep SNNs, owing to their firing mechanism, asynchronous transmission, and TTFS-based decision strategy.



(a) (b)

Figure 2. The over-sparsity problem caused by asynchronous transmission and TTFS-based decision strategy. (a) The presynaptic (gray) neuron fires later than the postsynaptic (blue) neuron j, making it ineffective in contributing to the firing of the postsynaptic neuron. As a result, this presynaptic neuron becomes invalid. (b) After learning, the postsynaptic (blue) neuron fires at an earlier time, leading to the appearance of additional invalid neurons.

According to the firing mechanism of spiking neurons, a neuron emits one spike only when its membrane potential reaches the firing threshold. This leads to fewer activated spiking neurons compared to ANNs. In addition, the asynchronous transmission of SNNs further diminishes the number of valid activated neurons. As shown in Fig. 2(a), when the presynaptic neuron i fires after the postsynaptic neuron j, the spike of neuron i becomes invalid for neuron j. Furthermore, TTFS-based SNNs make decisions based on the earliest spike, and the synaptic weights are trained to make the target neuron fire as early as possible. As shown in Fig. 2(b), after learning, the firing time of the postsynaptic neuron j may become earlier, thereby further exacerbating the sparsity induced by asynchronous transmission.

In event-driven learning algorithms of SNNs, error backpropagation depends on the timing of spikes. Therefore, if a spiking neuron does not fire or does not contribute to the postsynaptic neuron, it will not participate in learning. The issue of over-sparsity not only limits the information representation capability of SNNs but also leads to learning failure. Consequently, the achievement of a harmonious balance between sparsity and performance in the training of TTFS-based deep SNNs remains an open question.

The complexity of finding the 'causal set': Due to the sparsity and asynchronicity of SNNs, not all presynaptic neurons $\{i|i = 1, \ldots, N\}$ contribute to their postsynaptic neuron j, where the contributory neurons are defined as the 'causal set' of the postsynaptic neuron j, denoted as $C_j = \{i|t_i < t_j\}$ [33, 60]. To leverage standard GPU-accelerated ANN training packages, it is crucial to identify the 'causal set' of each neuron during the feed-forward computation process [33, 60]. However, this process is highly complex and time-consuming. For example, in order to find the 'causal set', Zhou *et al*. [60] proposed an approach where they first sort all of this neuron input spikes $\{t_i|i = 1, \cdots N\}$ in ascending order and subsequently consider each individual spike t_i for inclusion in the causal set. This approach, however, is computationally expensive and results in inefficient training even with the implementation of a space-for-time strategy [60].

Despite the success of algorithms with this approach in utilizing standard GPU-accelerated ANN training packages, the training process involves a significant amount of sorting and iteration. This results in higher demand for training resources and increased latency, especially as the network depth increases. Consequently, the process of finding the 'causal set' may become a bottleneck for training deep SNNs. In the next section, we propose a dynamic firing threshold mechanism for spiking neurons to avoid this resource-intensive process.

3. Method

In this section, we first introduce the dynamic firing threshold (DFT) for spiking neurons and discuss how it effectively addresses the above issues. Then, based on the DFT, we further propose a direct training algorithm for TTFS-based SNN, referred to as DTA-TTFS.

3.1. DFT-based Spiking Neuron Model

As demonstrated in Fig. 3(a), the proposed DFT is a layer-dependent and time-varying function, which is defined as

$$\vartheta^l(t) = \begin{cases} T_w(l+1) - t, & \text{if} \quad T_w l \le t \le T_w(l+1), \\ +\infty, & \text{otherwise}, \end{cases} \tag{5}$$

where $\vartheta^l(t)$ represents the firing threshold of the neurons in the l-th layer and T_w denotes the length of the permissible spike time window. In this work, T_w is set to 1 (unless otherwise stated) and the input layer is regarded as the 0-th layer. Eq. 5 reveals that the firing threshold is infinite outside the interval $t \in [T_w l, T_w(l+1)]$, and neurons in the l-th layer can only generate a spike within this time interval. Therefore, as shown in Fig. 4, the generated spikes are non-overlapping across different layers, which we refer to as the layer-dependent firing feature.

