Parallel Training Time-to-First-Spike Spiking Neural Networks

Kaiwei Che[1,2], Wei Fang[1,*], Peng Xue[2,3], Yifan Huang[4], Zhengyu Ma[2], Yonghong Tian[1,2,4,*]

[1]School of Electronic and Computer Engineering, Shenzhen Graduate School, Peking University, China
[2]Pengcheng Laboratory, China
[3]Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China
[4]School of Computer Science, Peking University, China

Abstract

Spiking Neural Networks (SNNs) offer a promising energy-efficient computing paradigm owing to their event-driven properties and biologically inspired dynamics. Among various encoding schemes, Time-to-First-Spike (TTFS) is particularly notable for its extreme sparsity, utilizing a single spike per neuron to maximize energy efficiency. However, two significant challenges persist: effectively leveraging TTFS sparsity to minimize training costs on Graphics Processing Units (GPUs), and bridging the performance gap between TTFS-based SNNs and their rate-based counterparts. To address these issues, we propose a parallel training algorithm for accelerated execution and a novel decoding strategy for enhanced performance. Specifically, we derive both forward and backward propagation equations for parallelized TTFS SNNs, enabling precise calculation of first-spike timings and gradients. Furthermore, we analyze the limitations of existing output decoders and introduce a membrane potential–based decoder, complemented by an incremental time-step training strategy, to improve accuracy. Our approach achieves state-of-the-art accuracy for TTFS SNNs on several benchmarks, including MNIST (99.51%), Fashion-MNIST (93.14%), CIFAR-10 (95.06%), and CIFAR-100 (74.07%).

Code — https://github.com/CheKaiWei/PTTFS

Introduction

Spiking Neural Networks (SNNs) (Maass 1997) mimic biological systems via event-driven spike communication, enabling highly energy-efficient computation. This efficiency has been successfully demonstrated on various neuromorphic hardware platforms, such as TrueNorth (Merolla et al. 2014), Loihi (Davies et al. 2021), and Tianjic (Pei et al. 2019). The functionality of SNNs is fundamentally governed by neural coding principles, which dictate how information is represented by spikes. Drawing inspiration from neurobiology, researchers have developed numerous coding schemes to translate data for these networks. Among these schemes, rate coding correlates information with neuronal firing rates (Adrian 1926), but it has been shown to be inefficient for tasks requiring rapid processing (Lestienne 2001). In contrast, temporal coding leverages the precise timing of spikes to encode information, a mechanism that facilitates swift computation and is observed in the human visual system (Thorpe, Fize, and Marlot 1996), the rat hippocampus (O'Keefe and Recce 1993), and mouse retinal ganglion cells (Portelli et al. 2016). A prominent temporal scheme is Time-to-First-Spike (TTFS) coding (Rueckauer and Liu 2018; Mostafa et al. 2017; Kheradpisheh and Masquelier 2020). By constraining each neuron to fire at most once, TTFS theoretically achieves minimal energy consumption. This single-spike constraint fosters extreme sparsity, making TTFS a highly promising and efficient coding strategy for neuromorphic hardware.

Despite its theoretical promise, the practical application of TTFS coding is impeded by two critical challenges. First, the high sparsity of TTFS often fails to translate into correspondingly fast execution time. To compensate for the limited representational capacity of the single-spike constraint, existing approaches typically require many simulation time-steps (Rueckauer and Liu 2018; Stanojevic et al. 2023), which leads to a near-linear increase in execution time on Graphics Processing Units (GPUs). Recent Recurrent Neural Network (RNN)-like frameworks (Yang et al. 2024; Che et al. 2024) manage to reduce time-steps by enforcing serialized forward propagation. This design stops them from using modern GPUs' massive parallelism, often leading to slower execution than parallelizable rate-based SNNs (Fang et al. 2023b; Huang et al. 2024). Second, the task performance of directly trained TTFS SNNs generally lags behind competitive alternatives. The core difficulty arguably stems from the severe information bottleneck imposed by the single-spike constraint, which complicates the network training process.

To tackle the issue of slow execution, we derive the corresponding forward and backward propagation equations for parallelized TTFS operations, enables efficient and lossless calculation of first-spike timings, fully leveraging the massively parallel computing power of modern GPUs. Second, to bridge the performance gap, we theoretically analyze why TTFS accuracy lags rate-based methods and identify the output decoder as a key bottleneck. We then introduce a new membrane potential-based decoder, complemented by a training strategy that incrementally increases simulation time-steps to boost model accuracy. In summary, our main contributions are:

*Wei Fang and Yonghong Tian are corresponding authors.

- **A parallel training framework for TTFS SNNs** that significantly accelerates both training and inference by enabling efficient parallel computation on GPUs.
- **A novel membrane potential-based decoder and an incremental time-step training strategy**, which collectively boost the task accuracy of TTFS models to new state-of-the-art levels.

Related Work

Neural Coding in SNNs

SNNs primarily employ two neural coding strategies: rate coding and temporal coding. Rate coding represents information through the average firing rate of neurons and is common in both ANN-to-SNN conversion and direct training methods (Rueckauer et al. 2017; Han, Srinivasan, and Roy 2020; Li et al. 2021; Wu et al. 2024). Its main drawback, however, is the inherent accuracy-latency trade-off, which requires extended time-steps for reliable spike rate estimation. Temporal coding offers a more efficient alternative by encoding information in the precise timing of individual spikes. Various temporal schemes exist, such as phase coding (Kim et al. 2018; Chen et al. 2021) and burst coding (Park et al. 2019; Li and Zeng 2022). Among them, TTFS coding is particularly promising. By constraining each neuron to a single spike, TTFS theoretically achieves the lowest possible energy consumption (Kheradpisheh and Masquelier 2020; Kheradpisheh et al. 2022). Recent advancements in TTFS have focused on direct training using RNN-like frameworks, which significantly reduce latency compared to ANN-to-SNN conversion techniques (Rueckauer and Liu 2018; Zhang et al. 2019). For example, TQ-TTFS (Yang et al. 2024) proposed a time-quantized framework, while ETTFS (Che et al. 2024) improved the training of deep TTFS networks with a novel weight initialization and decoder. Despite their efficiency, these RNN-like models are fundamentally limited by their serial forward propagation mechanism, which slows down execution.

