PARALLEL TRAINING IN SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neurons mimic the spatiotemporal dynamics of biological neurons and their spike-based communication, endowing Spiking Neural Networks (SNNs) with biological plausibility and low-power operation. Yet these dynamics impose strict temporal dependencies on neuronal states, preventing parallel training and creating a fundamental bottleneck to efficient, scalable optimization. This work introduces a novel functional perspective to address this challenge. Specifically, we argue that the reset mechanism, which induces state dependencies, should be removed. However, any modification must satisfy two principles: i) preserving — and even enhancing — the functions of reset as a core biological mechanism; and ii) enabling parallel training without sacrificing SNNs' inherently serial inference, which underpins their energy efficiency. To this end, we identify functions of the reset mechanism and analyze how to reconcile parallel training with serial inference, upon which we propose a dynamic decay spiking neuron that combines a causal convolution structure with an optimized spike firing pattern. We demonstrate the efficiency and effectiveness of our approach across diverse network architectures and task benchmarks, including image classification, neuromorphic event processing, time-series forecasting, and language modeling.

1 INTRODUCTION

Spiking neurons incorporate information across spatial and temporal into a membrane potential, i.e., the neuronal state. If this potential surpasses a threshold, the neuron fires a spike and the potential is reset; otherwise, it decays (Maass, 1997). Thus, SNNs exhibit spike-based event-driven dynamics: sparse accumulations occur only upon spike transmissions between neurons, while the network stays idle otherwise (Roy et al., 2019). Deploying SNNs on neuromorphic hardware (Merolla et al., 2014; Davies et al., 2018; Pei et al., 2019) yields significant energy savings. For example, the asynchronous sensing-computing neuromorphic chip Speck consumes merely 0.42 mW at idle, and its dynamic power under typical vision scenarios can be kept within the mW range (Yao et al., 2024).

Directly training large-scale SNNs has long been a core challenge in the field. The progress can be viewed in three stages. i) Trainability under spike communication constraints: surrogate-gradient methods (Wu et al., 2018; Neftci et al., 2019) were proposed to handle the spike activation function, which is not differentiable, so that SNNs can be trained with backpropagation algorithm. ii) Going deeper without performance degradation: to reduce accuracy degradation in deeper SNNs, researchers introduced spiking residual connections (Fang et al., 2021a; Hu et al., 2025), new network designs (Zhou et al., 2023; Yao et al., 2024), various normalization methods (Zheng et al., 2021), and training optimization methods (Li et al., 2021; Guo et al., 2022). iii) Efficient training under complex spatiotemporal dynamic constraints: the goal is to study how to efficiently train larger SNNs under longer sequences, laying the foundation for directly training large spiking models.

Regarding the challenge mentioned in the third stage above, the reset mechanism prevents parallel training, which makes SNN training very costly. One line of work keeps reset but speeds up training by decoupling spatial and temporal dependencies, for example by dropping temporal dependence during backpropagation (Xiao et al., 2022; Meng et al., 2023), by letting only a subset of neurons carry temporal information (Hu et al., 2024; Xu et al., 2025), or by using single-step pretraining followed by multi-step fine-tuning (Lin et al., 2024; Yao et al., 2023b). Another line of work removes reset. PSN (Fang et al., 2023b) first took this direction and added a learnable parameter matrix along

the time dimension to compensate for the role of reset. Some subsequent studies have improved upon PSN, but the resulting models cannot support serial inference (Li et al., 2024; Su et al., 2024). Another idea is to eliminate reset and approximate the membrane potential of some spiking neuron via value approximation (Chen et al., 2025; Shen et al., 2025; Feng et al., 2025), but this path is limited because the best possible performance cannot exceed that of the approximated neuron.

This work takes a novel functional perspective to analyze what constitutes a good design for parallel training in SNNs. We begin by focusing on the reset mechanism of vanilla spiking neurons, identifying its functions as introducing nonlinearity and controlling the membrane potential. Meanwhile, we highlight the drawbacks of the reset mechanism, including its inability to adequately fulfill these functions and its hindrance in parallel training. Finally, we provide a general design guideline for parallel spiking neurons, which can be summarized as: i) aiming to preserve and even enhance the functions of reset, rather than mimicking it directly; and ii) deriving the parallel formulation from the serial one to ensure compatibility between parallel training and serial inference.

Based on these insights, we propose a dynamic decay spiking neuron with a causal convolution structure and an optimized spike firing pattern. It can be shown that our approach can perform the functions of the reset mechanism more flexibly and thoroughly, while also supporting both parallel training and serial inference. We evaluate the advantages of our method in terms of training efficiency, generality across multiple tasks and network architectures, and energy consumption. Its general effectiveness spans from convolutional neural networks to Transformers, and across tasks including image classification, neuromorphic event processing, time-series forecasting, and language modeling. The key contributions of this work are as follows:

- **A Novel Functional View.** Parallel training in spiking neural networks is not merely about replacing the reset mechanism with a seemingly effective technique. Instead, it requires a systematic analysis of how the functions of reset are preserved or enhanced by the modification, which helps us understand the limitations of existing approaches.
- **Design under an Insightful Guideline.** From a functional perspective, we provide a design strategy for parallel spiking neurons. Following this, we propose a dynamic decay spiking neuron, which implements functions better than reset and remains compatible with serial inference.
- **Generality.** Our method demonstrates consistently competitive performance across various network architectures and tasks, while also exhibiting training efficiency and energy benefits.

2 Related Works

Spiking Neuron. The transmission of electrical signals in biological neurons can be modeled using a series of differential equations. Common spiking neuron models include Hodgkin-Huxley neurons (Hodgkin & Huxley, 1952), Leaky Integrate-and-Fire (LIF) neurons (Abbott, 1999), Izhikevich neurons (Izhikevich, 2003), etc. Among these, LIF neurons are the preferred choice for training deep SNNs due to their simplicity (Fang et al., 2021a). Currently, the two main techniques for addressing the non-differentiability issue in deep SNNs are converting an artificial neural network (ANN) into its SNN counterpart (Han et al., 2020; Bu et al., 2022), i.e. ANN-to-SNN, and direct training methods (Wu et al., 2018; Neftci et al., 2019; Yao et al., 2023b) which use surrogate gradients to implement backpropagation through time. In this paper, we focus on the latter approach.

Parallel Training in SNNs. Existing methods use parallelizable modules to directly replace the reset mechanism or approximate the membrane potential of vanilla spiking neurons. For the former, PSN (Fang et al., 2023b) introduces a learnable parameter matrix. In subsequent works, the alternatives focus primarily on the update method of membrane potential (Yarga & Wood, 2023; Li et al., 2024; Su et al., 2024; Xue et al., 2025) and the design of firing functions (Huang et al., 2024b; Chen et al., 2024; Shen et al., 2025; Bal & Sengupta, 2025). However, these methods either abandon the inherent serial inference characteristics of vanilla spiking neurons or fail to fully compensate for the functions of the reset mechanism. For the latter, the approximation methods for membrane potential range from a simple Bernoulli spike emission condition (Chen et al., 2025), a pre-trained surrogate dynamic network (Shen et al., 2025), to fixed-point iteration (Feng et al., 2025), but they do not offer superior performance beyond vanilla spiking neurons. There are also hybrid approaches such as the refractory LIF model (Zhong et al., 2024), in which the membrane potential of the substitute is progressively approximated during the iterative process.

Dynamic Decay. For SNNs, the decay factor, usually termed as the membrane time constant in LIF neurons, implies a limitation on expressiveness due to its fixed nature. PLIF (Fang et al., 2021b) improves neuronal dynamics by making the decay factor learnable. Subsequent methods parameterize the decay factor via adjusting the parameter expression (Fang et al., 2023b; Kosta & Roy, 2023; Shi et al., 2023; Dan et al., 2025; Zhang et al., 2025b), integrating bidirectional parameters (Su et al., 2024), and introducing a complementary bypass (Huang et al., 2024a). In addition, some studies apply decay to the firing threshold (Yin et al., 2021; Bittar & Garner, 2022). However, the decay factor remains static after training. In recent works, gating mechanisms (Yao et al., 2022; Wang et al., 2024), adaptive membrane time constant (Zhang et al., 2025a) or self-connection circuit (Wang & Yu, 2024) have been employed to capture various biological features and enhance adaptiveness. What they have in common is that after training, the decay factor still changes with variations in input, membrane potential, and output spikes. This data-dependent paradigm inspires us to delve deeper into dynamic decay that is solely related to input.

3 A Functional View of Parallelizing Spiking Neurons

Removing the reset mechanism makes spiking neurons trainable in parallel. To understand what this change truly does, we need to answer two basic questions: i) What is the function of reset; ii) How can we compensate for that function, or even improve upon it. The first helps us make sense of prior work, and the second is the key to design parallel spiking neurons.

3.1 Reset Mechanism and Its Function

Hard and Soft Reset. In biological neurons, the depolarized membrane potential is restored to the resting state after the soma fires a spike (Luo, 2020). Spiking neurons abstract the neuronal dynamics described above. Considering the trade-off between bio-plausibility and computational efficiency, the most widely used spiking neuron is the LIF, whose discrete iterative form is as follows (Wu et al., 2018):

$$H_t = \beta V_{t-1} + (1-\beta)X_t, \tag{1}$$

$$S_t = \Theta(H_t - V_{\text{th}}), \tag{2}$$

$$V_t = \begin{cases} H_t(1-S_t) + V_{\text{reset}}S_t, & \text{hard reset} \\ H_t - V_{\text{th}}S_t, & \text{soft reset} \end{cases}. \tag{3}$$

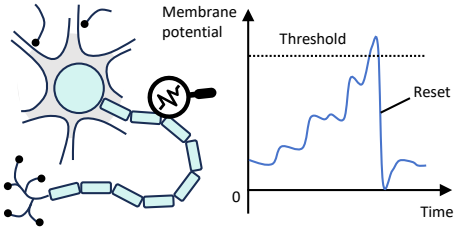

Figure 1: Illustration of a biological neuron (left) and the reset mechanism in neuronal dynamics (right).