Based on the membrane potential dynamics in Eq. 2 and the proposed DFT in Eq. 5, the output of neuron j can be divided into three categories:

- Spikes at $t_j^l = T_w l$: Since the firing threshold is infinity when $t < T_w l$, the neuron cannot fire a spike before $t = T_w l$ even though the membrane potential is strong (i.e., $V_j^l(t) \ge \vartheta^l(T_w l)$ and $t < T_w l$). As shown in Fig. 3(b), the earliest spike time for DFT-based spiking neurons in the l-th layer is $t_j^l = T_w l$.

- Spikes within $(T_w l, T_w(l+1)]$: The neuron generates a spike when its membrane potential exceeds the threshold, i.e.$V_j^l(t) \ge \vartheta^l(t)$. The firing threshold ϑ^l in this time interval is a linear function, which decays linearly from 1 to 0. As depicted in Fig. 3(c), by combining the membrane potential in Eq. 2 and the firing threshold in Eq. 5, the precise spiking time t_j^l can be calculated as

$$t_j^l = \frac{T_w(l+1) + \sum\limits_{i \in C_j} w_{ij}^l t_i^{l-1}}{1 + \sum\limits_{i \in C_j} w_{ij}^l}. \tag{6}$$

- Nonspike: The neuron will not fire a spike if its membrane potential is below the dynamic firing threshold (i.e., $V_j^l(t) < \vartheta^l(t)$). Fig. 3(d) shows three typical scenarios of nonspike.

Overall, the spike time of a DFT-based neuron j in the l-th layer can be represented as

$$t_j^l = \begin{cases} T_w l, & \text{if} \quad V_j^l(T_w l) \ge \vartheta^l(T_w l), \\ \frac{T_w(l+1) + \sum\limits_{i \in C_j} w_{ij}^l t_i^{l-1}}{1 + \sum\limits_{i \in C_j} w_{ij}^l}, & \text{if} \quad T_w l < t_j^l \le T_w(l+1), \\ nonspike, & \text{otherwise}. \end{cases} \tag{7}$$

In the following, we will analyze how the proposed DFT effectively addresses issues of over-sparsity of spikes and the complexity of finding the 'causal set'.

1) Over-sparsity of spikes: As described in Eq. 5, the DFT is a piecewise function, whose linearly decreased part and layer-dependent firing feature effectively address the over-sparsity of spikes.

The linearly decreased part of the DFT is purposefully designed to tackle the over-sparsity arising from the spiking neuron firing mechanism. By gradually decreasing the firing threshold from 1 to 0, the neuron j remains active as long as its membrane potential is non-negative at $T_w(l+1)$. As a result, the proposed DFT effectively reduces the number of inactive neurons. Moreover, the linear decay design enhances the discriminative nature of information from active neurons.

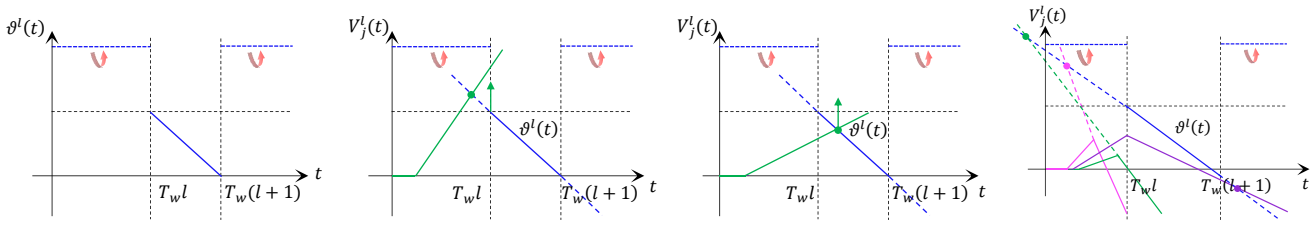


Figure 3. Proposed dynamic firing threshold and output of the DFT-based spiking neurons. (a) Dynamic firing threshold function(DFT). (b) Spikes at $t_j^l = T_w l$. (c) Spikes within $(T_w l, T_w(l+1)]$. (d) Nonspike. This case is further divided into three sub-cases, which are shown by the curves of different colors.

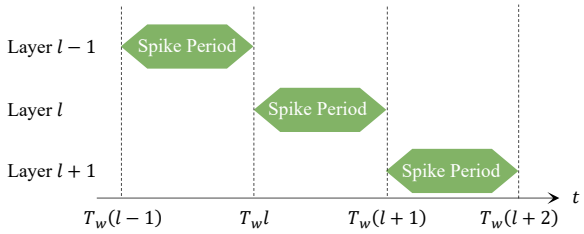


Figure 4. The propagation pipeline of the SNN with DFT.