Parallel Training of SNNs

While SNNs with RNN-like architectures have achieved impressive results (Fang et al. 2021; Bellec et al. 2018; Yin, Corradi, and Bohté 2021), their training speed is constrained by an inherent architectural limitation: the reliance on sequential forward propagation. This prevents them from fully leveraging the massive parallel computing power of GPUs, creating a significant computational bottleneck.

To unlock parallel computation in SNNs, one line of research has focused on modifying the neuron model itself. Initial approaches, such as the Parallel Spiking Neuron (PSN) (Fang et al. 2023b) and the Stochastic Parallelizable Spiking Neuron (Yarga and Wood 2023), achieved parallelization by omitting the neuron's reset mechanism. This simplification, however, introduced inaccuracies in neuronal dynamics and led to undesirably high firing rates. Subsequent works sought to correct these issues. The Parallel Spiking Unit (PSU) (Li et al. 2024) introduced a probabilistic reset, and the Parallel Multi-compartment Spiking Neuron (PMSN) (Chen et al. 2024) used a more complex structure to handle temporal information. Despite improving accuracy, these solutions increase the model's internal complexity, which can impede overall execution speed.

Another prominent research direction integrates SNNs with State Space Models (SSMs). These methods, such as SpikingLMUFormer (Liu et al. 2024) and Spiking S4 (Du, Liu, and Chua 2024; Stan and Rhodes 2024), have successfully enabled parallel training. However, a major drawback of many SSM-based approaches is their tendency to neglect or oversimplify crucial neuronal dynamics, particularly the deterministic reset mechanism. Although a recent model attempts to approximate these dynamics with a surrogate network (Shen et al. 2025), effectively modeling them within a parallel framework remains an open challenge.

Method

At-Most-One-Spike Spiking Neuron Model

The At-Most-One-Spike (AMOS) spiking neuron model can be described as (Kheradpisheh and Masquelier 2020):

$$H[t] = f(H[t-1], X[t]), \tag{1}$$

$$S[t] = (1 - F[t-1]) \cdot \Theta(H[t] - V_{\text{th}}), \tag{2}$$

$$F[t] = F[t-1] + S[t], \tag{3}$$

where t is the time-step, $H[t]$ is the membrane potential after charging, $X[t]$ is the input current, f is the neuronal charging function, $F[t]$ is the firing mask, V_{th} is the threshold, and $\Theta(x)$ is the Heaviside step function with $\Theta(x) = 1$ for $x \geq 0$ and $\Theta(x) = 0$ for $x < 0$. The initial states are $H[-1] = F[-1] = 0$. When $S[i] = 1$ at time-step i, $F[i] = 1$ prevents subsequent firing to achieve the at-most-one-spike property. For the AMOS LIF neuron, the neural charge function f is identical to the standard LIF neuron:

$$H[t] = (1 - \frac{1}{\tau_{\text{m}}}) \cdot H[t-1] + \frac{1}{\tau_{\text{m}}} \cdot X[t], \tag{4}$$

where τ_{m} is the membrane time constant.

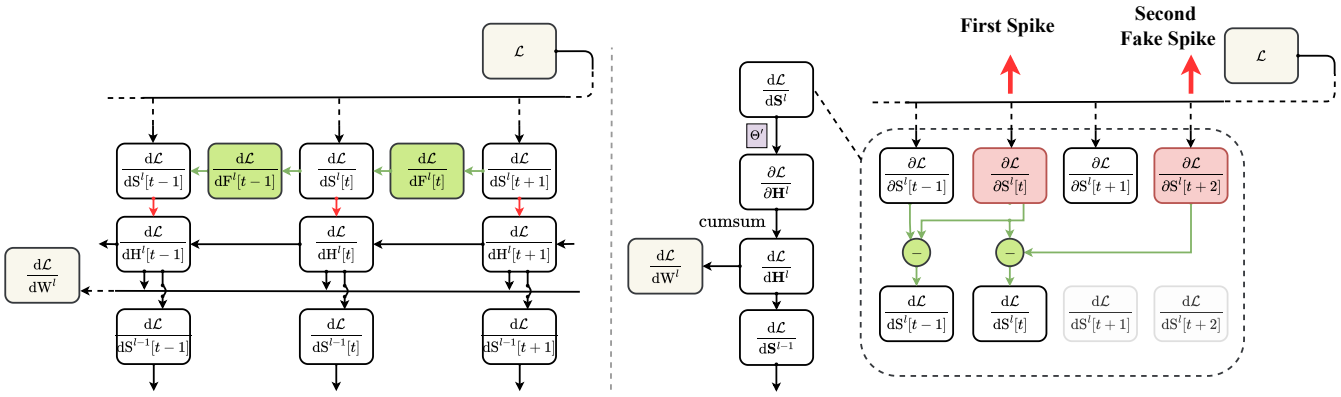
AMOS Neuron Forward Parallelization

Parallelizing the neural charge function. Unlike traditional spiking neurons, AMOS neurons naturally do not require the reset stage because they can only fire once. Therefore, the absence of neuronal reset allows exact parallelization without approximation. More specifically, Eq. (4) does not depend on the neuronal firing (Eq. (2)) or the firing mask update (Eq. (3)). We can easily rewrite Eq. (4) into a non-iterative formulation:

$$\boldsymbol{H} = \frac{1}{\tau_{\text{m}}} \boldsymbol{A} \cdot \text{cumsum}(\boldsymbol{A}^{-} \odot \boldsymbol{X}), \tag{5}$$

where $\boldsymbol{A} = [\alpha^0, \alpha^1, \ldots, \alpha^{T-1}]$, $\boldsymbol{A}^{-} = [\alpha^0, \alpha^{-1}, \ldots, \alpha^{1-T}]$, $\alpha = 1 - \frac{1}{\tau_{\text{m}}}$, $\boldsymbol{X} = [X[0], X[1], \ldots, X[T-1]]$, and $\odot$ is element-wise multiplication. The operator cumsum is defined as:

$$\text{cumsum}([x[0], \ldots, x[T-1]]) = [\sum_{k=0}^{0} x[k], \ldots, \sum_{k=0}^{T-1} x[k]]. \tag{6}$$



(a) AMOS neuron backward (b) AMOS neuron parallel backward

Figure 1: Backward process of the At-Most-One-Spike (AMOS) neuron. (a) Sequential execution. (b) We decompose the backward process into different stages and solve them in parallel by matrix operations, rather than step-by-step.