In Eq. 1, the current input X_t is integrated with the membrane potential V_{t-1} from last timestep, and the decay factor $\beta = 1 - \frac{1}{\tau_m}$, where τ_m is membrane time constant. In Eq. 2, the Heaviside step function $\Theta(x) = 1$ when $x \geq 0$, i.e. the membrane potential H_t exceeds the threshold V_{th}, indicating that a spike is fired; otherwise, it is set to 0. According to how the membrane potential is regulated based on output spikes, reset can be generally categorized into hard and soft reset as depicted in Eq. 3. In hard reset, the charged membrane potential H_t will be set to a constant V_{reset} if a spike is fired, otherwise it will remain unchanged. V_{reset} is commonly set to 0 for simplicity. In contrast, soft reset subtracts H_t by V_{th} when a spike is fired.

Functions of Reset Mechanism. The first function is to **introduce nonlinearity**. Specifically, the reset mechanism enriches the temporal dynamics of spiking neurons by establishing the following nonlinear relationship between the membrane potential and the input:

Definition 3.1. If the expression $H_t = g(X_1, X_2, ..., X_t)$ is not a linear equation, then the hidden state with respect to the inputs is considered nonlinear.

Remark: Without reset, Eq. 1 can be expanded into a linear form with respect to input.

$$H_t = \sum_{i=1}^{t} \beta^{t-i}(1-\beta)X_i. \tag{4}$$

In contrast, both hard and soft reset insert the firing function $f(.)$ into the iteration of membrane potential at two adjacent timesteps. Taking hard reset as an example, if letting $V_{\text{reset}} = 0$ and

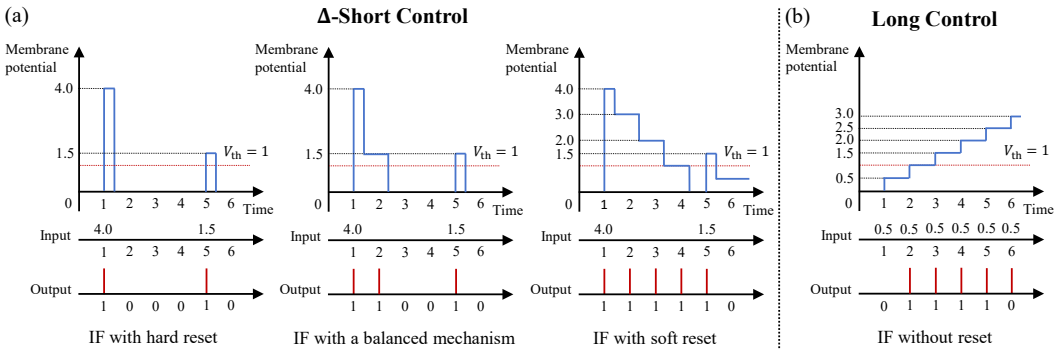

Figure 2: Target scenarios for Δ-short control and long control. (a) Hard reset provides short control at $\Delta = 1$ level, but it does not allow differentiation between inputs of varying importance. In contrast, soft reset extends the control duration as the input magnitude increases, which can lead to continuous spike firing. We expect to find a balanced approach that flexibly determines the duration of the membrane potential's effect based on the input. (b) Without reset, IF neuron carries a risk of membrane potential explosion even with a relatively small constant input sequence $\{0.5\}$.

combining Eq. 1 and Eq. 3, we will derive one-step iteration form of the membrane potential.

$$H_t = \beta(1 - f(H_{t-1}))H_{t-1} + (1 - \beta)X_t. \tag{5}$$

Obviously, it cannot be transformed into an input-dependent linear equation. This is similar for soft reset as well. Therefore, we conclude that reset introduces nonlinearity, and several parallel spiking neuron designs (Fang et al., 2023b; Yarga & Wood, 2023; Bal & Sengupta, 2025) ignore this role.

The second function is to **control membrane potential.** The reset mechanism constrains the membrane potential within a suitable range and averts ceaseless spike firing. For clarity, we quantitatively describe the control ability over the membrane potential as Δ-short control and long control:

Definition 3.2. There exists an $\Delta \in \mathbb{N}^+$ such that, for any $t > \Delta$, if $H_{t-\Delta} \geq V_{\text{th}}$ and $X_{t-\Delta+1}$, ..., $X_t < V_{\text{th}}/\Delta$, it always holds that $H_t < V_{\text{th}}$. In this case, the spiking neuron is said to have **Δ-short control** over the membrane potential.

Remark: The reset mechanism controls how long a large membrane potential affects the spiking neuron. For example, consider an IF neuron without a decay factor, with $V_{\text{th}} = 1$ and $H_1 = X_1 = 4$. In hard reset, the membrane potential is immediately set to 0 after a spike firing, so the effect of the large input lasts for $\Delta = 1$ timestep. In contrast, with a soft reset, the spike persists for 4 timesteps. Δ-short control ensures that a very large input affects the spiking neuron only within a relatively short time window Δ, thereby preventing prolonged spike firing (see Fig. 2).

Definition 3.3. If the input sequence $\{X_t\}$ has an upper bound C, then the membrane potential sequence $\{H_t\}$ also has an upper bound C_H. In this case, the spiking neuron is said to have **long control** over the membrane potential.

Remark: Long control prevents sustained spike firing or even membrane potential explosion caused by small inputs that accumulate without being reset (see Fig. 2). In other words, long control keeps the membrane potential stable over an arbitrarily long period of time.

3.2 Guidance for Design Beyond Reset

Towards Better Functional Realization. Although we have identified two functions of reset, the reset mechanism itself is not the optimal realization of these functions. Specifically, the effect of nonlinearity is binary—either 0 or 1—lacking diversity. The control of the membrane potential by hard or soft reset is rather mechanical and lacks flexibility. In hard reset, no matter how large the membrane potential is, its effect lasts only one timestep, which makes it difficult to distinguish the importance of different inputs. Similarly, in soft reset, the amount subtracted from the membrane potential is fixed. When facing a large input, many timesteps are required to cancel out its effect, which can lead to continuous spike firing. Therefore, in many existing parallel spiking neuron designs, structures that attempt to approximate either hard reset (Shen et al., 2025; Feng et al., 2025)

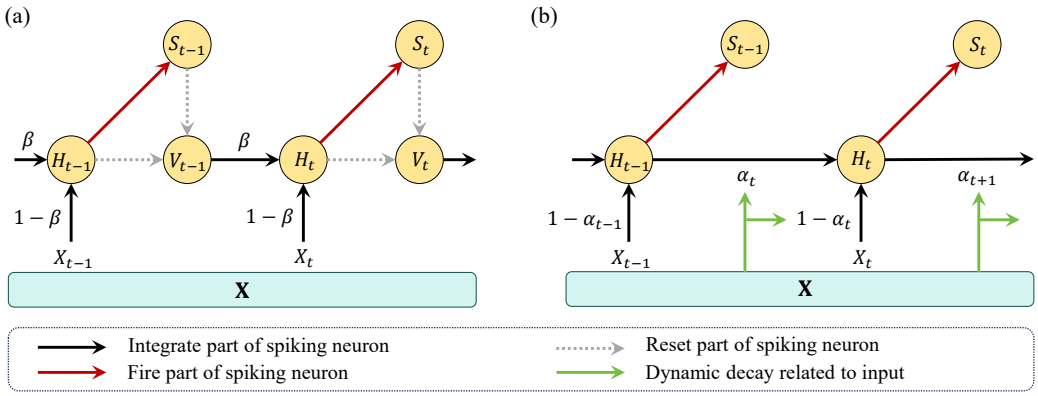

Figure 3: Illustration of the computational process of LIF spiking neuron and reset-free spiking neuron with dynamic decay. (a). In LIF neuron, the current input and the membrane potential from last timestep are integrated with a constant decay factor β. The integrated H_t determines the firing of the spike S_t, which in turn decides whether H_t is reset. (b). After replacing the reset mechanism with dynamic decay α_t, the membrane potential can be computed both serially and in parallel.

or soft reset (Li et al., 2024; Huang et al., 2024b; Chen et al., 2024) can at best reproduce functions similar to those of the reset mechanism, but cannot achieve functions beyond it. Recognizing the inherent limitations of reset helps us focus on enhancing the two functions abstracted from it.

Parallel Training and Serial Inference. A natural idea to realize parallel training in SNNs is to replace the reset mechanism with other parallelizable technique. However, some previous works (Fang et al., 2023b; Li et al., 2024; Su et al., 2024) abandon the inherent efficient serial inference ability of vanilla spiking neurons, namely, *the membrane potential at a given timestep can be computed solely from preceding membrane potential (or a small fixed set of states) and the current input.* As a result, SNNs incur greater computational and memory overhead during inference, and even cannot operate beyond the training length. Therefore, we argue that parallel training in SNNs should remain compatible with serial inference, enabling appropriate computational modes at different stages.

Based on the above analysis, we suggest that a good design strategy for parallelizing spiking neurons is: i) Remove the reset mechanism; ii) Focus on preserving or even enhancing its functions, rather than approximating the reset mechanism itself; iii) Start by improving the serial formulation and then derive its parallel counterpart, rather than assuming a parallel form without a serial basis.

4 Methods: Dynamic Decay Spiking Neuron

The analysis in Sec. 3 provides general guidance for designing parallelizable spiking neurons. Building on this, we propose a Dynamic Decay Spiking Neuron **(DSN)**, which includes two modifications to vanilla spiking neurons: i) the reset mechanism is removed and the constant decay β is replaced with a dynamic decay α_t. Here, α_t is obtained via a causal convolution over the input. ii) the spike firing pattern is optimized by incorporating emerging integer-valued training techniques (Luo et al., 2025). DSN has the following vectorized serial form:

$$\mathbf{H}_t = \boldsymbol{\alpha}_t \odot \mathbf{H}_{t-1} + (1 - \boldsymbol{\alpha}_t) \odot \mathbf{X}_t. \tag{6}$$

$$\mathbf{S}_t = \mathrm{Clip}[\mathrm{Round}(\mathbf{H}_t), 0, N]. \tag{7}$$

Here, the input $\mathbf{X}_t \in \mathbb{R}^{C\times 1}$ has C channels. $\odot$ denotes element-wise product. $\mathrm{Round}(\cdot)$ indicates rounding to the nearest integer. $\mathrm{Clip}[x, 0, N]$ means clipping the input x to the range $[0, N]$. N is a positive integer, representing the upper limit of the number of spikes to be emitted.