The layer-dependent firing feature enables SNN with non-overlapping spike time windows. This effectively resolves the over-sparsity arising from asynchronous transmission and TTFS-based decision strategy. Fig. 4 illustrates that the spike time window of postsynaptic neurons occurs later than that of presynaptic neurons. Consequently, when the presynaptic neuron fires a spike, its postsynaptic neuron absolutely receives it. Furthermore, the non-overlapping spike window provides output layer neurons with a predetermined earliest spike time, preventing them from being forced forward indefinitely during the training process.

2) The complexity of finding the 'causal set': The layer-dependent firing feature of DFT circumvents the need for SNNs to identify the 'causal set', resulting in efficient training. This efficiency is further improved by the output analytic equation of the DFT-based neuron.

As depicted in Fig. 4, the layer-dependent firing feature ensures that in the l-th layer of the SNN utilizing DFT, the spike time is controlled within the interval $[T_w l, T_w(l+1)]$. Therefore, we can easily learn that neurons in the (l-1)-th layer must fire earlier than neurons in the l-th layer. In other words, all active presynaptic neurons must remain in the 'causal set' of their postsynaptic neurons. Therefore, the DFT mechanism prevents the need for resource-intensive sorting and iteration process, enhancing training efficiency and facilitating its extension to deeper network architecture.

In addition, the employed ReL-PSP spiking neuron and the DFT provide an analytical relation between input spikes and output spikes, as described in Eq.7. Therefore, the dynamics of SNNs within the training loop need not be simulated using long time steps. This allows the utilization of standard GPU-accelerated training packages of ANNs, which further leads to an efficient training process.

3.2. DTA-TTFS Learning Algorithm

Consider a task with n categories, where each output neuron corresponds to a specific category. In this task, the target category is represented by the output neuron that fires the earliest. To assess the effectiveness of the model, we adopt the widely utilized cross-entropy loss function L, which aims to maximize the output value of the target neuron. In order to ensure the desired neuron fires earliest and others fire as late as possible, we utilize the negative spike times of the output layer [58, 6], i.e., $-t$, for evaluation. Therefore, the loss function is given by

$$L\left(\mathbf{t}, d\right) = -\log \frac{\exp(-\mathbf{t}[d])}{\sum_{i=1}^{n} \exp(-\mathbf{t}[i])}, \tag{8}$$

where $\mathbf{t}$ is the vector of spike times in the output layer, and d is the desired category index.

In order to minimize the $L(\mathbf{t}, d)$, the DTA-TTFS algorithm adjusts spike times by modifying the synaptic weights during gradient backpropagation. For driving the gradient backpropagation, it is necessary to calculate derivatives of the output spike time t_j^l with respect to the synaptic weight w_{ij}^l and the input spike time t_i^{l-1}. According to Eq. 7, we can easily get these two derivatives,

$$\frac{\partial t_j^l}{\partial w_{ij}^l} = \begin{cases} \frac{t_i^{l-1} - t_j^l}{1 + \sum\limits_{i \in C_j} w_{ij}^l}, & \text{if} \quad T_w l < t_j^l < T_w(l+1), \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$

$$\frac{\partial t_j^l}{\partial t_i^{l-1}} = \begin{cases} \frac{w_{ij}^l}{1 + \sum\limits_{i \in C_j} w_{ij}^l}, & \text{if} \quad T_w l < t_j^l < T_w(l+1), \\ 0, & \text{otherwise}. \end{cases} \tag{10}$$

Following Eq. 9 and Eq. 10, gradient backpropagation can be directly applied.

Datasets	Models	Network Architecture	Neural Coding	Method	Accuracy	Sparsity
MNIST	Mostafa 2017 [33]	MLP[1]	TTFS	DT	97.55%	0.51
	Zhang *et al*. 2020 [58]	CNN[1]	TTFS	DT	99.4%	0.6614
	Zhou *et al*. 2021 [60]	CNN[2]	TTFS	DT	99.33%	0.94
	DTA-TTFS	CNN[1]	**TTFS**	**DT**	**99.4%**	**0.3913**
CIFAR-10	Wu *et al*. 2022 [46]	VGG11	Rate	conv	91.24%	no
	Park *et al*. 2020 [37]	VGG16	TTFS	conv	91.43%	0.2459
	Zhou *et al*. 2021 [60]	VGG16	TTFS	DT	92.68%	0.62
	Park *et al*. 2021 [38]	VGG16	TTFS	DT	91.90%	0.1746
	DTA-TTFS	**VGG11**	**TTFS**	**DT**	**91.17%**	**0.4387**
	DTA-TTFS	**VGG16**	**TTFS**	**DT**	**93.05%**	**0.2561**
CIFAR-100	Park *et al*. 2020 [37]	VGG16	TTFS	conv	68.79%	0.2994
	Park *et al*. 2021 [38]	VGG16	TTFS	DT	65.98%	0.2780
	DTA-TTFS	**VGG16**	**TTFS**	**DT**	**69.66%**	**0.2845**