Note that the cumsum operation can be implemented efficiently with nearly $\mathcal{O}(\log(T))$ time complexity on GPUs.

First spike extraction. With membrane potentials $\boldsymbol{H} = [H[0], H[1], \dots, H[T-1]]$, we can compute spikes $\boldsymbol{S} = [S[0], S[1], \dots, S[T-1]]$ by searching for the first time-step when the membrane potential is greater than or equal to the threshold. Particularly, to avoid the no-spike problem, the AMOS neuron is generally forced to fire a spike at the last time-step if it has not fired at previous time-steps. Thus, $S[t]$ is computed as:

$$S[t] = \begin{cases} 1, t = t_{\mathrm{f}} = \mathrm{argmin}\{i | H[i] \geq V_{\mathrm{th}} \text{ or } i = T-1\} \\ 0, t \neq t_{\mathrm{f}} \end{cases}, \tag{7}$$

where t_{f} is the firing time-step. To generate $S[t]$ in parallel, we first ignore the one-spike constraint and compute fake spikes $\tilde{\boldsymbol{S}}$ as:

$$\tilde{\boldsymbol{S}} = \Theta(\boldsymbol{H} - V_{\mathrm{th}}). \tag{8}$$

Then we replace $\tilde{S}[T-1]$ with 1 and obtain t_{f} by the argmax operation:

$$t_{\mathrm{f}} = \mathrm{argmax}\{\tilde{S}[0], \tilde{S}[1], \dots, \tilde{S}[T-2], 1\}. \tag{9}$$

Note that argmax operation always returns the first index of 1 in Eq. (9). Finally, we can generate $\boldsymbol{S}$ from t_{f} as in Eq. (7).

AMOS Neuron Backward Parallelization

The argmax operation creates a gradient discontinuity. Using Eqs. (8)-(9) directly in the autograd mechanism would truncate the computation graph, preventing effective back-propagation. We address this in the following by redefining a parallelizing backward propagation.

BPTT formulation. Denote the loss as $\mathcal{L}$. The gradient of the membrane potential is:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}H[t]} = \underbrace{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t]}\frac{\partial S[t]}{\partial H[t]}}_{\text{spatial gradient}} + \underbrace{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}H[t+1]}\frac{\partial H[t+1]}{\partial H[t]}}_{\text{temporal gradient}}, \tag{10}$$

where $\frac{\partial S[t]}{\partial H[t]} = (1 - F[t-1])\Theta'(H[t] - V_{\mathrm{th}})$ and $\frac{\partial H[t+1]}{\partial H[t]} = \frac{\partial f}{\partial H[t]}$. Note that $\Theta'(x)$ is defined by the surrogate gradient (Neftci, Mostafa, and Zenke 2019). The critical term $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t]}$ decomposes as:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t]} = \frac{\partial \mathcal{L}}{\partial S[t]} + \underbrace{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t]}\frac{\partial F[t]}{\partial S[t]}}_{\text{mask gradient}} = \frac{\partial \mathcal{L}}{\partial S[t]} + \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t]}. \tag{11}$$

The mask gradient propagates via:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t]} &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t+1]}\frac{\partial F[t+1]}{\partial F[t]} + \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t+1]}\frac{\partial S[t+1]}{\partial F[t]} \\ &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t+1]} - \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t+1]}\Theta(H[t+1] - V_{\mathrm{th}}). \end{aligned} \tag{12}$$

Gradients change by the second fake spike. It can be found that $\Theta(H[t+1] - V_{\mathrm{th}}) = \tilde{S}[t+1]$ is the spike without the one-spike constraint. If $\tilde{S}[t+1] = 0$, then:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t]} = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t+1]}. \tag{13}$$

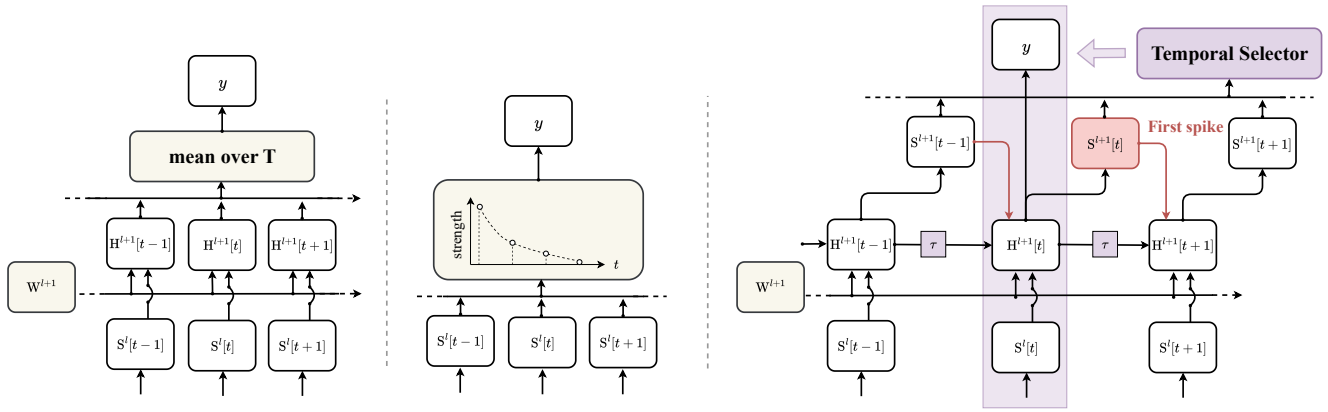
If $\tilde{S}[t+1] = 1$, then:

$$\begin{aligned} \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t]} &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t+1]} - \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t+1]} \\ &= \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t+1]} - (\frac{\partial \mathcal{L}}{\partial S[t+1]} + \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}F[t+1]}) \\ &= -\frac{\partial \mathcal{L}}{\partial S[t+1]}. \end{aligned} \tag{14}$$

Eqs. (13)-(14) reveal that $\frac{\partial \mathcal{L}}{\partial F[t]}$ propagates unchanged until the a **fake** spike occurs.