In Eq. 19, we derive the dynamic decay $\boldsymbol{\alpha}_t$ from $\mathbf{X}_{t-k+1:t}$ (k inputs from $\mathbf{X}_{t-k+1}$ to $\mathbf{X}_t$) as follows:

$$\boldsymbol{\alpha}'_t = \mathrm{CausalConv1D}(\mathbf{X}_{t-k+1:t}), \tag{8}$$

$$\boldsymbol{\alpha}_t = \mathrm{Sigmoid}(\boldsymbol{\alpha}'_t)^{1/\tau} \tag{9}$$

Here, $\mathrm{CausalConv1D}(\cdot)$ is a causal 1D convolution to mix the features from the past k inputs. $\mathrm{Sigmoid}$ function is chosen to constrain $\boldsymbol{\alpha}_t$ between 0 and 1. τ is a hyperparameter to fine-tune $\boldsymbol{\alpha}_t$.

Design Rationale. After removing the reset mechanism, we find that a varying decay factor can also introduce nonlinearity and control membrane potential, thereby restoring the functions of reset. This forms the basis of our initial design. The causal convolution is usually short but has been shown to be effective in capturing short-term dependency (Gu & Dao, 2024; De et al., 2024). The optimized spike firing pattern helps reduce training overhead and learn better representations (Yao et al., 2025). Moreover, we can choose to introduce an extra learnable parameter $\mathbf{W} \in \mathbb{R}^{C\times C}$ to mix the features across different channels of $\boldsymbol{\alpha}'_t$ before applying the $\mathrm{Sigmoid}$ function in Eq. 9, i.e. $\mathbf{W}\boldsymbol{\alpha}'_t$. This **enhanced DSN** is suitable as a complete block to further improve the modeling ability of SNNs.

Functions Superior to Reset. DSN is a specific implementation of dynamic decay α_t, which can be related to input at preceding timesteps and is usually limited to between 0 and 1 using a non-linear activation function with learnable parameters W:

$$\alpha_t = \phi(X_t, X_{t-1}, ...|W) \in [0, 1] \tag{10}$$

In fact, we can prove that dynamic decay in Eq. 10 can implement all the functions of reset.

Proposition 4.1. *Dynamic decay can introduce nonlinearity and enabling more flexible Δ-short and long control of the membrane potential than the reset mechanism.*

Remark: We provide the detailed proof in Appendix A.1. An intuitive interpretation is that the variability of α_t broadens the expressive range of nonlinearity and allows adaptive control of the membrane potential. Additionally, we note that the proposition holds for both binary and integer-valued spike firing, showing that dynamic decay is a general and powerful alternative to reset.

From Serial Inference to Parallel Training. The iterative Eq. 19 can be rewritten into a general form determined solely by $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_t$:

$$\mathbf{H}_t = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \boldsymbol{\alpha}_j \right) (\mathbf{1} - \boldsymbol{\alpha}_i) \odot \mathbf{X}_i. \tag{11}$$

We stack $\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_T$ to obtain $\mathbf{H} \in \mathbb{R}^{C\times T}$, do the same for $\mathbf{X} \in \mathbb{R}^{C\times T}$, and can finally get $\mathbf{H} = \mathbf{X}\mathbf{W}$ parallel form (See Appendix A.3 for the detailed derivation):

$$\begin{aligned} \mathbf{H} &= \mathbf{X}\left(\left(\left(\frac{\mathbf{1}-\mathbf{A}}{\mathbf{P}}\right)^{T}\mathbf{P}\right)\odot\mathbf{M}\right). \\ \mathbf{P}_t &= \prod_{i=1}^{t}\boldsymbol{\alpha}_i,\ \mathbf{A}_t = \boldsymbol{\alpha}_t,\ \mathbf{M}_{ij} = \begin{cases}1, & j \geq i\\ 0, & j < i\end{cases}. \end{aligned} \tag{12}$$

Here, $\mathbf{P}$ and $\mathbf{A} \in \mathbb{R}^{C\times T}$. $\mathbf{M} \in \mathbb{R}^{T\times T}$ is a causal mask for the interaction of neuronal inputs. $\frac{\mathbf{1}-\mathbf{A}}{\mathbf{P}}$ and $\odot$ denote element-wise division and product, respectively. During training, the dynamic decay $\mathbf{A}$, membrane potential $\mathbf{H}$ and their gradients can be computed rapidly in parallel[1] with Triton-based acceleration operators (Tillet et al., 2019; Yang & Zhang, 2024). During inference, we switch to Eq. 19 for efficient serial inference, which requires to store only minimal states from the causal convolution and recurrent structure, thereby reducing both computational and memory overhead.

5 Experiments

In this section, we evaluate the proposed DSN in terms of training efficiency, generality, and energy consumption. Specifically, the neuronal generality includes: i) The effectiveness of spiking neuron design, including the causal convolution structure and integer-valued training technique; ii) Flexible adaptation to various network architectures, such as convolutional neural networks and Transformers; iii) Competitive performance across multiple tasks.

[1]In practice, we avoid computing $\mathbf{H}$ via matrix multiplication due to numerical instability of $\mathbf{P}$ as the denominator (Yang et al., 2024). Instead, we use a two-stage parallel scan algorithm (Martin & Cundy, 2018) for Eq. 19 to derive $\mathbf{H}$.

Table 1: Comparison of training time (ms) for different spiking neurons. The timestep 32^* is for Sequential CIFAR10; others are for CIFAR10. Param: parameters in C spiking neurons of a layer.

Methods	Parallel	Serial	Param.	Forward				Backward			
				32^*	128	256	512	32^*	128	256	512
LIF (Abbott, 1999)	✗	✓	0	4.90	19.53	41.31	77.42	5.84	25.34	52.77	101.88
PSN (Fang et al., 2023b)	✓	✗	T^2	0.20	0.24	0.20	0.21	0.23	0.22	0.24	0.25
DSN (Ours)	✓	✓	NC	0.68	0.75	0.68	0.69	0.71	0.64	0.69	0.64

5.1 Training Efficiency

In this section, we build a convolution-based SNN on CIFAR10 (Krizhevsky et al., 2009) and Sequential CIFAR10. For CIFAR10, we adjust the training timesteps by repeating the input images. For Sequential CIFAR10 where the images from CIFAR10 are input into the model in sequential pixel form, the timestep is equal to the width 32 of the images. We measure the average time for a forward and backward pass through the first activation layer with 4096 neurons over 100 trials. The results in Table 1 lead to the following conclusions:

- DSN benefits from the acceleration provided by parallel training. In contrast to LIF neuron, whose wall-clock time grows linearly with the number of timesteps, DSN maintains nearly constant runtime. On Sequential CIFAR10, DSN achieves $7.2\times$ and $8.2\times$ speedups in forward and backward pass, respectively, compared to LIF neuron. This gap further widens with larger timesteps.
- Given comparable training time, DSN is more parameter-efficient than PSN for longer sequences. For instance, in a layer with $C = 1024$ spiking neurons and $T = 512$ timesteps, DSN requires only $NC = 4 \times 1024$ parameters, whereas PSN requires $T^2 = 512^2$, resulting in substantial memory and computational overhead. Additionally, the acceleration operators of DSN have room for further optimization, as discussed in Appendix B.1.

5.2 Generality

5.2.1 Effectiveness of Spiking Neuron Design

We evaluate the effectiveness of spiking neuron design on Sequential CIFAR10, including the causal convolution structure and integer-valued training technique. Results in Table 2 show that:

First, dynamic decay is the primary contributor to performance. Removing the causal convolution eliminates meaningful neuronal dynamics and leads to a significant degradation. We also test alternative decay structures, including fully connected layers for inter-channel interaction (89.28%), low-rank mappings (86.72%), and inter-channel convolution (86.76%). In comparison, causal convolution remains a simple yet effective design.

Second, integer-valued training technique effectively complements dynamic decay. However, not all spiking neurons see similar improvements; for example, it even causes a performance drop in PSN. In contrast, models with decay structures such as LIF, PLIF, and DSN experience performance gains, with DSN benefiting the most.

Table 2: Ablation studies on Sequential CIFAR10. ivt: integer-valued training.

Methods	Accuracy (%)
DSN (Ours)	89.78
DSN w/o conv	84.53 (-5.25)
DSN w/o ivt	87.45 (-2.33)
LIF (Abbott, 1999)	81.50
LIF w/ ivt	82.16 ($+0.66$)
PLIF (Fang et al., 2021b)	83.49
PLIF w/ ivt	84.25 ($+0.76$)
PSN (Fang et al., 2023b)	88.45
PSN w/ ivt	86.84 (-1.61)

Based on the above experiments, we design DSN by combining causal convolution with integer-valued training technique, and validate its generality across different model architectures and tasks.

5.2.2 Generality Across Multiple Tasks

We evaluate DSN on four types of tasks: image classification, neuromorphic event processing, time-series forecasting, and language modeling. The first two tasks use convolutional neural network,

Table 3: Performance on Sequential CIFAR10 and CIFAR100. The timestep is 32. † means enhanced DSN mentioned in Sec. 4. Param: parameters (M). Acc: accuracy (%).

Methods	Parallel	Serial	S-CIFAR10		S-CIFAR100	
			Param.	Acc.	Param.	Acc.
LIF (Abbott, 1999)	✗	✓	0.513	81.50	0.537	55.45
PLIF (Fang et al., 2021b)	✗	✓	0.513	81.50	0.537	55.45
PSN (Fang et al., 2023b)	✓	✗	0.521	88.45	0.544	62.21
IPSU (Li et al., 2024)	✓	✗	0.517	87.28	0.540	59.76
PMSN (Chen et al., 2024)	✓	✓	0.540	90.97	0.560	66.08
DSN (Ours)	✓	✓	0.519	89.78	0.542	64.70
DSN† (Ours)	✓	✓	0.683	**92.96**	0.707	**68.48**

Table 4: Performance on ImageNet and CIFAR10-DVS. T: Timesteps.