Table 1. Comparison of accuracy and sparsity between the DTA-TTFS method and other related works. DT: direct training. conv: ANN-to-SNN conversion. MLP[1]: 784-800-10. CNN[1]: 784-16C5-P2-32C5-P2-800-128-10. CNN[2]: 784-32C5-16C5-10.

4. Experiments

In this section, we first evaluate the performance of the DTA-TTFS algorithm. Then, we carry out validation studies to illustrate the impact of the DFT. Finally, we employ theoretical analysis along with the neuromorphic hardware simulator to evaluate the energy consumption during the inference process.

4.1. Details

We conduct experiments on three image classification benchmark datasets: MNIST[2], CIFAR-10, and CIFAR-100[3]. All experiments are conducted using the PyTorch library, which allows for efficient training on multi-GPU machines with accelerated computations and optimized memory usage. For the MNIST dataset, we employ a network structure with 784-16C5-P2-32C5-P2-800-128-10 and train it for 150 epochs. As for CIFAR datasets, we adopt augmentation techniques [7, 10] and train the VGG11 and VGG16 network structures without Batch Normalization (BN) for 300 epochs. Moreover, in order to maintain the initial parameter domain at a good level, the SNN is initialized with the parameters from the pre-trained identical ANN when training CIFAR datasets. During the training process, we adopt the Adam optimizer and implement milestones learning rate decay, with an initial learning rate $1e^{-3}$ for MNIST and $1e^{-4}$ for CIFAR datasets.

4.2. Comparison with Related Works

Tab. 1 presents a comparison of the accuracies and sparsity achieved by DTA-TTFS and other related works. Experimental results demonstrate that the DTA-TTFS algorithm exhibits excellent performance, with an accuracy of 99.4% on MNIST, 93.05% on CIFAR-10, and 69.66% on CIFAR-100. Obviously, our method outperforms all previous direct learning algorithms of TTFS-based SNNs. Moreover, our method still maintains competitive sparsity when achieving high performance. The sparsity is calculated by S/N averaged on the entire testing set, where S represents the total number of spikes and N denotes the total number of neurons. Specifically, the DTA-TTFS achieves sparsity of 0.3913 on MNIST, 0.2561 on CIFAR-10, and 0.2845 on CIFAR-100. For the MNIST dataset, when employing the identical network structure as described in [58], our method achieves the same accuracy while significantly reducing the sparsity from 0.6614 to 0.3913. For CIFAR datasets, our method surpasses related works in terms of accuracy while maintaining a competitive sparsity. In summary, the DTA-TTFS algorithm achieves a harmonious balance between accuracy and sparsity.

4.3. Validation Study

We conduct a series of validation experiments to confirm the functionality of the proposed DFT. These experiments aim to display two aspects. Firstly, the DFT provides SNNs with the layer-dependent firing feature. Additionally, it effectively addresses the over-sparsity of spikes. Validation experiments choose the well-trained VGG11 structure on the CIFAR-10 dataset.

In order to show the layer-dependent firing feature of the SNN with DFT, we record the spike activity in each layer for the first sample of the testing set. For the purpose of comparison, we also provide the spike activity of the SNN without DFT. The results are visualized in Fig. 5, elucidating the evident regulation of spike activity in each layer