We therefore define the *second fake spike* as the temporally closest fake spike immediately following the first actual spike. We conclude that the complete spike gradient is:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t]} = \begin{cases} 0 & , t > t_{\mathrm{f}} \\ \frac{\partial \mathcal{L}}{\partial S[t_{\mathrm{f}}]} - \frac{\partial \mathcal{L}}{\partial S[t_{\mathrm{sf}}]} & , t = t_{\mathrm{f}} \\ \frac{\partial \mathcal{L}}{\partial S[t]} - \frac{\partial \mathcal{L}}{\partial S[t_{\mathrm{f}}]} & , t < t_{\mathrm{f}} \end{cases}, \tag{15}$$



(a) Rate-based decoder **(b) Temporal weighting decoding** **(c) Temporal selector decoder**

Figure 2: A comparison of output decoding mechanisms. (a) Rate coding utilizes temporally-aggregated spike counts. (b) Temporal Weighting Decoding (TWD) (Che et al. 2024) fails to resolve spikes that co-occur in the same time-step. (c) Our proposed temporal selector decoder identifies the first spike time ($t_{1\text{st}}$) across the layer and uses the membrane potential snapshot at that instant for classification.

where t_{f} is the firing time-step and t_{sf} is the *second fake spike* time-step. Notably, both $-\frac{\partial \mathcal{L}}{\partial S[t_{\text{f}}]}$ and $-\frac{\partial \mathcal{L}}{\partial S[t_{\text{sf}}]}$ terms exhibit no temporal dependencies, enabling $\mathcal{O}(1)$ parallel computation, denoted $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{S}}$. The detailed process is shown in Figure 1. Building upon the preceding derivation, the complete membrane potential gradient is formulated as follows:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}H[t]} = \underbrace{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t]}(1 - F[t-1])\Theta'[t]}_{\text{spatial gradient}} + \underbrace{\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}H[t+1]}\frac{\partial f}{\partial H[t]}}_{\text{temporal gradient}}. \tag{16}$$

The *second fake spike* formulation enables $\mathcal{O}(1)$ parallel computation of $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}S[t]}$, denoted $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{S}}$. The term $(1 - F[t-1])$ represents that the gradient is set to zero when $t > t_{\text{f}}$. Once the firing time is determined, a simple mask operation can be generated as $\boldsymbol{F} = [F[0], F[1], \ldots, F[T-1]]$ with $\boldsymbol{F}[t \leq t_{\text{f}}] = 1$ and $\boldsymbol{F}[t > t_{\text{f}}] = 0$. The $\Theta'(H[t] - V_{\text{th}})$ term exhibits no temporal dependencies, facilitating parallel computation as $\boldsymbol{\Theta'}$. Consequently, the spatial gradient $\mathcal{G}_s$ is computed with $\mathcal{O}(1)$ parallel complexity:

$$\boldsymbol{\mathcal{G}}_s = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{S}} \odot \boldsymbol{F} \odot \boldsymbol{\Theta'}. \tag{17}$$

While $\frac{\partial f}{\partial H[t]}$ varies across neuron types, it remains temporally constant $\frac{\partial f}{\partial H[t]} = \tau_{\text{m}}$ for the AMOS LIF neuron. The membrane potential gradient recursion simplifies to:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}H[t]} = \mathcal{G}_s[t] + \alpha \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}H[t+1]}, \tag{18}$$

where $\alpha = 1 - \frac{1}{\tau_{\text{m}}}$. As suffix sums are required for backward propagation while cumsum operation efficiently compute prefix sums, we optimize Eq. (18) through temporal reversal:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{H}} = \mathrm{Rev}(\mathrm{cumsum}(\mathrm{Rev}(\boldsymbol{A}) \odot \boldsymbol{\mathcal{G}}_s)), \tag{19}$$

where the Rev operator is defined as:

$$\mathrm{Rev}([x[0], \ldots, x[T-1]]) = [x[T-1], \ldots, x[0]]. \tag{20}$$

Temporal Selector Decoder

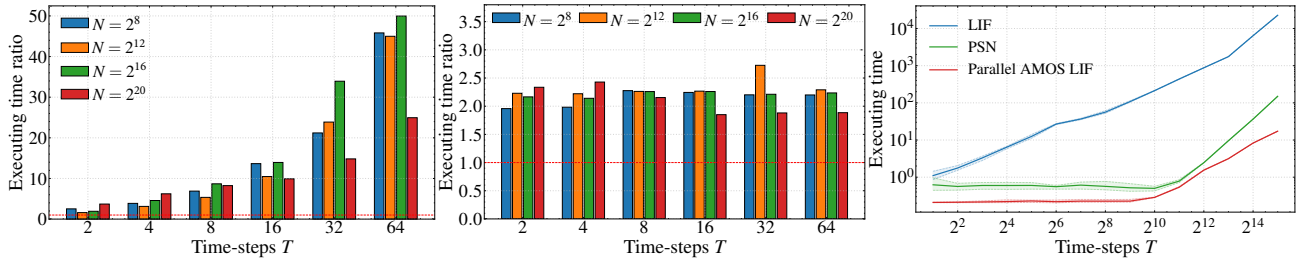
Theoretical basis for decoder. The efficacy of existing TTFS decoders, such as the decoder in TQ-TTFS (Yang et al. 2024) and Temporal Weighting Decoding (TWD) (Che et al. 2024), is largely confined to simple datasets like Fashion-MNIST. On more complex, larger-scale benchmarks like CIFAR-100, their performance collapses. As shown in our ablation study (Table 4), TWD's accuracy on CIFAR-100 drops to a mere 60.18%, while the TQ-TTFS decoder fails to converge altogether.

This empirical failure presents a paradox. At an inference time of $T = 1$, TTFS and rate-based schemes are theoretically analogous, since each neuron can fire at most once. The poor performance of TTFS decoders therefore suggests a fundamental flaw in how they interpret information. We hypothesize that this issue stems from their reliance on sparse temporal features, such as the timing of the first spike. In contrast, conventional rate-based decoders implicitly operate on dense, aggregate information, like the mean membrane potential of the output layer, which provides a more robust signal for classification. This conceptual difference is visualized in Figure 2. Motivated by this analysis, we contend that a robust and efficient TTFS decoder must be designed around three core principles. It must:

P1. Guarantee a strong baseline by being functionally equivalent to a rate-based decoder at the minimal inference time ($T = 1$).