Dataset	Methods	Architecture	Parallel	Serial	T	Accuracy (%)
ImageNet	MBPN (Guo et al., 2023)	ResNet18	✗	✓	4	63.14
	SEW ResNet (Fang et al., 2021a)	SEW ResNet18	✗	✓	4	63.18
	DeepTAGE (Liu et al., 2025)	ResNet18	✗	✓	4	**68.52**
	PMSN (Chen et al., 2024)	SEW ResNet18	✓	✓	4	66.64
	PSN (Fang et al., 2023b)	SEW ResNet18	✓	✗	4	67.63
	DSN (Ours)	SEW ResNet18	✓	✓	4	**68.21**
CIFAR10-DVS	SEW ResNet (Fang et al., 2021a)	Wide 7B Net	✗	✓	16	74.40
	GLIF (Yao et al., 2022)	Wide 7B Net	✗	✓	16	78.10
	DeepTAGE (Liu et al., 2025)	VGG-11	✗	✓	10	81.23
	RPSU (Li et al., 2024)	VGGSNN	✓	✗	10	82.00
	FPT (Feng et al., 2025)	VGG-11	✓	✗	10	**85.50**
	sliding PSN (Fang et al., 2023b)	VGGSNN	✓	✓	4, 8	82.30, **85.30**
	DSN (Ours)	VGGSNN	✓	✓	4, 8	**83.90**, **85.30**

while the latter two adopt Transformer or recurrent architectures. Competitive results across multiple datasets and network architectures demonstrate the general effectiveness of DSN.

Sequential CIFAR. The experimental setup and other hyperparameters are kept consistent with those of PSN (Fang et al., 2023b). Results in Table 3 show that our DSN exceeds PSN and is comparable to other baselines such as IPSU (Li et al., 2024). Furthermore, our enhanced DSN achieves state-of-the-art performance. Despite more parameters, it incorporates more intricate neuron interactions, which is essential for further improving model performance.

ImageNet. We further evaluate the performance of DSN on this larger-scale image classification task (Deng et al., 2009). The experimental settings are identical to Fang et al. (2023b). As illustrated in Table 4, our method still achieves relatively higher accuracy among parallel spiking neurons.

CIFAR10-DVS. To validate the effectiveness of our method in processing neuromorphic event, we select CIFAR10-DVS (Li et al., 2017) as the evaluation benchmark. We adopt the VGG architecture in Deng et al. (2022). As shown in Table 4, DSN shows performance comparable to sliding PSN.

Time-series Forecasting Tasks. On more realistic time-series forecasting tasks, we adapt DSN to the following datasets: Metr-la (Li et al., 2018): traffic flow records from Los Angeles; Pems-bay (Li et al., 2018): traffic flow records from the San Francisco Bay Area; Solar (Lai et al., 2018): solar power generation data. Baseline architectures include Transformer (Vaswani et al., 2017), iTransformer (Liu et al., 2024), and their respective SNN counterparts (Zhou et al., 2023; Lv et al., 2024). For all SNN-based time-series forecasting models, we replace the original LIF neurons with DSN and make architectural modifications (see Appendix B.3). The Root Relative Squared Error (RSE) and the coefficient of determination (R^2) is used as metrics. It can be seen from Table 5 that DSN-based architectures exhibit competitive performance on various tasks and prediction lengths.

Table 5: Experimental results of 3 time-series forecasting tasks with prediction lengths $L = 6, 24, 48, 96$. ↑ (↓) indicates the higher (lower) the better. All results are averaged across 3 random seeds. The leading zero before the decimal point is omitted. Param: parameters (M).

Methods	Spike	Param.	Metric	Metr-la				Pems-bay				Solar				Avg.
				6	24	48	96	6	24	48	96	6	24	48	96	
Transformer	✗	2.53	R^2↑	.727	**.554**	.413	**.284**	.785	.734	.688	.673	.953	.858	.759	.718	.679
			RSE↓	.551	**.704**	.808	**.895**	.502	.558	.610	.618	.223	.377	.504	**.545**	.575
Spikformer	✓	2.52	R^2↑	.713	.527	.399	.267	.773	.697	.686	.667	.929	.828	.744	.674	.659
			RSE↓	.565	.725	.818	.903	.514	.594	.606	.621	.272	.426	.519	.586	.596
Spikformer w/ PSN	✓	2.68	R^2↑	.716	.518	.401	.268	.738	.671	.666	.639	.861	.759	.554	.439	.603
			RSE↓	.562	.731	.815	.901	.553	.620	.624	.649	.383	.504	.685	.749	.648
Spikformer w/ DSN	✓	2.68	R^2↑	**.734**	.549	**.422**	.283	**.807**	**.745**	**.696**	**.683**	**.956**	**.860**	**.765**	**.736**	**.686**
			RSE↓	**.539**	.720	**.804**	.896	**.475**	**.538**	**.581**	**.594**	**.219**	**.373**	**.481**	.572	**.566**
iTransformer	✗	1.63	R^2↑	**.829**	.623	.439	**.285**	**.887**	.719	.685	.668	**.964**	**.879**	**.799**	**.738**	.710
			RSE↓	**.436**	.648	.780	**.878**	**.362**	.547	**.561**	.584	**.191**	.348	**.448**	.563	.529
iSpikformer	✓	1.63	R^2↑	.817	.618	**.440**	.279	.879	**.744**	.687	**.674**	.961	.876	.795	**.738**	.709
			RSE↓	.475	.668	**.752**	.905	.376	**.536**	.569	**.580**	.204	**.333**	.465	**.521**	.532
iSpikformer w/ DSN	✓	1.79	R^2↑	.823	**.624**	**.440**	.283	.883	.740	**.689**	.672	**.964**	**.879**	.798	.736	**.711**
			RSE↓	.450	**.646**	.755	.881	.368	.541	.564	.583	.199	.350	.450	.526	**.526**

WikiText-103. To demonstrate that DSN can model more complex sequences such as language, we evaluate its perplexity on WikiText-103 (Merity et al., 2017), a large-scale word-level dataset constructed from the English Wikipedia. Results in Table 6 show that DSN performs the best among spiking language models with a moderate number of parameters. Since our exploration of the model architecture is preliminary, further improvements of DSN-based language models can be expected in the future.

Table 6: Experimental results on WikiText-103 dataset. ↓ indicates the lower the better. Param: parameters (M). Ppl: perplexity.

Methods	Param.	Ppl. ↓
SpikeGPT (Zhu et al., 2024b)	213	39.75
SPikE-SSM (Zhong et al., 2024)	75	33.18
SpikingSSM (Shen et al., 2025)	75	33.94
DSN (Ours)	137	**29.60**

5.3 Energy Consumption

We follow the method in Yao et al. (2023a) to evaluate the energy consumption of the Sequential CIFAR network with different spiking neurons. Results in Table 7 show that although additional modules such as convolution operations were introduced, the total energy consumption of DSN is slightly lower than that of LIF due to its reduced spike firing rate. Additionally, PSN has the highest energy cost due to matrix multiplication and high spike firing rate. See Appendix B.5 for more details.

Table 7: Energy cost (mJ) of different methods.

Methods	S-CIFAR10	S-CIFAR100
LIF (Abbott, 1999)	107.80	121.36
PSN (Fang et al., 2023b)	235.87	241.64
DSN (Ours)	**104.24**	**110.29**

6 Conclusion

In this paper, we identify a critical limitation in existing efforts toward parallel training in SNNs: the neglect of preserving essential characteristics of vanilla spiking neurons, including the functions of the reset mechanism and the capability for serial inference. Under a new functional viewpoint, we summarize the functions of the reset mechanism in vanilla spiking neurons as introducing nonlinearity and controlling membrane potential. Based on this, we propose a guideline for designing parallel spiking neurons and introduce a dynamic decay spiking neuron that offers improved functions compared to reset while remaining compatible with serial inference. We verify the competitive training efficiency, generality across multiple tasks, and energy consumption of our method. Our work offers new insights into the exploration of high-performance spiking neurons with efficient training and inference abilities in the era of foundation models.

REFERENCES

Larry F Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–304, 1999.

Malyaban Bal and Abhronil Sengupta. P-spikessm: Harnessing probabilistic spiking state space models for long-range dependency tasks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=Sf4ep9Udjf.

Alexandre Bittar and Philip N Garner. A surrogate gradient spiking baseline for speech command recognition. *Frontiers in Neuroscience*, 16:865897, 2022.

Tong Bu, Wei Fang, Jianhao Ding, Penglin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Hanqi Chen, Lixing Yu, Shaojie Zhan, Penghui Yao, and Jiankun Shao. Time-independent spiking neuron via membrane potential estimation for efficient spiking neural networks. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025. doi: 10.1109/ICASSP49660.2025.10890472.

Xinyi Chen, Jibin Wu, Chenxiang Ma, Yinsong Yan, Yujie Wu, and Kay Chen Tan. Pmsn: A parallel multi-compartment spiking neuron for multi-scale temporal processing. *arXiv preprint arXiv:2408.14917*, 2024.

Yongping Dan, Changhao Sun, Hengyi Li, and Lin Meng. Adaptive spiking neuron with population coding for a residual spiking neural network. *Applied Intelligence*, 55(4):288, 2025.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

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.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009. doi: 10.1109/CVPR.2009.5206848. URL https://doi.org/10.1109/CVPR.2009.5206848.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=_XNtisL32jv.

Jason K. Eshraghian, Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *Proc. IEEE*, 111(9):1016–1054, 2023.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 21056–21069, 2021a.

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*, pp. 2661–2671, 2021b.

Wei Fang, Yanqi Chen, Jianhao Ding, Zhaofei Yu, Timothée Masquelier, Ding Chen, Liwei Huang, Huihui Zhou, Guoqi Li, and Yonghong Tian. Spikingjelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 9(40):eadi1480, 2023a.

Wei Fang, Zhaofei Yu, Zhaokun Zhou, Ding Chen, Yanqi Chen, Zhengyu Ma, Timothée Masquelier, and Yonghong Tian. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023b.

Wanjin Feng, Xingyu Gao, Wenqian Du, Hailong Shi, Peilin Zhao, Pengcheng Wu, and Chunyan Miao. Efficient parallel training methods for spiking neural networks with constant time complexity. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, Canada, July 13-19, 2025*, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Yufei Guo, Yuanpei Chen, Liwen Zhang, Xiaode Liu, Yinglei Wang, Xuhui Huang, and Zhe Ma. Im-loss: Information maximization loss for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pp. 156–166, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/010c5ba0cafc743fece8be02e7adb8dd-Paper-Conference.pdf.