[2]http://yann.lecun.com/exdb/mnist/

[3]https://www.cs.toronto.edu/ kriz/cifar.html

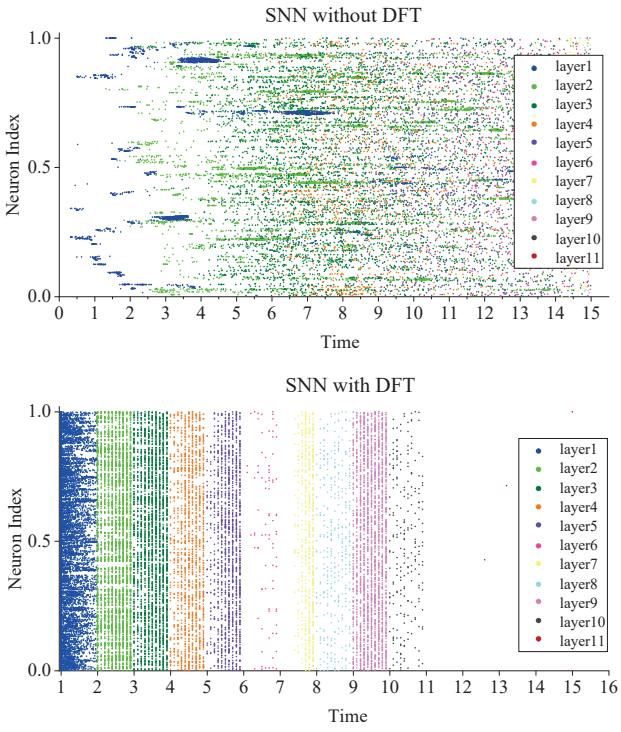


Figure 5. Spike activity in each layer of SNN with and without DFT. The vertical axis represents the neuron index scaled to [0,1].

within a permitted time window for the SNN incorporating DFT. In contrast, the spike activity in the SNN without DFT exhibits a disorderly and chaotic pattern.

In order to verify the efficacy of the DFT in addressing the over-sparsity of spikes, we count the number of inactive neurons per layer averaged on the whole testing set. This experiment is conducted on three models: the ANN, the SNN with DFT, and the SNN without DFT [58]. As shown in Fig. 6, the SNN without DFT displays significant sparsity, however, with only 10% accuracy. Conversely, the SNN with DFT exhibits a comparable amount of inactive neurons to that of ANN. As a result, the DFT achieves a harmonious trade-off between sparse representation and information representation ability.

4.4. Energy Estimation

4.4.1 Theoretical Analysis

Energy consumption is a critical metric to evaluate the efficiency of SNNs. To prove the energy efficiency of the proposed DTA-TTFS, we employ the theoretical analysis presented in [37]. This method analyzes energy consumption based on multiple factors, including latency, spike counts, as well as data related to neuromorphic architecture. More specifically, the estimated energy is defined as

$$E_{Total} = E_{Static} \times Latency + E_{Dynamic} \times Spikes, \quad (11)$$

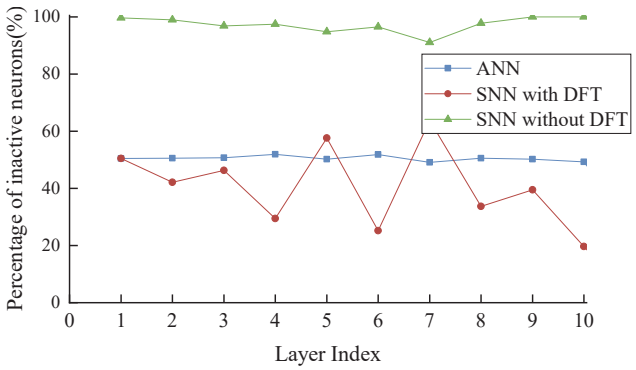


Figure 6. Percentage of inactive neurons per layer of the ANN, the SNN with DFT, and the SNN without DFT.

where $Latency$ is the required time steps, $Spikes$ is the total number of spikes, E_{Static} and $E_{Dynamic}$ are static and dynamic energy coefficients depending on the neuromorphic architecture. We choose two neuromorphic architectures, TrueNorth [32] and SpiNNaker [16], to conduct analysis. The energy parameters $(E_{Static}, E_{Dynamic})$ are set to (0.6, 0.4) for TrueNorth and (0.36, 0.64) for SpiNNaker. For comparison, we analyze various coding schemes, such as rate coding [19], phase coding [26], burst coding [36], and TTFS coding [37, 38]. Theoretical analysis is performed on CIFAR-10 with the well-trained VGG16 structure, and we normalized estimated energy consumption based on the rate coding. As indicated in Tab. 2, under the same conditions, the DTA-TTFS outperforms other methods in terms of energy efficiency. It is worth noting that, despite the fewer spikes achieved by [37, 38], their energy consumption remains higher than ours due to their demand for long latency.

