NEURAL INFORMATION PROCESSING SYSTEMS

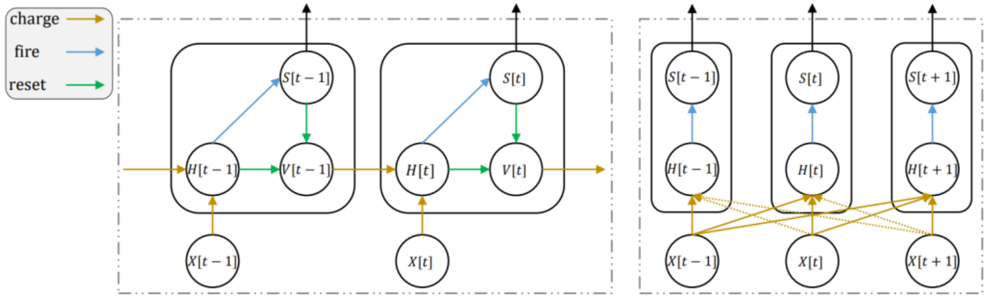
Multiplication-Free Parallelizable Spiking Neurons with Efficient Spatio-Temporal Dynamics

Peng Xue, Wei Fang∗, Zhengyu Ma, Zihan Huang, Zhaokun Zhou,
Yonghong Tian,Timothée Masquelier, Huihui Zhou∗

Parallel Spiking Neuron

The behaviors of vanilla spiking neurons can be described by three discrete-time equations:

$$H[t] = f(V[t-1], X[t]), \qquad (1)$$
$$S[t] = \Theta(H[t] - V_{th}), \qquad (2)$$
$$V[t] = \begin{cases} S[t] \cdot V_{reset} + (1 - S[t]) \cdot H[t], \text{hard reset} \\ H[t] - S[t] \cdot V_{th}, \text{soft reset} \end{cases}, \qquad (3)$$

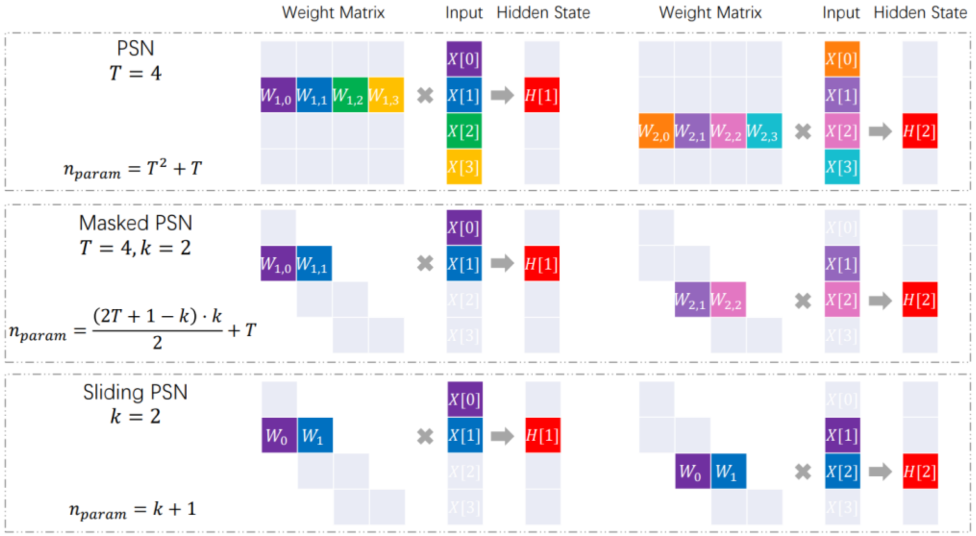


(a) The vanilla spiking neuron (b) The parallel spiking neuron

Fang et al [1] found that the neuronal dynamics could be expressed in a non-iterative form after removing the reset equation Eq.(3), and thus propose the Parallel Spiking Neuron (PSN) family. The simulation speed of PSN is much faster than the vanilla spiking neurons.

PSN's Problem

1. PSN family introduces the dense floating-point matrix multiplication in the neuron layer, which relies on massive multiply-accumulate operations and is hardware-unfriendly.
2. PSN family uses the channel-share weights, which fails to capture the subtle disparity of features in channels.
3. Sliding PSN only achieves stable performance with a large neuron order k, which is proportional to the inference memory and energy.