Yufei Guo, Yuhan Zhang, Yuanpei Chen, Weihang Peng, Xiaode Liu, Liwen Zhang, Xuhui Huang, and Zhe Ma. Membrane potential batch normalization for spiking neural networks. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 19363–19373. IEEE, 2023. doi: 10.1109/ICCV51070.2023.01779. URL https://doi.org/10.1109/ICCV51070.2023.01779.

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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 13555–13564. Computer Vision Foundation / IEEE, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, volume 9908 of *Lecture Notes in Computer Science*, pp. 630–645. Springer, 2016.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

Jiakui Hu, Man Yao, Xuerui Qiu, Yuhong Chou, Yuxuan Cai, Ning Qiao, Yonghong Tian, Bo Xu, and Guoqi Li. High-performance temporal reversible spiking neural networks with O(L) training memory and O(1) inference cost. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=s4h6nyjM9H.

Yifan Hu, Lei Deng, Yujie Wu, Man Yao, and Guoqi Li. Advancing spiking neural networks toward deep residual learning. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2): 2353–2367, 2025. doi: 10.1109/TNNLS.2024.3355393.

Yulong Huang, Xiaopeng Lin, Hongwei Ren, Haotian Fu, Yue Zhou, Zunchang Liu, Biao Pan, and Bojun Cheng. CLIF: Complementary leaky integrate-and-fire neuron for spiking neural networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 19949–19972. PMLR, 21–27 Jul 2024a.

Yulong Huang, Zunchang Liu, Changchun Feng, Xiaopeng Lin, Hongwei Ren, Haotian Fu, Yue Zhou, Hong Xing, and Bojun Cheng. Prf: Parallel resonate and fire neuron for long sequence learning in spiking neural networks. *arXiv preprint arXiv:2410.03530*, 2024b.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Adarsh Kumar Kosta and Kaushik Roy. Adaptive-spikenet: event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6021–6027. IEEE, 2023.

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

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: An event-stream dataset for object classification. *Frontiers in Neuroscience*, Volume 11 - 2017, 2017. ISSN 1662-453X. doi: 10.3389/fnins.2017.00309. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2017.00309.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Yang Li, Yinqian Sun, Xiang He, Yiting Dong, Dongcheng Zhao, and Yi Zeng. Parallel spiking unit for efficient training of spiking neural networks. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2024.

Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 23426–23439, 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/c4ca4238a0b923820dcc509a6f75849b-Paper.pdf.

Yihan Lin, Yifan Hu, Shijie Ma, Dongjie Yu, and Guoqi Li. Rethinking pretraining as a bridge from anns to snns. *IEEE Trans. Neural Networks Learn. Syst.*, 35(7):9054–9067, 2024. doi: 10.1109/TNNLS.2022.3217796. URL https://doi.org/10.1109/TNNLS.2022.3217796.

Wei Liu, Li Yang, Mingxuan Zhao, Shuxun Wang, Jin Gao, Wenjuan Li, Bing Li, and Weiming Hu. Deeptage: Deep temporal-aligned gradient enhancement for optimizing spiking neural networks. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=drPDukdY3t.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Liqun Luo. *Principles of neurobiology*. Garland Science, 2020.

Xinhao Luo, Man Yao, Yuhong Chou, Bo Xu, and Guoqi Li. Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In *European Conference on Computer Vision*, pp. 253–272. Springer, 2025.

Changze Lv, Yansen Wang, Dongqi Han, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. Efficient and effective time-series forecasting with spiking neural networks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

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

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory- and time-efficient backpropagation for training spiking neural networks. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pp. 6143–6153. IEEE, 2023. doi: 10.1109/ICCV51070.2023.00567. URL https://doi.org/10.1109/ICCV51070.2023.00567.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

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.

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.

ROYUD Nishino and Shohei Hido Crissman Loomis. Cupy: A numpy-compatible library for nvidia gpu calculations. *31st confernce on neural information processing systems*, 151(7), 2017.

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.

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

Guobin Shen, Jindong Li, Tenglong Li, Dongcheng Zhao, and Yi Zeng. $SpikePack$: Enhanced Information Flow in Spiking Neural Networks with High Hardware Compatibility. *arXiv e-prints*, art. arXiv:2501.14484, January 2025. doi: 10.48550/arXiv.2501.14484.

Shuaijie Shen, Chao Wang, Renzhuo Huang, Yan Zhong, Qinghai Guo, Zhichao Lu, Jianguo Zhang, and Luziwei Leng. Spikingssms: Learning long sequences with sparse and parallel spiking state space models. In Toby Walsh, Julie Shah, and Zico Kolter (eds.), *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pp. 20380–20388. AAAI Press, 2025. doi: 10.1609/AAAI.V39I19.34245. URL https://doi.org/10.1609/aaai.v39i19.34245.

Cong Shi, Li Wang, Haoran Gao, and Min Tian. Learnable leakage and onset-spiking self-attention in snns with local error signals. *Sensors*, 23(24):9781, 2023.

Qiaoyi Su, Shijie Mei, Xingrun Xing, Man Yao, Jiajun Zhang, Bo Xu, and Guoqi Li. Snn-bert: Training-efficient spiking neural networks for energy-efficient bert. *Neural Networks*, 180: 106630, 2024.

Kaiwen Tang, Zhanglu Yan, and Weng-Fai Wong. Sorbet: A neuromorphic hardware-compatible transformer-based spiking language model. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, Canada, July 13-19, 2025*, 2025.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pp. 10–19, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Haoran Wang, Herui Zhang, Siyang Li, and Dongrui Wu. Gated parametric neuron for spike-based audio recognition. *Neurocomputing*, 609:128477, 2024.

Lei Wang, Yu Cheng, Yining Shi, Zhengju Tang, Zhiwen Mo, Wenhao Xie, Lingxiao Ma, Yuqing Xia, Jilong Xue, Fan Yang, and Zhi Yang. TileLang: A Composable Tiled Programming Model for AI Systems. *arXiv e-prints*, art. arXiv:2504.17577, April 2025. doi: 10.48550/arXiv.2504.17577.

Lihao Wang and Zhaofei Yu. Autaptic synaptic circuit enhances spatio-temporal predictive learning of spiking neural networks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

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.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/82846e19e6d42ebfd4ace4361def29ae-Abstract-Conference.html.

Changqing Xu, Guoqing Sun, Yi Liu, Xinfang Liao, and Yintang Yang. Pararevsnn: A parallel reversible spiking neural network for efficient training and inference. *CoRR*, abs/2508.01223, 2025. doi: 10.48550/ARXIV.2508.01223. URL https://doi.org/10.48550/arXiv.2508.01223.

Peng Xue, Wei Fang, Zhengyu Ma, Zihan Huang, Zhaokun Zhou, Yonghong Tian, Timothée Masquelier, and Huihui Zhou. Channel-wise Parallelizable Spiking Neuron with Multiplication-free Dynamics and Large Temporal Receptive Fields. *arXiv e-prints*, art. arXiv:2501.14490, January 2025. doi: 10.48550/arXiv.2501.14490.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023a. URL http://papers.nips.cc/paper_files/paper/2023/hash/ca0f5358dbadda74b3049711887e9ead-Abstract-Conference.html.

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*, 45(8):9393–9410, 2023b.

Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications*, 15(1):4464, 2024.

Man Yao, Xuerui Qiu, Tianxiang Hu, Jiakui Hu, Yuhong Chou, Keyu Tian, Jianxing Liao, Luziwei Leng, Bo Xu, and Guoqi Li. Scaling spike-driven transformer with efficient spike firing approximation training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–18, 2025.

Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Sidi Yaya Arnaud Yarga and Sean UN Wood. Accelerating snn training with stochastic parallelizable spiking neurons. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2023.

Bojian Yin, Federico Corradi, and Sander M Bohté. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Machine Intelligence*, 3(10): 905–913, 2021.

Jiqing Zhang, Malu Zhang, Yuanchen Wang, Qianhui Liu, Baocai Yin, Haizhou Li, and Xin Yang. Spiking neural networks with adaptive membrane time constant for event-based tracking. *IEEE Transactions on Image Processing*, 34:1009–1021, 2025a. doi: 10.1109/TIP.2025.3533213.

Tianqing Zhang, Kairong Yu, Jian Zhang, and Hongwei Wang. Da-lif: Dual adaptive leaky integrate-and-fire model for deep spiking neural networks. In *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5, 2025b. doi: 10.1109/ICASSP49660.2025.10888909.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11062–11070, 2021.

Yan Zhong, Ruoyu Zhao, Chao Wang, Qinghai Guo, Jianguo Zhang, Zhichao Lu, and Luziwei Leng. Spike-ssm: A sparse, precise, and efficient spiking state space model for long sequences learning. *arXiv preprint arXiv:2410.17268*, 2024.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Rui-Jie Zhu, Yu Zhang, Ethan Sifferman, Tyler Sheaves, Yiqiao Wang, Dustin Richmond, Peng Zhou, and Jason K Eshraghian. Scalable matmul-free language modeling. *arXiv preprint arXiv:2406.02528*, 2024a.

Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason Eshraghian. Spikegpt: Generative pre-trained language model with spiking neural networks. *Trans. Mach. Learn. Res.*, 2024, 2024b.

A Details of Theoretical Analysis

A.1 Proof of Functions of Dynamic Decay

Proposition A.1. *Dynamic decay can introduce nonlinearity and enabling more flexible Δ-short and long control of the membrane potential than the reset mechanism.*

Proof. Firstly, with dynamic decay, the iteration form of the membrane potential is:

$$H_t = \alpha_t H_{t-1} + (1-\alpha_t)X_t. \tag{13}$$

which can be rewritten as

$$H_t = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \alpha_j \right) (1-\alpha_i) X_i. \tag{14}$$

Note that the coefficient of X_i is input-dependent, which implies that the combination of X_i is not actually a linear term. From Definition 3.1, we conclude that dynamic decay introduces nonlinearity.