Neural Coding	Time Step	Spike (10^6)	Acc. (%)	Normalized Energy	
				TrueNorth	SpiNNaker
Rate [19]	512	2.612	93.39	1	1
Phase [26]	1500	35.196	91.21	7.1476	9.6785
Burst [36]	1125	6.92	91.41	2.3781	2.4865
TTFS [37]	680	0.069	91.43	0.8074	0.4950
TTFS [38]	544	0.067	91.9	0.6478	0.3989
DTA-TTFS	**160**	**0.073**	**93.05**	**0.1987**	**0.1304**

Table 2. Theoretical analysis of energy consumption.

4.4.2 Hardware Simulation

In order to further verify the energy efficiency and hardware friendliness of our method, we map the SNN trained by the DTA-TTFS algorithm on a recently developed neuromorphic hardware simulator, called configurable asynchronous neuromorphic hardware simulator (CanMore) [55]. CanMore estimates hardware performance through system-level modeling and simulation. Fig. 7 illustrates the main

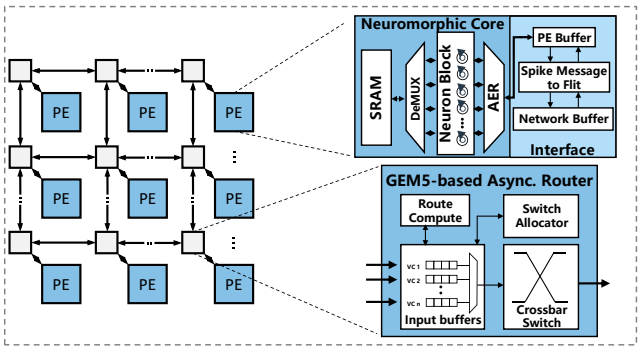


Figure 7. The framework of CanMore [55], including neuromorphic cores comprised of neurons and SRAM, and NoC routers consisting of buffers, route computation units, switch allocator, and crossbar switch.

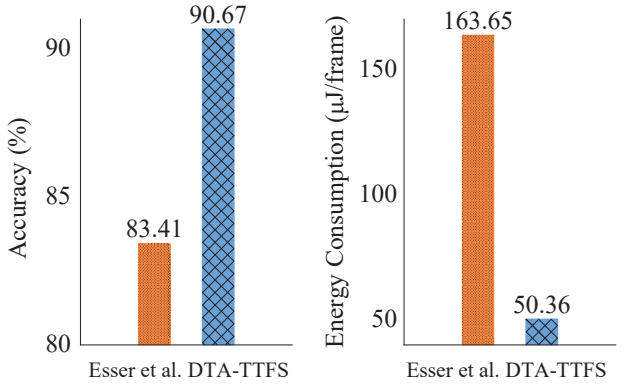


Figure 8. The accuracy and energy consumption of Esser *et al*. [13] and DTA-TTFS on CIFAR-10 dataset.

framework of the CanMore, including neuromorphic cores comprised of neurons and SRAM, and network-on-chip (NoC) routers consisting of buffers, route computation unit, switch allocator and crossbar switch. With the system-level asynchronous circuit modeling, CanMore is able to simulate various neuromorphic hardware architectures by changing the parameters and configuration files, estimate latency and energy consumption, and analyze spike messages routing information such as routing hops, congestion latency, routing directions, and so on.

In [13], a deep ANN is mapped to TrueNorth neuromorphic hardware. To the best of our knowledge, this is the only work that verifies the energy efficiency of SNNs on the CIFAR-10 dataset, while the majority of existing works focus on smaller datasets, such as MNIST [12, 58]. Here, we train an SNN on the CIFAR-10 dataset with the proposed DTA-TTFS method. After learning, the learned weights are transferred to CanMore for accelerating the inference operations. Specifically, our network structure is mapped by partitioning each layer into equally sized groups along the feature dimension, with each group being mapped to a neuromorphic core with a maximum of 1024 neurons. In addition, we compress the network by quantizing the weights to 4 bits. The mesh topology and XY routing algorithm are adopted in the simulated architecture. The energy consumption per frame during the inference process is calculated by

$$E_{Total} = E_{SOP} \times Spikes + E_{Hop} \times Hops + E_{Static}, \quad (12)$$

where E_{SOP} is the dynamic energy consumption per synaptic operation (SOP), which is calculated with the real technology parameters of the CMOS 22nm process in the simulation. In our work, E_{SOP} is 20pJ. E_{Hop} is the dynamic energy consumption per hop between the mesh routers on the NoC, which is 100pJ according to [21]. E_{Static} is static energy consumption, which is 3.52 μJ/frame in CMOS 22nm process. The average $Spikes$ and $Hops$ per frame during the inference process are 71M and 45.4M respectively, given by CanMore. Therefore, we obtain the energy consumption E_{Total} during the inference process 50.36 μJ/frame.