[1] Fang et al. Parallel spiking neurons with high efficiency and ability to learn long-term dependencies. Advances in Neural Information Processing Systems, 36, 2023.

Our solution

We introduces the Multiplication-Free Channel-wise Parallel Spiking Neurons (**mul-free channel-wise PSN**). The model features several key innovations:

1. **Channel-wise mechanism**: enabling the efficient capture of spatial-temporal dynamics without increasing FLOPs.
2. **Dilated convolution**: allowing the temporal receptive field to expand rapidly with network depth.
3. **Power-of-2 quantization**: converting the multiplication operation into a simple **bit-shift** (for integer inputs) or a **low-bit integer addition** to the exponent (for FP32/FP16 inputs).

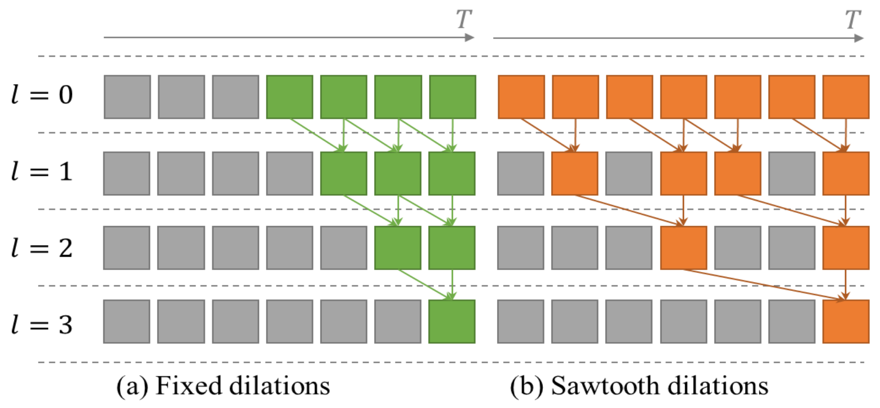
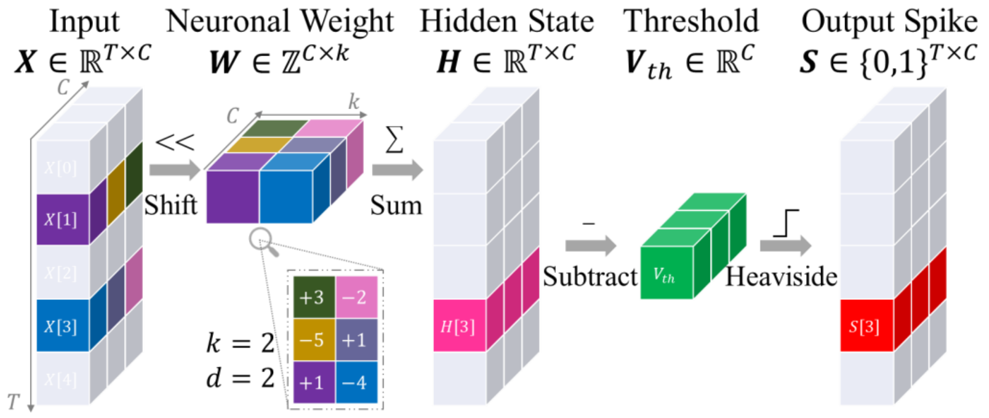


Figure 2: The temporal receptive field increases with depths at a slow rate in the sliding PSN with (a) fixed dilations and a fast rate in the channel-wise PSN with (b) sawtooth dilations.

Neuronal Dynamic

$$H[t][c] = \sum_{i=0}^{k-1} X[t-(k-1-i)\cdot d][c] \ll \log_2\left(W_q[c][i]\right), \quad (17)$$