Next, we show how α_t controls the membrane potential H_t. Given $\Delta \in \mathbb{N}^+$, suppose $H_{t-\Delta} \geq V_{\text{th}}$ and $X_{t-\Delta+1}, ..., X_t < V_{\text{th}}/\Delta$. Note that when

$$\alpha_{t-\Delta+1} < \frac{V_{\text{th}} - X_{t-\Delta+1}}{H_{t-\Delta} - X_{t-\Delta+1}} \in (0, 1] \tag{15}$$

We have

$$\begin{aligned} H_{t-\Delta+1} &= \alpha_{t-\Delta+1} H_{t-\Delta} + (1-\alpha_{t-\Delta+1}) X_{t-\Delta+1} \\ &= \alpha_{t-\Delta+1}(H_{t-\Delta} - X_{t-\Delta+1}) + X_{t-\Delta+1} \\ &< V_{\text{th}} - X_{t-\Delta+1} + X_{t-\Delta+1} = V_{\text{th}}. \end{aligned} \tag{16}$$

For $m = 2, 3, ..., \Delta$, we sequentially derive

$$\begin{aligned} H_{t-\Delta+m} &= \alpha_{t-\Delta+m} H_{t-\Delta+m-1} + (1-\alpha_{t-\Delta+m}) X_{t-\Delta+m} \\ &< \alpha_{t-\Delta+m} V_{\text{th}} + (1-\alpha_{t-\Delta+m}) V_{\text{th}} = V_{\text{th}} \end{aligned} \tag{17}$$

When $m = \Delta$, $H_t < V_{\text{th}}$. From Definition 3.2, every α_t satisfying Eq. 15 can guarantee Δ-short control ability and avert continuous firing when inputs are smaller than threshold.

For long control ability, suppose that $\{X_t\}$ has an upper bound C, i.e., $X_i \leq C, i = 1, 2, ..., t$. It is easy to get that $H_1 = (1-\alpha_1)X_1 \leq C$. Besides, if $H_{t-1} \leq C$, then

$$H_t = \alpha_t H_{t-1} + (1-\alpha_t)X_t \leq \alpha_t C + (1-\alpha_t)C = C \tag{18}$$

By mathematical induction, we know that $\{H_t\}$ has an upper bound C. From Definition 3.3, α_t has long control ability. □

A.2 Dynamic Decay in Integer-valued Training Case

When the integer-valued training technique is introduced, dynamic decay is still able to retain the two functions of the reset mechanism. According to Proposition 1 of Yao et al. (2025), integer-value output (with upper bound N) is equal to the sum of spikes generated by IF-SR (IF with Soft Reset) spiking neuron with N timesteps. Therefore, functions of the reset mechanism are still preserved at the single-neuron level. Consequently, Proposition 4.1 should still hold in the integer spike scenario.

In fact, assuming the integer spiking function $f(H_t) = \lfloor \text{Clip}(H_t, 0, N) \rfloor$ and $V_{\text{th}} = 1$, where $\text{Clip}(x, 0, N)$ means clipping the input x to the range [0, N], and $\lfloor . \rfloor$ is the floor function. Since the functions of non-linearity and membrane potential control in dynamic decay are independent of the choice of the spike firing function, this concludes our functional analysis in Proposition 4.1 in integer-valued training case.

A.3 Detailed derivation of Dynamic Decay from serial to parallel Form

After removing the reset mechanism, the membrane potential iteration is no longer influenced by spike firing, allowing us to focus solely on the following serial formulation:

$$\mathbf{H}_t = \boldsymbol{\alpha}_t \odot \mathbf{H}_{t-1} + (1 - \boldsymbol{\alpha}_t) \odot \mathbf{X}_t. \tag{19}$$

Here, $\mathbf{H}_t, \boldsymbol{\alpha}_t, \mathbf{X}_t \in \mathbb{R}^{C\times 1}$ has C channels. $\odot$ denotes element-wise product. The above equation can be rewritten into a general form determined solely by $\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_t$:

$$\mathbf{H}_t = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \boldsymbol{\alpha}_j \right) (\mathbf{1} - \boldsymbol{\alpha}_i) \odot \mathbf{X}_i. \tag{20}$$

We stack $\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_T$ to obtain $\mathbf{H} \in \mathbb{R}^{C\times T}$, do the same for $\mathbf{X} \in \mathbb{R}^{C\times T}$, and can finally get $\mathbf{H} = \mathbf{X}\mathbf{W}$ parallel form, where $\mathbf{W}_{ij} = (\prod_{k=i+1}^{j} \boldsymbol{\alpha}_k)(\mathbf{1} - \boldsymbol{\alpha}_i)$ for $j \geq i$, otherwise $\mathbf{W}_{ij} = 0$.

We first temporarily ignore the term $(\mathbf{1} - \boldsymbol{\alpha}_i)$. Then, an all-ones upper triangular matrix $\mathbf{M}$ can capture the distinction between cases for i and j. We observe that $\prod_{k=i+1}^{j} \boldsymbol{\alpha}_k$ can be rewritten into a form with indices starting from 1 by matrix multiplication and reciprocals:

$$\mathbf{W} = \left(\left(\frac{\mathbf{1}}{\mathbf{P}} \right)^{T} \mathbf{P} \right) \odot \mathbf{M}, \text{where } \mathbf{P} \in \mathbb{R}^{C\times T}, \mathbf{P}_j = \prod_{k=1}^{j} \boldsymbol{\alpha}_k \tag{21}$$

Finally, by taking $(\mathbf{1} - \boldsymbol{\alpha}_i)$ into account, we obtain the parallel form

$$\mathbf{H} = \mathbf{X} \left(\left(\left(\frac{\mathbf{1} - \mathbf{A}}{\mathbf{P}} \right)^{T} \mathbf{P} \right) \odot \mathbf{M} \right), \text{where } \mathbf{A} \in \mathbb{R}^{C\times T}, \mathbf{A}_i = \boldsymbol{\alpha}_i \tag{22}$$

B Experimental Details

B.1 Acceleration

Measurement of GPU Resource Utilization. GPU computational cores have theoretical limits on memory throughput (bytes/s) and compute throughput (FLOPs/s). An algorithm's GPU resource utilization is determined by how closely its achieved throughput approaches these theoretical peaks. For example, FlashAttention-2 (Dao, 2024) increases the proportion of matrix multiplication, parallelizes attention operations, and achieves over 50% of the theoretical peak throughput, significantly accelerating model training. The process of updating the membrane potential in a spiking neuron module involves sequential execution across multiple compute kernels. We employ NVIDIA Nsight Compute tool to analyze the kernel execution of spiking neurons. Denoting the execution time of the i-th compute core as t_i and its ratio of throughput to the theoretical peak as Throughput_i, the GPU resource utilization rate of the module is defined as the weighted average throughput of total N compute cores:

$$\text{Throughput}_{avg} = \frac{\sum_{i=1}^{N} t_i * \text{Throughput}_i}{\sum_{i=1}^{N} t_i} \tag{23}$$

We evaluate this metric on the spiking neurons in the first activation layer of the neural network used for Sequential CIFAR10, and the experimental setup can be found in B.2. All the experiments are carried out on an NVIDIA A100 GPU.

Discussion about the Optimization Level. Traditional PyTorch-based training approaches for SNNs often suffer from substantial redundant computation and memory access due to serial processing, leading to a sharp increase in training time as the number of timestep grows. To improve training speed, certain critical operators can be implemented using CUDA. By modifying the original computation and memory access patterns, finely controlling GPU threads and memory utilization, and developing specialized computational instructions in hardware, the algorithm's runtime can be significantly reduced. For example, Spikingjelly (Fang et al., 2023a) employs CuPy (Nishino & Loomis, 2017) to implement operators of spiking neurons such as LIF model, combining multiple operations into a single CUDA kernel. This reduces kernel invocation overhead and improves

computational efficiency, achieving CUDA-level acceleration, and markedly reducing the difficulty of training SNNs. However, crafting customized CUDA operators for specific algorithms requires significant development effort and specialized expertise. Therefore, researchers are exploring intermediate optimization frameworks for kernel programming, such as Triton (Tillet et al., 2019) and TileLang (Wang et al., 2025), which aim to offer high optimization performance while being more accessible for researchers to develop. In short, the optimization levels of an algorithm are as follows:

$$\text{Torch} < \text{Triton} < \text{CUDA} \tag{24}$$

Generally speaking, the higher the optimization level of an algorithm, the faster its execution speed.

To provide a fairer evaluation of the acceleration gains brought by DSN parallelization, we expand the acceleration experiment in Table 8. Our discussion of the results is as follows:

Table 8: A comparison of the training time (in ms) and the ratio of memory (Mem.) and compute (Comp.) throughput to the theoretical peak (%) of different spiking neurons and implementation. For training time, the timestep 32^* is for Sequential CIFAR10 and $128 \sim 512$ are for CIFAR10.

Methods	Parallel	Level	Forward				Backward				Throughput rate	
			32^*	128	256	512	32^*	128	256	512	Mem.	Comp.
LIF (Abbott, 1999)	✗	Torch	4.90	19.53	41.31	77.42	5.84	25.34	52.77	101.88	31.02	15.93
LIF (Fang et al., 2023a)	✗	CUDA	0.59	0.56	0.60	0.58	0.27	0.25	0.25	0.25	47.13	18.40
PSN (Fang et al., 2023b)	✓	CUDA[1]	0.20	0.24	0.20	0.21	0.23	0.22	0.24	0.25	39.56	43.33
DSN (Ours)	✗	Triton	7.00	26.74	52.92	106.39	20.62	68.42	136.74	267.13	18.96	18.12
DSN (Ours)	✓	Triton	0.68	0.75	0.68	0.69	0.71	0.64	0.69	0.64	**58.58**	**44.10**

- The parallel training speed of DSN is higher than that of serial training. Notably, under serial training, the Triton implementation of DSN performs similarly to the Torch implementation of LIF. This is because the Triton kernel in the experiment is invoked step by step over time, rather than processing all timesteps in a single pass. As a result, the memory access and computation patterns across timesteps are essentially the same as those in the Torch implementation of LIF.
- The parallel training speed of DSN at Triton level is only slightly lower than the serial training speed of LIF at CUDA level. This is understandable because the optimization level of Triton is lower than that of CUDA. Although LIF can achieve acceleration by increasing the optimization level, the algorithm itself cannot be parallelized. As a result, its utilization of GPU resource is limited. A clear indication is that the memory throughput and compute throughput of LIF at CUDA level (47.13% / 18.40%) are even lower than those of DSN at Triton level (58.58% / 44.10%). Hence, since DSN is parallelized, if we also accelerate it at the CUDA level, the training speed is expected to surpass that of LIF at CUDA level.