The accuracy and energy consumption are reported in Fig. 8. our work achieves a classification accuracy of 90.67% on CIFAR-10, surpassing the accuracy reported in [13] by 7.26%. In addition, the energy consumption of our work is 50.36 μJ/frame, which is 69.2% reduced compared with the energy consumption 163.65μJ/frame in [13]. These results confirm that the proposed DTA-TTFS algorithm is effective and energy-efficiency.

5. Conclusion

In this study, we analyze the shortcomings of existing methods in achieving high-performance TTFS-based deep SNNs. In order to overcome these issues, we propose an efficient DFT mechanism for spiking neurons and introduce the DTA-TTFS algorithm for TTFS-based SNNs. We evaluate our method on the image classification task, and the results demonstrate that our method achieves a fine trade-off between accuracy and sparsity. Additionally, we estimate the energy consumption of our method through theoretical analysis and the neuromorphic hardware accelerator CanMore. Both theoretical analysis and hardware verification confirm the energy efficiency of the DTA-TTFS algorithm. Overall, our method not only attains high performance but also is quite hardware friendly. In our future work, we intend to leverage DTA-TTFS to train more advanced network models, including SNN-based Attention [64, 51] and Transformer architectures [63].

Acknowledgment

This work was supported by the National Science Foundation of China under Grants 62236007 and 61976043 and 62106038, and in part by the Sichuan Science and Technology Program under Grants 2022YFG0313 and 2023YFG0259.

References

[1] 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.

[2] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, volume 48, pages 419–424. Bruges, 2000.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11–20, 2022.

[5] 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. *arXiv preprint arXiv:2303.04347*, 2023.

[6] Iulia-Maria Comşa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function: learning with backpropagation. *IEEE transactions on neural networks and learning systems*, 33(10):5939–5952, 2021.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

[8] 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.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[11] 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)*, pages 1–8. ieee, 2015.

[12] Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. Backpropagation for energy-efficient neuromorphic computing. *Advances in neural information processing systems*, 28, 2015.

[13] Steven K Esser, Paul A Merolla, John V Arthur, Andrew S Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J Berg, Jeffrey L McKinstry, Timothy Melano, Davis R Barch, et al. From the cover: Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences of the United States of America*, 113(41):11441, 2016.

[14] 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.

[15] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2661–2671, 2021.

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

[17] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[18] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772. PMLR, 2014.

[19] 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*, pages 13558–13567, 2020.

[20] Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[21] Masashi Imai, Thiem Van Chu, Kenji Kise, and Tomohiro Yoneda. The synchronous vs. asynchronous noc routers: an apple-to-apple comparison between synchronous and transition signaling asynchronous designs. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–8. IEEE, 2016.

[22] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[23] Yeying Jin, Aashish Sharma, and Robby T Tan. Dc-shadownet: Single-image hard and soft shadow removal using unsupervised domain-classifier guided network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5027–5036, 2021.

[24] Yeying Jin, Wenhan Yang, and Robby T Tan. Unsupervised night image enhancement: When layer decomposition meets light-effects suppression. In *European Conference on Computer Vision*, pages 404–421. Springer, 2022.

[25] S. R. Kheradpisheh and T. Masquelier. Temporal backpropagation for spiking neural networks with one spike per neuron. *International Journal of Neural Systems*, 30(6), 2020.

[26] Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018.

[27] Jinyu Li et al. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[28] 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*, pages 6316–6325. PMLR, 2021.

[29] 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, 2021.

[30] Yuguo Liu, Wenyu Chen, Hanwen Liu, Yun Zhang, Malu Zhang, and Hong Qu. Biologically plausible sparse temporal word representations. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

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

[32] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[33] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

[34] 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.

[35] Zihan Pan, Malu Zhang, Jibin Wu, Jiadong Wang, and Haizhou Li. Multi-tone phase coding of interaural time difference for sound source localization with spiking neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2656–2670, 2021.