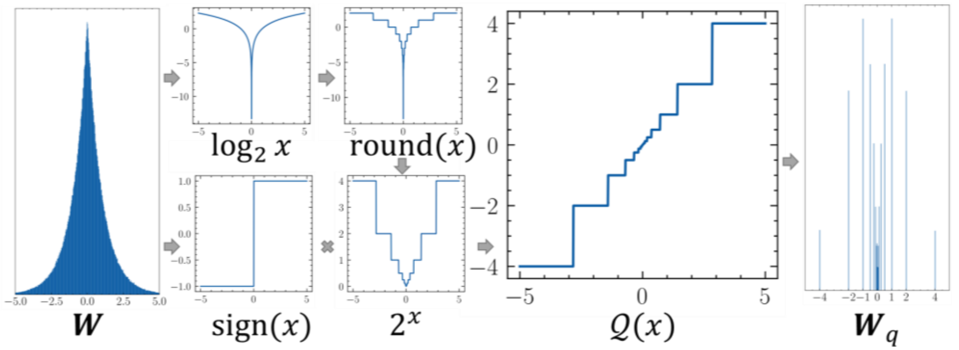


Figure 3(a): The workflow of power-of-2 quantizer.

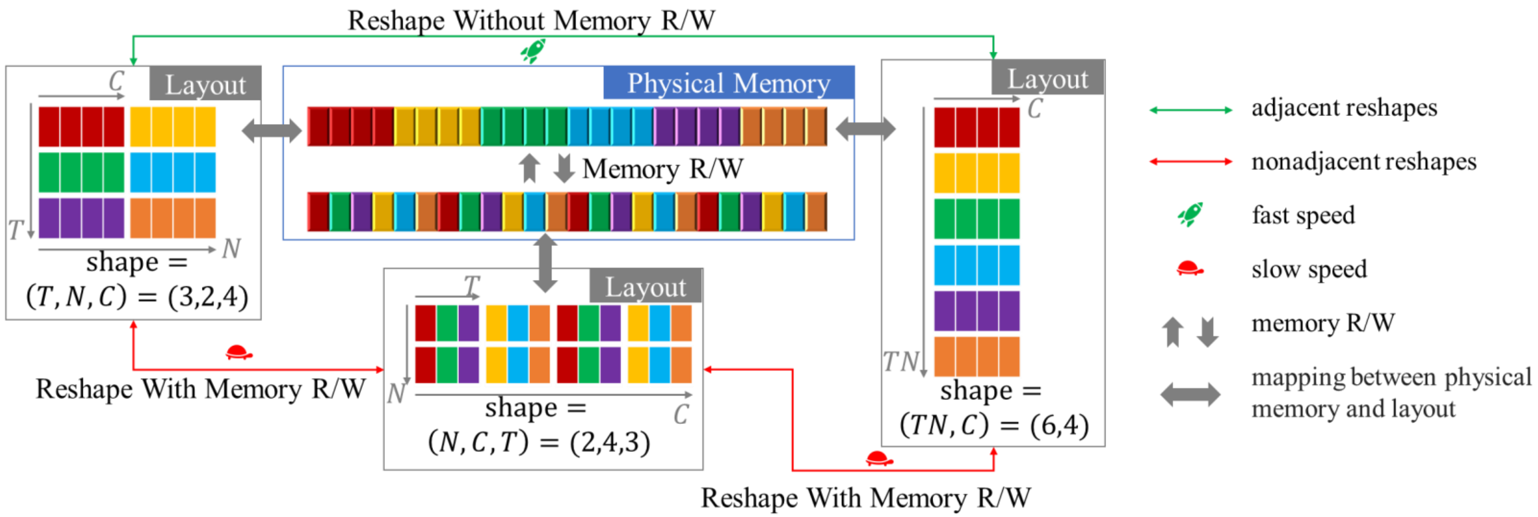
Training Acceleration

SNN data layouts

- Time-first layout: shaped as $(T, N, C, \dots)$
- Time-last layout: shaped as $(N, C, \dots, T)$

Vanilla implementation of mul-free channel-wise PSN (Eq.(17))

- Vanilla implementation: PyTorch's 1-D Convolution (*Conv1d*), which requires the shape of inputs as (N, C, T).
- Exiting unavoidable reshape operation $(T, N, C, \dots) \rightleftharpoons (N*, C, T)$ and $(N, C, \dots, T) \rightleftharpoons (N*, C, T)$ before and after the *Conv1d*.
- Reshape operations of the nonadjacent dimensions require costly memory reading/writing operations.