Triton-based Operator. We found that the dynamic decay form of DSN matches the HGRN operator in flash-linear-attention library[2]. Therefore, in this paper, we leverage this Triton operator to enable parallel training of DSN.

B.2 Sequential CIFAR, ImageNet and CIFAR10-DVS

In this work, we set DSN hyperparameters $N = 4$, $k = 4$, and $\tau = 0.5$.

Sequential CIFAR. We use the width of the image as the sequence length ($L = 32$) to obtain a serialized version of CIFAR dataset. The model architecture is consistent with that of PSN (Fang et al., 2023b) as detailed in Table 9. For hyperparameter settings, the training is conducted over 256 epochs with a cosine decay learning rate schedule, starting at a maximum of 0.001. We set the batch size to 128 and select AdamW optimizer (Loshchilov & Hutter, 2019) with zero weight decay.

[1]The membrane potential in PSN can be expressed as a matrix multiplication, which enables efficient execution on GPUs using FP16 GEMM kernels and specialized Tensor Core hardware. Therefore, this constitutes acceleration at the CUDA level.

[2]https://github.com/fla-org/flash-linear-attention

Table 9: Configurations of Conv-based SNNs for Sequential CIFAR dataset. BN: BatchNorm, FC: Fully Connected.

Stage	Layer Specification	Configuration
1	Conv1D-BN-DSN Block $\times$ 3	Conv: (3, stride=1, padding=1), Dim: 128
	Average Pooling	Feature size: 32 $\rightarrow$ 16
2	Conv1D-BN-DSN Block $\times$ 3	Conv: (3, stride=1, padding=1), Dim: 128
	Average Pooling	Feature size: 16 $\rightarrow$ 8
3	Flatten-FC1-DSN-FC2	FC1: 1024 $\rightarrow$ 256, FC2: 256 $\rightarrow$ class_num

ImageNet and CIFAR10-DVS. For ImageNet, our experimental setup is identical to that of PSN (Fang et al., 2023b). For CIFAR10-DVS, we use AdamW (Loshchilov & Hutter, 2019) as the optimizer with a learning rate of 0.001, while keeping all other settings consistent with PSN.

B.3 Time-series Forcasting Tasks

We rely on two Pytorch-based frameworks to build the baseline networks: SnnTorch (Eshraghian et al., 2023) and SpikingJelly (Fang et al., 2023a). For SNNs with LIF neurons, we set the training timestep $T = 4$, the threshold $V_{\text{th}} = 1.0$, and the decay rate $\beta = 0.99$. For SNNs with DSN neurons, thanks to integer-valued training techniques, we do not directly expand timesteps to perform 0-1 encoding for temporal tasks. Instead, $N = 4$ is regarded as the expanded 4 timesteps. The architecture and size of DSN-based model are aligned with Lv et al. (2024). For training hyperparameters, we use a batch size of 128 and employ Adam optimizer (Kingma & Ba, 2015) with a learning rate of 1×10^{-4}. An early stopping strategy is implemented with a tolerance of 30 epochs. The experiments are conducted using 4 NVIDIA RTX A6000 GPUs.

To assess the performance of our model, we use the Root Relative Squared Error (RSE) and the coefficient of determination (R^2), defined as follows:

$$\text{RSE} = \sqrt{\frac{\sum_{m=1}^{M} ||\mathbf{Y}^m - \hat{\mathbf{Y}}^m||^2}{\sum_{m=1}^{M} ||\mathbf{Y}^m - \bar{\mathbf{Y}}||^2}}, \tag{25}$$

$$\text{R}^2 = \frac{1}{MCL} \sum_{m=1}^{M} \sum_{c=1}^{C} \sum_{l=1}^{L} \left[1 - \frac{(Y_{c,l}^m - \hat{Y}_{c,l}^m)^2}{(Y_{c,l}^m - \bar{Y}_{c,l})^2} \right]. \tag{26}$$

In these formulas, M is the size of the test set, C is the number of channels, and L is the prediction length. $\bar{\mathbf{Y}}$ represents the average of $\mathbf{Y}^m$. $Y_{c,l}^m$ denotes the l-th future value of the c-th variable in the m-th sample, while $\bar{Y}_{c,l}$ is its average across all samples. $\hat{\mathbf{Y}}^m$ and $\hat{Y}_{c,l}^m$ denote the ground truth values. Unlike Mean Squared Error (MSE) or Mean Absolute Error (MAE), these metrics are less sensitive to the absolute scale of the dataset, making them particularly well suited for time-series forecasting tasks.

Regarding the improvements in Spikformer (Zhou et al., 2023), in addition to replacing the spiking neurons, we also make architectural modifications to achieve better performance. Specifically, we expand the first DSN in the Feed-Forward Network (FFN) block to an enhanced DSN to improve the interaction between different neuron channels. However, this increases the total number of parameters in the FFN block by $16C^2$, where C is the number of channels. To maintain the same total parameter count $8C^2$ of the FFN block, we reduce the expansion ratio of the linear mapping from the usual 4 to 2. The architecture of the FFN block before and after modification is shown in Fig. 4.

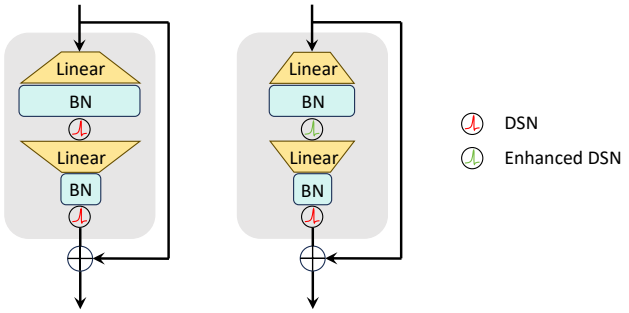

Figure 4: The FFN block in Spikformer with DSN before (left) and after (right) modification. BN: BatchNorm.

B.4 WIKITEXT-103

Considering the complexity of modeling language models, we use enhanced DSN in this experiment. Our language model is built by stacking DSN-based blocks:

$$X' = X + \mathrm{DSN}(\mathrm{LayerNorm}(X))W \tag{27}$$

In Eq. 27, before entering DSN neurons, the input X first passes through a $\mathrm{LayerNorm}$ layer to maintain training stability for deep networks. The spike-driven output signal undergoes sparse computation in a linear mapping layer with learnable parameters W. Additionally, we use a membrane shortcut (Hu et al., 2025) to achieve identity mapping (He et al., 2016) with spike-driven characteristics.

Our experiment is implemented on 8 NVIDIA A800 GPUs. The hyperparameters are largely based on S4 (Gu et al., 2022) and SpikingSSM (Shen et al., 2025), as shown in Table 10. The key difference is that we shorten the length of the training text to 1024. To maintain the number of tokens per training step, we increase the batch size per GPU to 8. Notably, we do not further explore the architecture design and hyperparameters of this experiment, which could be an avenue for future research.

Table 10: Configurations of DSN-based language model on WikiText-103.

Configurations	WikiText-103
Layer Depth	16
Model Dimension	1024
Learning Rate	5e-4
Learning Rate Schedule	Cosine Decay, with 500 warmup steps
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Weight Decay	0.01
Batch Size per GPU	8
Epochs	100

B.5 ANALYSIS OF ENERGY CONSUMPTION

We follow the method in (Yao et al., 2023a) to evaluate the energy consumption of the Sequential CIFAR network using different spiking neurons. Specifically, the energy consumption for floating-point operations (FLOPs) is calculated by $E_{MAC} \cdot FLOPs$, while the energy consumption for spike-based operations is calculated by $E_{AC} \cdot T \cdot R \cdot FLOPs$. Here, $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$ in 45nm technology. T denotes timestep and R denotes the spike firing rate. The FLOPs of the n-th Conv1D layer are $k_n \cdot d_n \cdot c_{n-1} \cdot c_n$, where k_n is the kernel size, d_n is the sequence channel number, c_{n-1} and c_n are the input and output convolution channel numbers, respectively. The FLOPs of the m-th fully connected layer are $i_m \cdot o_m$, where i_m and o_m are the input and output channels of the layers.

The energy consumption from LIF neurons itself is usually considered negligible compared to that of the network architecture, including convolution and fully connected layers. In contrast, PSN and DSN have more complex internal structures, leading to non-negligible energy consumption. We present a statistical method for FLOPs within spiking neurons and summarize it in Table 11.

Table 11: Statistical methods of FLOPs within spiking neurons. c: number of spiking neuron. T: Timestep. k: kernel size of causal convolution.

Spiking Neuron	Internal Structure	FLOPs
LIF (Abbott, 1999)	Update of Membrane Potential	$c \cdot T$
PSN (Fang et al., 2023b)	Update of Membrane Potential	$c \cdot T^2$
DSN (Ours)	Causal Conv1D	$k \cdot c \cdot T$
	Sigmoid Function	$c \cdot T$
	Update of Membrane Potential	$c \cdot T$

The spike firing rates of different layers[3] in Conv-based SNN for Sequential CIFAR using different spiking neurons are presented in Table 12. Our DSN exhibits a lower spike firing rate than that of LIF, which helps offset the additional energy cost introduced by the dynamic decay module.

Table 12: Spike firing rates of Conv-based SNN for Sequential CIFAR10 and CIFAR100. Convx: Conv1D of the x-th layer. FC: Fully Connected.