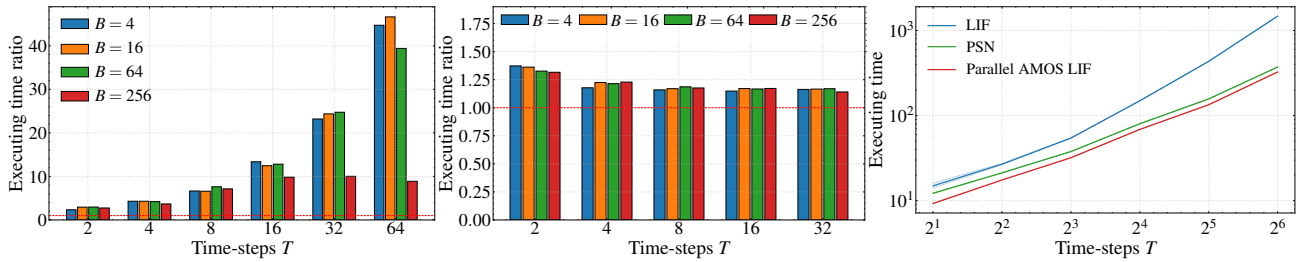
P2. Preserve the primacy of early information by prioritizing the first spikes, thereby supporting adaptive early stopping capabilities—a hallmark advantage of TTFS.

P3. Promote a general trend of performance improvement as inference time (T) increases, thus making productive use of additional temporal data.



(a) Ratio of LIF to Parallel AMOS LIF (b) Ratio of PSN to Parallel AMOS LIF (c) Executing time with fixed $N = 2^{10}$

Figure 3: Isolated neuron benchmark of the Parallel AMOS LIF, PSN, and LIF neuron. N is the number of neurons.



(a) Ratio of LIF to Parallel AMOS LIF (b) Ratio of PSN to Parallel AMOS LIF (c) Executing time with fixed $B = 16$

Figure 4: Full network benchmark of the Parallel AMOS LIF, PSN, and LIF neuron in the same model (Fang et al. 2023b).

Decoder formulation. We propose the **temporal selector decoder**, designed to satisfy these three properties. The decoder comprises a synaptic layer and a neuron layer, generating both spikes $S^{l+1}[t]$ and membrane potentials $H^{l+1}[t]$ as shown in Figure 2 (c). We apply channel-wise learnable time constants τ to control membrane decay dynamics. To achieve properties P1 and P2, we introduce a temporal selector, which identifies the time of the first spike across all neurons, denoted as $t_{1\text{st}}$, and then retrieves the membrane potentials of all neurons at that time. If no spikes are emitted, $t_{1\text{st}}$ is the final time-step instead. During inference, the neuron at $t_{1\text{st}}$ with the highest membrane potential is selected for classification. During training, the membrane potentials are treated as classification probabilities for computing the loss. When $T = 1$, since $t_{1\text{st}}$ is always 1, the decoder directly outputs the membrane potentials, achieving equivalence with rate coding (for P1). For $T > 1$, the temporal selector prevents the network from using information after the first spike ($t_{1\text{st}}$), thereby preserving first-priority and enabling early stopping as P2.

For progressive accuracy scaling as P3, we gradually increase T during training. When $T = 1$, the network achieves the rate coding baseline. For $T > 1$, the temporal selector is employed during training, with the firing threshold V_{th} serving as a confidence parameter. We call it a confidence parameter because it can control when the execution can stop. Higher V_{th} requires a larger membrane potential value to fire. By setting a suitable confidence parameter V_{th}, we ensure a smooth and stable improvement in network performance, approaching its theoretical limit. Further training details are provided in the Supplementary Materials.

Experiments

This section evaluates our parallel AMOS LIF neuron in terms of both simulation speed and classification accuracy. All experiments are conducted using the SpikingJelly framework (Fang et al. 2023a).

Simulation Speed Benchmark

Isolated neuron benchmark. First, we benchmark the isolated neuron's performance in training mode, following the method of PSN (Fang et al. 2023b) by varying the neuron count N and the number of time-steps T, corresponding evaluation in inference mode is reported in the Supplementary Materials. We report the execution time ratios of the conventional LIF neuron to our Parallel AMOS LIF neuron in Figure 3(a) and the PSN neuron to our Parallel AMOS LIF neuron in Figure 3(b). The absolute execution times at a fixed neuron count of $N = 2^{10}$ are shown in Figure 3(c). In all tested scenarios, the Parallel AMOS LIF neuron executes substantially faster than both the LIF neuron and PSN. Notably, the performance gap between the standard LIF neuron and our Parallel AMOS LIF neuron widens as the number of time-steps T increases.

Furthermore, Figure 3(c) highlights a *computation-bound region* where all parallel methods approach $\mathcal{O}(1)$ complexity. This phenomenon occurs when the GPU workload becomes limited by its computational capacity rather than memory bandwidth, as described by the Roofline model

Method	MNIST		Fashion-MNIST		CIFAR-10		CIFAR-100	
	Model	Acc.	Model	Acc.	Model	Acc.	Model	Acc.
Conversion								
TTFS clamped (Rueckauer and Liu 2018)	LeNet-5	98.53%	-	-	-	-	-	-
LC-TTFS (Yang et al. 2023)	-	-	-	-	ResNet-20	92.72%	ResNet-20	72.36%
T2FSNN (Park et al. 2020)	-	-	-	-	VGG-16	91.43%	VGG-16	68.79%
Directly training								
Mostafa (Mostafa 2017)	FC800-FC10	97.5%	-	-	-	-	-	-
Comsa et al. (Comsa et al. 2020)	FC340-FC10	97.9%	-	-	-	-	-	-
Göltz et al. (Göltz et al. 2021)	FC350-FC10	97.2%	-	-	-	-	-	-
S4NN (Kheradpisheh and Masquelier 2020)	FC600-FC10	97.4%	FC1000-FC10	88.0%	-	-	-	-
BS4NN (Kheradpisheh et al. 2022)	FC600-FC10	97.0%	FC1000-FC10	87.3%	-	-	-	-
Zhang et al. (Zhang et al. 2021)	FC400-FC10	98.1%	SCNN[1]	90.1%	-	-	-	-
Sakemi et al. (Sakemi et al. 2023)	SCNN[2]	99.25%*	SCNN[2]	89.5%*	SCNN[3]	80.0%*	-	-
STiDi-BP (Mirsadeghi et al. 2023)	SCNN[4]	99.2%	SCNN[5]	92.8%	-	-	-	-
DTA-TTFS (Wei et al. 2023)	SCNN[1]	99.4%	-	-	VGG16	93.05%	VGG16	69.66%
TQTTFS (Yang et al. 2024)	FC400-FC10	98.6%	SCNN[1]	90.2%	SCNN[6]	67.47%†	-	-
ETTFS (Che et al. 2024)	SCNN[1]	99.48%	SCNN[5]	92.9%	SCNN[6]	90.56%	SCNN[6]	62.4%
Parallel AMOS LIF	SCNN[1]	**99.51%**	SCNN[5]	**93.14%**	SCNN[6]	**95.06%**	SCNN[6]	**74.07%**