Training Acceleration

Efficient Implementations

- **Time-first layout**
  - Using a custom CUDA kernel to directly perform convolutions along the *T* dimension.
  - Using PyTorch's vectorising map function (*Vmap*) to parallelize computations over the *C* dimension, and the matrix multiplication (*MM*) to process other dimensions.
- **Time-last layout**
  - Custom CUDA kernel or *Vmap* + *MM*, similar to time-first implementation.
  - Using 2-D convolution to implement the 1-D convolution, with the weight and stride as 1 to handle the " ... “ dimension.
  - Using *Vmap* to vectorize the *C* dimension and *conv1d* to handle other dimensions.

Autoselect acceleration algorithm

- Automatic choose the fastest implementation.

Algorithm 1 Autoselect acceleration algorithm

Require: An SNN stacked with L layers $\{M_0, M_1, ..., M_{L-1}\}$. The layer M_l has n_l optional acceleration methods. The input sequence $\boldsymbol{X}_0$.

```
for Ω ← {time-first, time-last}
    Reshape X_0 to Ω
    t_Ω = 0
    for l ← 0, 1, ...L − 1
        for i ← 0, 1, ..., n_l − 1
            Record the current time T_0
            Execute the forward propagation Y_l = M_l(X_l)
            Record the current time T_1
            Randomize a tensor Z_l with the same shape as Y_l
            Record the current time T_2
            Execute the backward propagation M'_l(Z_l)
            Record the current time T_3
            t_{l,i} = T_1 − T_0 + T_3 − T_2
        Choose the faster method a_{Ω,l} = argmin_i(t_{l,i})
        t_Ω ← t_Ω + min(t_{l,i})
```

Outputs: The layout $\Omega^* = \text{argmin}_\Omega(t_\Omega)$ and the acceleration method $a_{\Omega*,l}$ for M_l

Results

Static and Neuromorphic Data Classification

Table 2: Comparison with the state-of-the-art SNN methods on the SHD dataset.

Method	Network	Parallel	Accuracy(%)
Hammouamri et al. [39]	Two-layer FC + LIF + Learned Delay	✗	95.07 ± 0.24
Li et al. [30]	Four-layer FC + RPSU	✓	92.49
Chen et al. [29]	Two-layer FC + PMSN	✓	95.10
Yarga and Wood [16]	Two-layer FC + Stochastic PSN + Learned Delay	✓	95.01
Ours	Two-layer FC + Mul-free Channel-wise PSN + Learned Delay	✓	$\mathbf{95.50 \pm 0.36}$

Table 3: Comparison of test accuracy (%) of spiking neurons on sequential CIFAR datasets.

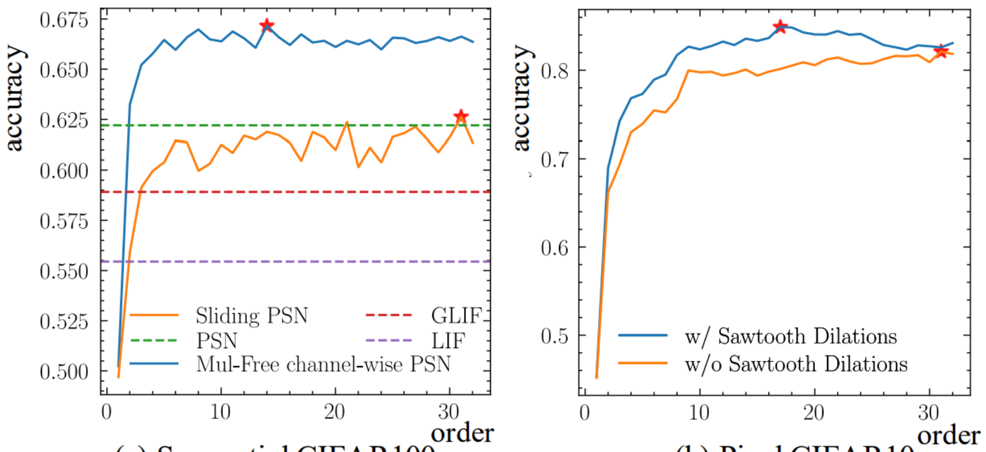
Datasets	**Ours**	**PMSN**[29]	**PSN**[15]	**Masked PSN**[15]	**Sliding PSN**[15]	**GLIF**[13]	**PLIF**[5]	**LIF**
Sequential CIFAR10	**91.17**	90.97	88.45	85.81	86.70	83.66	83.49	81.50
Sequential CIFAR100	**66.21**	66.08	62.21	60.69	62.11	58.92	57.55	53.33

Table 4: Comparison with the state-of-the-art ANN and SNN methods on the DVS-Lip dataset.