Dataset	Methods	Conv2	Conv3	Conv4	Conv5	Conv6	FC1	FC2	Average
Sequential CIFAR10	LIF (Abbott, 1999)	0.1511	0.1422	0.1811	0.1553	0.1457	0.0926	0.0647	0.1499
	PSN (Fang et al., 2023b)	0.2200	0.3101	0.1575	0.1542	0.1516	0.1439	0.1239	0.2143
	DSN (Ours)	0.1349	0.1337	0.1301	0.1301	0.0982	0.0380	0.0484	**0.1238**
Sequential CIFAR100	LIF (Abbott, 1999)	0.2264	0.1281	0.1881	0.1581	0.1561	0.1018	0.1584	0.1698
	PSN (Fang et al., 2023b)	0.3221	0.2127	0.1887	0.1682	0.1509	0.1735	0.1458	0.2229
	DSN (Ours)	0.1384	0.1420	0.1404	0.1349	0.1240	0.0362	0.0973	**0.1324**

B.6 Approximation Experiment

Dynamic decay adaptively retains part of historical information stored in the membrane potential based on changing input. From the perspective of approximation, dynamic decay is powerful to simulate the behaviors of spiking neurons with various internal structures. During training, the spiking neuron learns to construct different reset mechanisms to model different input by regulating decay. For example, if the information is better suited to be encoded in the form of hard reset, the spiking neuron only needs to approximate a binary classifier to decide whether to set α_t to be constant β or 0. This plasticity of dynamics potentially breeds rich memory abilities. To verify the expressiveness of spiking neurons with dynamic decay, we design an experiment of using dynamic decay to approximate the behaviors of multiple LIF neurons with hard or soft reset.

Overview. To begin with, we manually construct two distinct datasets with a timestep of $T = 128$ named A and B, and split them into training and test set. Dataset A has input signals following a normal distribution with parameters (μ, σ^2), while dataset B is a collection of more regular signals including sine functions, sigmoid functions, step functions and Poisson encoding with different parameters. Afterwards, these signals are input into 6 LIF neurons with different reset mechanisms and membrane time constants. Then, we apply dynamic decay across $C = 6$ channels with the same input signals to approximate the membrane potential with that of the LIF neurons described above, using the Mean Squared Error (MSE) loss function. Lastly, we calculate the spike firing accuracy of the test set as evaluation metric.

Datasets. The normal distribution parameters of Dataset A are $\mu = 1, \sigma = 2$. A total of 11,000 samples are collected, with a training-to-test ratio of $10 : 1$. The signal generation methods of

[3]Notably, the input to the first convolutional layer are floating-point numbers of the original sequence, rather than processed spikes. Therefore, this layer is not involved in the calculation of the spike firing rate.

Dataset B is shown in Table 13. Each type of signal generates 200 samples (totally 800 samples), with 10% randomly selected as test set and the remaining samples used for training.

Table 13: The signal generation methods of Dataset B. $x = 0, 1, ..., T-1$. The notation $[a : b : c]$ means selecting c evenly spaced values from a to b. For example, $[5 : 15 : 5]$ is equal to $5, 7.5, 10, 12.5, 15$. Different parameters can be combined with each other to obtain samples with different characteristics, with the corresponding c multiplied. For Poisson Encoding, $\mathrm{Random}(\cdot)$ denotes the random sampling of a floating-point number from the interval $[0, 1]$, and each set of parameters is repeated 8 times to generate 8 samples.

Signal Type	Formulation	Specification
Sine Function	$\text{input} = A\sin(\omega x) + B$	$\omega = 2\pi\frac{C-1}{T-1}, A = [-2:3:5], B = [-2:3:8], C = [5:15:5]$
Sigmoid Function	$\text{input} = A \cdot \mathrm{Sigmoid}(x')$	$x' = \frac{20}{T-1}x - 10 + B, A = [-2:5:10], B = [10:10:20]$
Step Function	$\text{input} = A \cdot \mathrm{Heaviside}(x')$	$x' = x - B, A = [-2:5:10], B = [0:T:20]$
Poisson Encoding	$\text{input} = A \cdot \mathrm{Heaviside}(p)$	$p = \mathrm{Random}(x) - p_0, A = [-1:5:5], p_0 = [0.3:1:5]$

Spiking Neurons. We set the threshold of LIF neurons to be 1, and the structure of dynamic decay is as follows:

$$\mathbf{X}'_t = \mathrm{CausalConv1D}_{\mathrm{up}}(\mathbf{X}_{t-k+1:t}) \tag{28}$$

$$\mathbf{X}''_t = \mathrm{ReLU}(\mathbf{X}') \tag{29}$$

$$\mathbf{X}'''_t = \mathrm{CausalConv1D}_{\mathrm{down}}(\mathbf{X}''_{t-k+1:t}) \tag{30}$$

$$\boldsymbol{\alpha}_t = \mathrm{Sigmoid}(\mathbf{X}'''_t)^{1/\tau} \tag{31}$$

Here, $\mathbf{X}_t \in \mathbb{R}^{C\times 1}$, and $\mathbf{X}_{t-k+1:t}$ denotes k inputs from $\mathbf{X}_{t-k+1}$ to $\mathbf{X}_t$. $\mathrm{CausalConv1D}(\cdot)$ is causal 1D convolution and the indices up and down represent the expansion of the input channels from C to eC, and the reduction from eC to C, respectively. τ is a hyperparameter to fine-tune $\boldsymbol{\alpha}_t$. In this experiment, we set $k = e = 8$, and $\tau = 0.5$.

Training. We set a batch size of 128 and employ Adam optimizer (Kingma & Ba, 2015) with a cosine decay schedule whose peak learning rate is 1×10^{-2}. The training epochs are 100. To align with the techniques used in the main text, we also conducted experiments using dynamic decay to approximate integer-valued spikes (we set $N = 4$). Since integer-valued spikes are only meaningful in the case of soft reset (Yao et al., 2025), we fit only LIF neurons with soft reset in this case.

Results and Discussions. Results in Table 14 and Fig. 5 show that dynamic decay generally fits well to LIF neurons with different reset structures under various types of input signals, indicating its potential for expressiveness. When the spikes become integers, the fitting accuracy of dynamic decay further improves on both datasets, supporting our view that the integer-valued training technique and dynamic decay have a complementary effect in terms of expressiveness. Specifically, there are two details that merit our attention:

- As the membrane time constant increases, the fitting accuracy declines. This could be due to the growing influence of historical information on the integration mechanism of the spiking neuron, and modeling such information has always been a challenging task. However, in the current modeling of the LIF model, the value of τ_m typically does not exceed the range specified in Table 14 (usually $\tau_m = 2$ in (Yao et al., 2023a; 2025)), and our focus is on more general fitting scenarios. Additionally, the introduction of integer-valued spikes can significantly suppress this fitting error.
- When the membrane potential approaches the threshold, the error between the membrane potential predicted by dynamic decay and the true value generated by the LIF neuron is small for noise that follows a normal distribution. However, for a sine wave signal, the error between the two is larger (see Fig. 5). We speculate that the cause of this difference lies in the fact that the proportion of data near the threshold is smaller for the sine signal compared to the noise signal with a mean μ equal to the threshold. This makes it more difficult for dynamic decay to learn how to handle membrane potential fluctuations near the threshold.

Table 14: Experimental results of applying dynamic decay to approximate various LIF neurons with reset mechanisms on manually constructed datasets with different signals. Each channel of the parallel spiking neuron with dynamic decay is fitted with a LIF neuron. We report the spike firing accuracy (%) across 6 different channels and average them. *: results with integer-valued spike.

Channel	LIF neurons to fit	Dataset A	Dataset B	Dataset A*	Dataset B*
1	hard reset, $\tau_m = 4/3$	99.49	98.36	—	—
2	hard reset, $\tau_m = 2$	95.10	95.50	—	—
3	hard reset, $\tau_m = 4$	85.87	90.18	—	—
4	soft reset, $\tau_m = 4/3$	99.03	98.20	99.30	99.01
5	soft reset, $\tau_m = 2$	93.87	96.40	98.50	98.14
6	soft reset, $\tau_m = 4$	84.83	91.65	97.59	97.82
Average	—	92.97	95.05	98.46	98.32

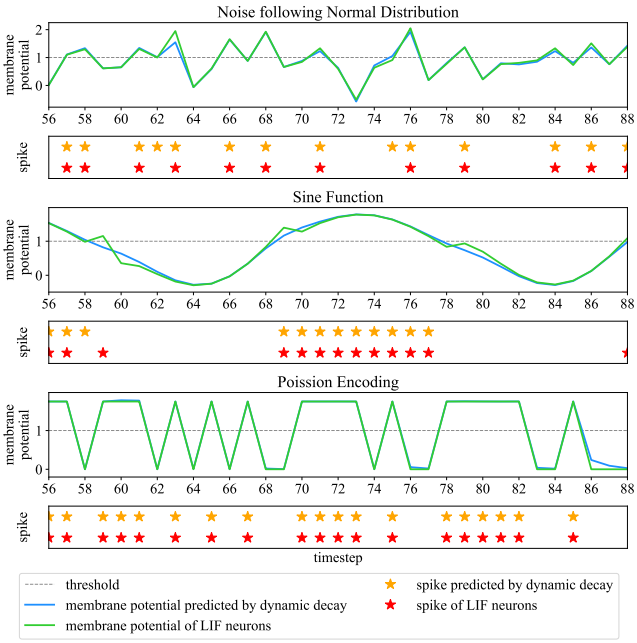

Figure 5: Signal responses including membrane potential and spike for LIF neuron and its dynamic decay prediction on channel 2. Subplots from top to bottom depict the responses to a noise following normal distribution, sine function, and Poisson encoding.

C LIMITATIONS

Design of Dynamic Decay. In this paper, we propose using causal convolution to model the relationship between decay factors and inputs. In fact, dynamic decay is a design paradigm that allows for various concrete implementations. Developing more effective dynamic decay mechanisms to improve the internal dynamics of spiking neurons is a promising direction for future research.

Operator Optimization. DSN employs Triton-based operators to strike a balance between training efficiency and ease of development. With more effort invested in developing CUDA-level operators, the training speed and GPU resource utilization could be further improved.

Neuromorphic Chip Deployment. There are still some gaps before DSN can be deployed on neuromorphic chips. On the one hand, according to existing research (Yao et al., 2025), the integer output can be converted to asynchronously emitted spikes with five key advantages when deployed on neuromorphic chips. On the other hand, further algorithmic optimizations might help avoid complex computations within DSN. For example, instead of multiplying between floating point weights and activations, the weights could be quantized to ternary values (-1, 0, 1) (Ma et al., 2024;

Zhu et al., 2024a), or a spiking function could be inserted to convert floating point inputs into spikes in advance. For the sigmoid function, the base could be replaced with 2 to enable lightweight computations using shift operations (Tang et al., 2025).