[36] Seongsik Park, Seijoon Kim, Hyeokjun Choe, and Sungroh Yoon. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[37] Seongsik Park, Seijoon Kim, Byunggook Na, and Sungroh Yoon. T2fsnn: Deep spiking neural networks with time-to-first-spike coding. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[38] Seongsik Park and Sungroh Yoon. Training energy-efficient deep spiking neural networks with time-to-first-spike coding. *arXiv preprint arXiv:2106.02568*, 2021.

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

[40] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, and Michael Pfeiffer. Theory and tools for the conversion of analog to spiking convolutional neural networks. *arXiv preprint arXiv:1612.04052*, 2016.

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

[42] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

[43] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12), 2017.

[44] Yuchen Wang, Malu Zhang, Yi Chen, and Hong Qu. Signed neuron with memory: Towards simple, accurate and high-efficient ann-snn conversion. In *International Joint Conference on Artificial Intelligence*, 2022.

[45] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[46] 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, 2021.

[47] 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.

[48] 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, pages 1311–1318, 2019.

[49] Qi Xu, Yaxin Li, Jiangrong Shen, Jian K Liu, Huajin Tang, and Gang Pan. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7886–7895, 2023.

[50] Qi Xu, Yaxin Li, Jiangrong Shen, Pingping Zhang, Jian K Liu, Huajin Tang, and Gang Pan. Hierarchical spiking-based model for efficient image classification with enhanced feature extraction and encoding. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[51] Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 2023.

[52] Qiang Yu, Huajin Tang, Kay Chen Tan, and Haizhou Li. Rapid feedforward computation by temporal encoding and learning with spiking neurons. *IEEE transactions on neural networks and learning systems*, 24(10):1539–1552, 2013.

[53] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

[54] Jilin Zhang, Dexuan Huo, Jian Zhang, Chunqi Qian, Qi Liu, Liyang Pan, Zhihua Wang, Ning Qiao, Kea-Tiong Tang, and Hong Chen. 22.6 anp-i: A 28nm 1.5 pj/sop asynchronous spiking neural network processor enabling sub-o. 1 μj/sample on-chip learning for edge-ai applications. In *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 21–23. IEEE, 2023.

[55] Jian Zhang, Jilin Zhang, Dexuan Huo, and Hong Chen. Anas: Asynchronous neuromorphic hardware architecture search based on a system-level simulator. in proc. 60th acm/ieee design automation conference (dac), 2023. In *Proc. 60th ACM/IEEE Design Automation Conference (DAC), 2023*, 2023.

[56] Malu Zhang, Xiaoling Luo, Yi Chen, Jibin Wu, Ammar Belatreche, Zihan Pan, Hong Qu, and Haizhou Li. An efficient threshold-driven aggregate-label learning algorithm for multimodal information processing. *IEEE Journal of Selected Topics in Signal Processing*, 14(3):592–602, 2020.

[57] Malu Zhang, Hong Qu, Ammar Belatreche, Yi Chen, and Zhang Yi. A highly effective and robust membrane potential-driven supervised learning method for spiking neurons. *IEEE transactions on neural networks and learning systems*, 30(1):123–137, 2018.

[58] Malu Zhang, Jiadong Wang, Jibin Wu, Ammar Belatreche, Burin Amornpaisannon, Zhixuan Zhang, Venkata Pavan Kumar Miriyala, Hong Qu, Yansong Chua, Trevor E Carlson, et al. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE transactions on neural networks and learning systems*, 33(5):1947–1958, 2021.

[59] 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.

[60] 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, pages 11143–11151, 2021.

[61] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, pages 3–11. Springer, 2018.

[62] Zongwei Zhou, Jae Shin, Lei Zhang, Suryakanth Gurudu, Michael Gotway, and Jianming Liang. Fine-tuning convolutional neural networks for biomedical image analysis: actively and incrementally. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7340–7351, 2017.

[63] Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, YAN Shuicheng, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations*, 2022.

[64] Rui-Jie Zhu, Qihang Zhao, Tianjing Zhang, Haoyu Deng, Yule Duan, Malu Zhang, and Liang-Jian Deng. Tcja-snn: Temporal-channel joint attention for spiking neural networks. *arXiv preprint arXiv:2206.10177*, 2022.

[65] Yaoyu Zhu, Zhaofei Yu, Wei Fang, Xiaodong Xie, Tiejun Huang, and Timothée Masquelier. Training spiking neural networks with event-driven backpropagation. In *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.