Table 1: Comparison on MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. Some prior works report results only in figures (denoted by *). The symbol † denotes our implementation of TQTTFS (Yang et al. 2024). Refer to Supplementary Materials for more details about symbolic representations of the network structure.

(Williams, Waterman, and Patterson 2009). Analyzing execution times within this region reveals the intrinsic computational cost of each neuron model, free from memory access constraints. For instance, at $T = 2^8$ within this region, if we normalize the execution time of the Parallel AMOS LIF neuron to 1.0, the relative time for PSN is 2.18, while for the LIF neuron, it is a striking 2381.30.

Full network benchmark. To evaluate performance in a practical application, we benchmarked a spiking convolutional network (Fang et al. 2023b) with input dimensions of $[T, B, 3, 32, 32]$, where T is the number of time-steps and B is the batch size. We varied both T and B to analyze the time complexity under different conditions. The results, depicted in Figure 4, again show the superiority of our method. We report the execution time ratios of the network using LIF versus Parallel AMOS LIF neurons (a) and PSN versus Parallel AMOS LIF neurons (b), alongside the absolute execution times for a fixed batch size of 16 (c). These findings confirm that our approach provides a significant speedup over the standard LIF neuron and a substantial performance gain even when compared to the highly optimized PSN.

Comparison with State-of-the-Art Methods

Accuracy. As shown in Table 1, we compare our method with both conversion-based and direct-training approaches for TTFS SNNs. Our framework outperforms all existing direct-training TTFS methods across the tested datasets, achieving state-of-the-art accuracy of 95.06% on CIFAR-10 and 74.07% on CIFAR-100. It is worth noting that a prominent competing method, DTA-TTFS (Wei et al. 2023), requires an auxiliary, pre-trained ANN to guide its training process, whereas our method trains end-to-end.

Furthermore, our method's accuracy is superior even to some conversion-based methods using much larger VGG-16 network (138.36M parameters) on CIFAR, while our model contains only 68.93M parameters. Moreover, these conversion methods typically necessitate a large number of time-steps (e.g., $T \geq 1000$) to reach high accuracy, a requirement that is often impractical for real-time applications. In contrast, our method achieves competitive accuracy with significantly fewer time-steps, making it far more suitable for efficient, real-time inference on neuromorphic hardware.

Method	MNIST	Fashion-MNIST	CIFAR-10	CIFAR-100
T2FSNN	20*	–	400*	–
Mostafa	3.5*	–	–	–
Comsa et al.	20*	–	–	–
S4NN	89.7	–	–	–
BS4NN	112.2*	100.20*	–	–
STiDi-BP	71.1	61.3	–	–
DTA-TTFS	160	–	160	160
TQ-TTFS	2.93†	4.21†	4.62†	5.12†
ETTFS	1.01	2.03	2.93	3.15
Ours	**1.01**	**1.83**	**1.53**	**1.48**

Table 2: Comparison of time-steps on MNIST, Fashion-MNIST, CIFAR-10, and CIFAR-100. Each element denotes: inference time-steps. Some prior works report results only in figures (denoted by *). † denotes our implementation.

Latency and energy consumption. A key advantage of step-by-step TTFS SNNs, including ours, over other layer-by-layer TTFS SNNs is their lower latency (Che et al. 2024). This step-by-step mode is particularly well-suited for neuromorphic hardware as it allows for early stopping of the inference process once a decision can be made. Consequently, this often leads to a significantly lower number of required time-steps compared to conventional SNNs, which typically

Methods	# Spikes	E1 (pJ)	E2 (Normalized)
LIF	6.64×10^9	40.97	$1\times$
PSN	5.34×10^9	44.55	$0.81\times$
Ours	$\mathbf{2.03 \times 10^9}$	**22.31**	$\mathbf{0.31\times}$

Table 3: Energy estimation comparison among different methods using the same architecture (Fang et al. 2023b) and same time-steps ($T = 4$) on CIFAR-100. # Spikes denotes number of spikes. E1 and E2 are two different energy estimation metrics, detailed in Supplementary Materials.

operate over a fixed and often lengthy period. We report the average inference time-steps for various methods in Table 2. We also report the energy estimation comparison among different methods on CIFAR-100 as shown in Table 3. Our model, featuring TTFS coding and a temporal selector decoder, substantially reduces the average inference latency and number of spikes, thereby enabling faster inference and lower theoretical energy consumption than prior works.

Ablation Study

To validate the contribution of each component in our proposed framework, which achieves high accuracy with low latency, we conduct a series of ablation studies.

Decoder comparison. We first evaluate the impact of different decoding mechanisms on performance. Using the CIFAR-100 dataset, we compare our proposed temporal selector decoder against several alternatives: TWD (Che et al. 2024), the TQ-TTFS decoder (Yang et al. 2024), and standard rate coding. All decoders were integrated into the same base network architecture and trained with identical hyperparameters as in (Fang et al. 2023b). The results are summarized in Table 4. The results reveal significant performance disparities among the decoders. Both the TWD and TQ-TTFS decoder struggled on the complex CIFAR-100 dataset, with the latter failing to converge entirely. TWD achieved only 60.18% accuracy and, like TQ-TTFS decoder, also suffered from convergence issues, particularly with fewer time-steps. This suggests that while these simpler decoding methods may be effective for datasets with fewer classes and samples (e.g., Fashion-MNIST), their performance degrades substantially on more challenging tasks.