Method	Frontend	Backend	Accuracy(%)
Tan et al. [38]	ResNet-18 (ANN)	BiGRU (ANN)	72.1
Bulzomi et al. [43]	Modified Spiking ResNet + PLIF	FC (Stateful Synapses)	60.2
Dampfhoffer et al. [42]	ResNet-18 (ANN)	BiGRU (ANN)	75.1
	Spiking ResNet-18 + PLIF	FC (Stateful Synapses)	68.1
	Spiking ResNet-18 + PLIF	SpikGRU2+ (Bi-direction + Sigmoid Gates + Ternary Spikes)	75.3
Ours	Modified Spiking ResNet-18 + Mul-free Channel-wise PSN	FC (Stateful Synapses)	70.9

Results

Ablation Study



(a) Sequential CIFAR100 (b) Pixel CIFAR10

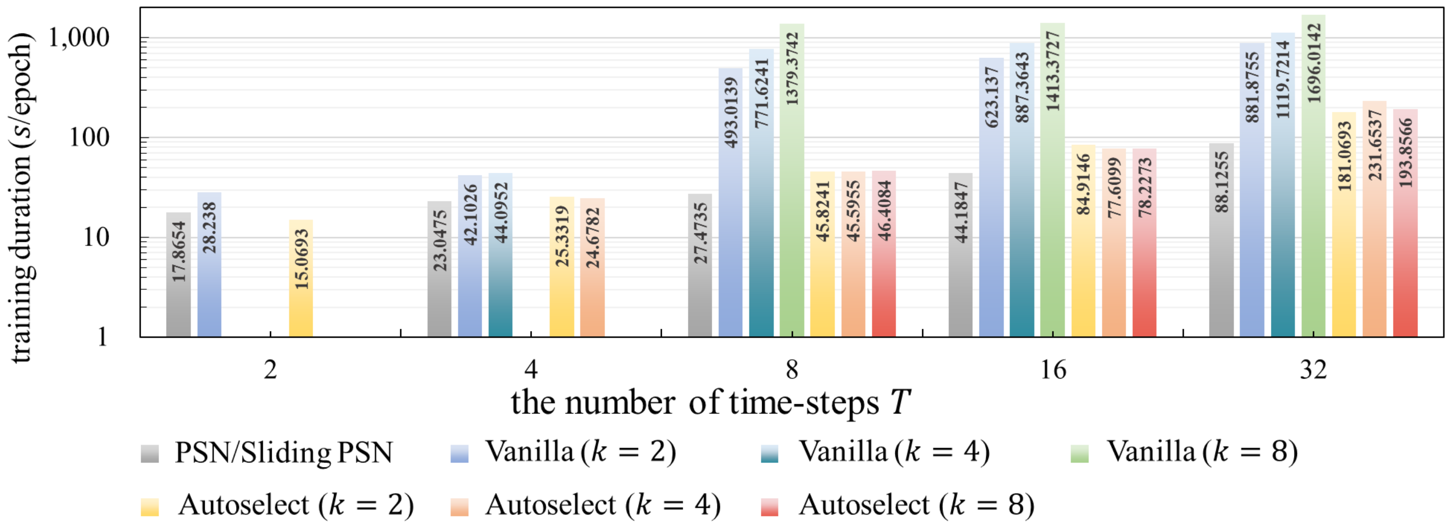
Step-by-step Inference Memory

Neuron	k	Accuracy(%)	Memory(MB)
Sliding PSN	32	62.11	2635
Ours	4	65.77	547

Computational Energy

Neuron Layer			**Synaptic Layer**		**Total Energy**
Neuron	Operations	Energy (μJ)	Operations	Energy (μJ)	(μJ)
PSN	1.91×10^7 MUL 1.97×10^7 ADD	88.56	0.041×10^6 FLOPs 3.194×10^6 SOPs	3.06	91.62
Ours	7.32×10^6SHIFT 7.92×10^6ADD	8.08	0.041×10^6 FLOPS 2.660×10^6 SOPs	2.58	10.66

Training Speed



Thanks!