In contrast, both our proposed temporal selector decoder and the standard rate coding decoder performed robustly. At a minimal latency of $T = 1$, both methods achieved comparable performance. As the number of time-steps increase, both decoders saw corresponding improvements in accuracy, demonstrating that our decoder not only possesses superior convergence properties on complex datasets but also achieves a performance level on par with the well-established rate coding method, validating its effectiveness.

Impact of decoder confidence threshold. We evaluated the effect of decoder confidence threshold V_{th} on the distribution of first-spike times ($t_{1\text{st}}$) and accuracy on the CIFAR-100 dataset (with $T = 4$), as detailed in Table 5. The analysis shows that a minimal threshold (e.g., $V_{\text{th}} = 1$) causes premature classifications, resulting in the lowest overall performance. Increasing V_{th} systematically shifts classifications to later time-steps, as a higher membrane potential must be accumulated to reach the confidence requirement. This experiment also highlights that our decoder adaptively allocates computational resources: easy samples are classified quickly and accurately in early time-steps, while difficult samples require more integration time. This demonstrates that our temporal selector decoder effectively distinguishes between sample difficulties, thereby optimizing the balance between accuracy and energy consumption.

T	1	2	4	8
Rate	$\mathbf{71.8}\%$	73.82%	73.64%	73.82%
TQ-TTFS decoder	1%	1%	1%	1%
TWD	1%	53.82%	60.18%	57.27%
TSD (Ours)	71.71%	$\mathbf{74.04}\%$	$\mathbf{74.07}\%$	$\mathbf{74.04}\%$

Table 4: Comparison among various decoders across different number of time-steps T on CIFAR-100. TWD is temporal weighting decoding in ETTFS (Che et al. 2024). TQ-TTFS decoder is proposed by Yang et al. (Yang et al. 2024). TSD denotes the proposed temporal selector decoder.

V_{th} \ $t_{1\text{st}}$	0		1		2		3		Accuracy
1	100%	72.94%	0%	0%	0%	0%	0%	0%	72.94%
2	97.7%	74.24%	2.3%	**52.50%**	0%	0%	0%	0%	73.74%
5	89.6%	78.41%	7.3%	35.93%	2.6%	27.7%	0.5%	14.31%	73.67%
10	81.7%	**82.8%**	3.9%	47.0%	1.7%	**42.6%**	12.7%	**30.4%**	**74.07%**

Table 5: Analysis of the relationship between the decoder confidence threshold (V_{th}), classification latency (distribution of $t_{1\text{st}}$), and accuracy on CIFAR-100 ($T = 4$). The gray bars show the cumulative percentage of samples classified at each time-step. The final accuracy for a given V_{th} is shown at the end of each row (e.g., 72.94% accuracy when 100% of samples are classified with $t_{1\text{st}} = 0$).

Conclusion

In this paper, we addressed two critical challenges hindering the practical application of TTFS SNNs: their often-paradoxical slow execution speed despite inherent sparsity, and a persistent accuracy gap compared to other neural network paradigms. To overcome these limitations, we introduced two primary contributions. First, we proposed a novel parallel forward and backward propagation algorithm that significantly accelerates the training phase without introducing any approximation. Second, we developed a new temporal selector decoder, which, when combined with a progressive time-step training methodology, substantially boosts classification performance. Our extensive experiments, conducted across multiple benchmark datasets, validate the effectiveness of our approach, demonstrating state-of-the-art accuracy and superior computational efficiency.

Acknowledgments

The study was funded by the National Natural Science Foundation of China under contracts No. 62425101, No. 62332002, No. 62027804, No.62088102, and the major key project of the Peng Cheng Laboratory (PCL2025A02).

References

Adrian, E. D. 1926. The impulses produced by sensory nerve endings: Part I. *The Journal of physiology*, 61(1): 49.

Bellec, G.; Salaj, D.; Subramoney, A.; Legenstein, R.; and Maass, W. 2018. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31.

Che, K.; Fang, W.; Ma, Z.; Yuan, L.; Masquelier, T.; and Tian, Y. 2024. Efficiently Training Time-to-First-Spike Spiking Neural Networks from Scratch. *arXiv e-prints*, arXiv–2410.

Chen, X.; Wu, J.; Ma, C.; Yan, Y.; Wu, Y.; and Tan, K. C. 2024. PMSN: A Parallel Multi-compartment Spiking Neuron for Multi-scale Temporal Processing. *arXiv preprint arXiv:2408.14917*.

Chen, Y.; Qu, H.; Zhang, M.; and Wang, Y. 2021. Deep Spiking Neural Network with Neural Oscillation and Spike-Phase Information. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8): 7073–7080.

Comsa, I. M.; Potempa, K.; Versari, L.; Fischbacher, T.; Gesmundo, A.; and Alakuijala, J. 2020. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8529–8533. IEEE.

Davies, M.; Wild, A.; Orchard, G.; Sandamirskaya, Y.; Guerra, G. A. F.; Joshi, P.; Plank, P.; and Risbud, S. R. 2021. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5): 911–934.

Du, Y.; Liu, X.; and Chua, Y. 2024. Spiking structured state space model for monaural speech enhancement. In *2024 IEEE International Conference on Acoustics, Speech and Signal Processing*, 766–770. IEEE.

Fang, W.; Chen, Y.; Ding, J.; Yu, Z.; Masquelier, T.; Chen, D.; Huang, L.; Zhou, H.; Li, G.; and Tian, Y. 2023a. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40): eadi1480.

Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; and Tian, Y. 2021. 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.

Fang, W.; Yu, Z.; Zhou, Z.; Chen, D.; Chen, Y.; Ma, Z.; Masquelier, T.; and Tian, Y. 2023b. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. *Advances in Neural Information Processing Systems*, 36.

Göltz, J.; Kriener, L.; Baumbach, A.; Billaudelle, S.; Breitwieser, O.; Cramer, B.; Dold, D.; Kungl, A. F.; Senn, W.; Schemmel, J.; et al. 2021. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature machine intelligence*, 3(9): 823–835.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13558–13567.

Huang, Y.; Liu, Z.; Feng, C.; Lin, X.; Ren, H.; Fu, H.; Zhou, Y.; Xing, H.; and Cheng, B. 2024. PRF: Parallel Resonate and Fire Neuron for Long Sequence Learning in Spiking Neural Networks. arXiv:2410.03530.

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.

Kheradpisheh, S. R.; Mirsadeghi, M.; Masquelier, T.; et al. 2022. BS4NN: Binarized spiking neural networks with temporal coding and learning. *Neural Processing Letters*, 54(2): 1255–1273.

Kim, J.; Kim, H.; Huh, S.; Lee, J.; and Choi, K. 2018. Deep neural networks with weighted spikes. *Neurocomputing*, 311: 373–386.

Lestienne, R. 2001. Spike timing, synchronization and information processing on the sensory side of the central nervous system. *Progress in neurobiology*, 65(6): 545–591.

Li, B.; Leng, L.; Shen, S.; Zhang, K.; Zhang, J.; Liao, J.; and Cheng, R. 2024. Efficient Deep Spiking Multilayer Perceptrons With Multiplication-Free Inference. *IEEE Transactions on Neural Networks and Learning Systems*.

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. PMLR.

Li, Y.; and Zeng, Y. 2022. Efficient and Accurate Conversion of Spiking Neural Network with Burst Spikes.

Liu, Z.; Datta, G.; Li, A.; and Beerel, P. A. 2024. LMU-Former: Low Complexity Yet Powerful Spiking Model With Legendre Memory Units. *arXiv preprint arXiv:2402.04882*.

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.

Mirsadeghi, M.; Shalchian, M.; Kheradpisheh, S. R.; and Masquelier, T. 2023. Spike time displacement-based error backpropagation in convolutional spiking neural networks. *Neural Computing and Applications*, 35(21): 15891–15906.

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.

Mostafa, H.; Pedroni, B. U.; Sheik, S.; and Cauwenberghs, G. 2017. Fast classification using sparsely active spiking networks. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–4.

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.

O'Keefe, J.; and Recce, M. L. 1993. Phase relationship between hippocampal place units and the EEG theta rhythm. *Hippocampus*, 3(3): 317–330.

Park, S.; Kim, S.; Choe, H.; and Yoon, S. 2019. Fast and efficient information transmission with burst spikes in deep spiking neural networks. In *Proceedings of Annual Design Automation Conference (DAC)*, 1–6.

Park, S.; Kim, S.; Na, B.; and Yoon, S. 2020. T2FSNN: deep spiking neural networks with time-to-first-spike coding. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, DAC '20. IEEE Press. ISBN 9781450367257.

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.

Portelli, G.; Barrett, J. M.; Hilgen, G.; Masquelier, T.; Maccione, A.; Di Marco, S.; Berdondini, L.; Kornprobst, P.; and Sernagor, E. 2016. Rank Order Coding: a Retinal Information Decoding Strategy Revealed by Large-Scale Multielectrode Array Retinal Recordings. *eNeuro*, 3(3).

Rueckauer, B.; and Liu, S.-C. 2018. Conversion of analog to spiking neural networks using sparse temporal coding. In *2018 IEEE international symposium on circuits and systems (ISCAS)*, 1–5. IEEE.

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.

Sakemi, Y.; Yamamoto, K.; Hosomi, T.; and Aihara, K. 2023. Sparse-firing regularization methods for spiking neural networks with time-to-first-spike coding. *Scientific Reports*, 13(1): 22897.

Shen, S.; Wang, C.; Huang, R.; Zhong, Y.; Guo, Q.; Lu, Z.; Zhang, J.; and Leng, L. 2025. Spikingssms: Learning long sequences with sparse and parallel spiking state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 20380–20388.

Stan, M.-I.; and Rhodes, O. 2024. Learning long sequences in spiking neural networks. *Scientific Reports*, 14: 21957.

Stanojevic, A.; Woźniak, S.; Bellec, G.; Cherubini, G.; Pantazi, A.; and Gerstner, W. 2023. An exact mapping from ReLU networks to spiking neural networks. *Neural Networks*, 168: 74–88.

Thorpe, S.; Fize, D.; and Marlot, C. 1996. Speed of processing in the human visual system. *nature*, 381(6582): 520–522.

Wei, W.; Zhang, M.; Qu, H.; Belatreche, A.; Zhang, J.; and Chen, H. 2023. Temporal-Coded Spiking Neural Networks with Dynamic Firing Threshold: Learning with Event-Driven Backpropagation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 10552–10562.

Williams, S.; Waterman, A.; and Patterson, D. 2009. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4): 65–76.

Wu, K.; Yao, M.; Chou, Y.; Qiu, X.; Yang, R.; Xu, B.; and Li, G. 2024. RSC-SNN: Exploring the Trade-off Between Adversarial Robustness and Accuracy in Spiking Neural Networks via Randomized Smoothing Coding. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 2748–2756.

Yang, Q.; Zhang, M.; Wu, J.; Tan, K. C.; and Li, H. 2023. LC-TTFS: Towards lossless network conversion for spiking neural networks with TTFS coding. *IEEE Transactions on Cognitive and Developmental Systems*.

Yang, Y.; Xuan, Z.; Kang, Y.; et al. 2024. TQ-TTFS: High-Accuracy and Energy-Efficient Spiking Neural Networks Using Temporal Quantization Time-to-First-Spike Neuron. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 836–841. IEEE.

Yarga, S. Y. A.; and Wood, S. U. N. 2023. Accelerating SNN Training with Stochastic Parallelizable Spiking Neurons. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Yin, B.; Corradi, F.; and Bohté, S. M. 2021. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3: 905–913.

Zhang, L.; Zhou, S.; Zhi, T.; Du, Z.; and Chen, Y. 2019. Tdsnn: From deep neural networks to deep spike neural networks with temporal-coding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 1319–1326.

Zhang, M.; Wang, J.; Wu, J.; Belatreche, A.; Amornpaisannon, B.; Zhang, Z.; Miriyala, V. P. K.; Qu, H.; Chua, Y.; Carlson, T. E.; et al. 2021. Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5): 1947–1958.