SpikingSSMs: Learning Long Sequences with Sparse and Parallel Spiking State Space Models

Shuaijie Shen[*,1,2], Chao Wang[*,1,2], Renzhuo Huang[1,2], Yan Zhong[2,3], Qinghai Guo[2], Zhichao Lu[4], Jianguo Zhang[†,1,5], Luziwei Leng[†,2]

[1] SUSTech | [2] Huawei | [3] Peking University | [4] CityU | [5] Pengcheng Lab

State Space Models (SSMs)

$$h_t = Ah_{t-1} + Bx_t$$
$$y_t = Ch_t + Dx_t$$

SSMs

Utilizing FFT for parallel training with $\mathcal{O}(L \log L)$ complexity.

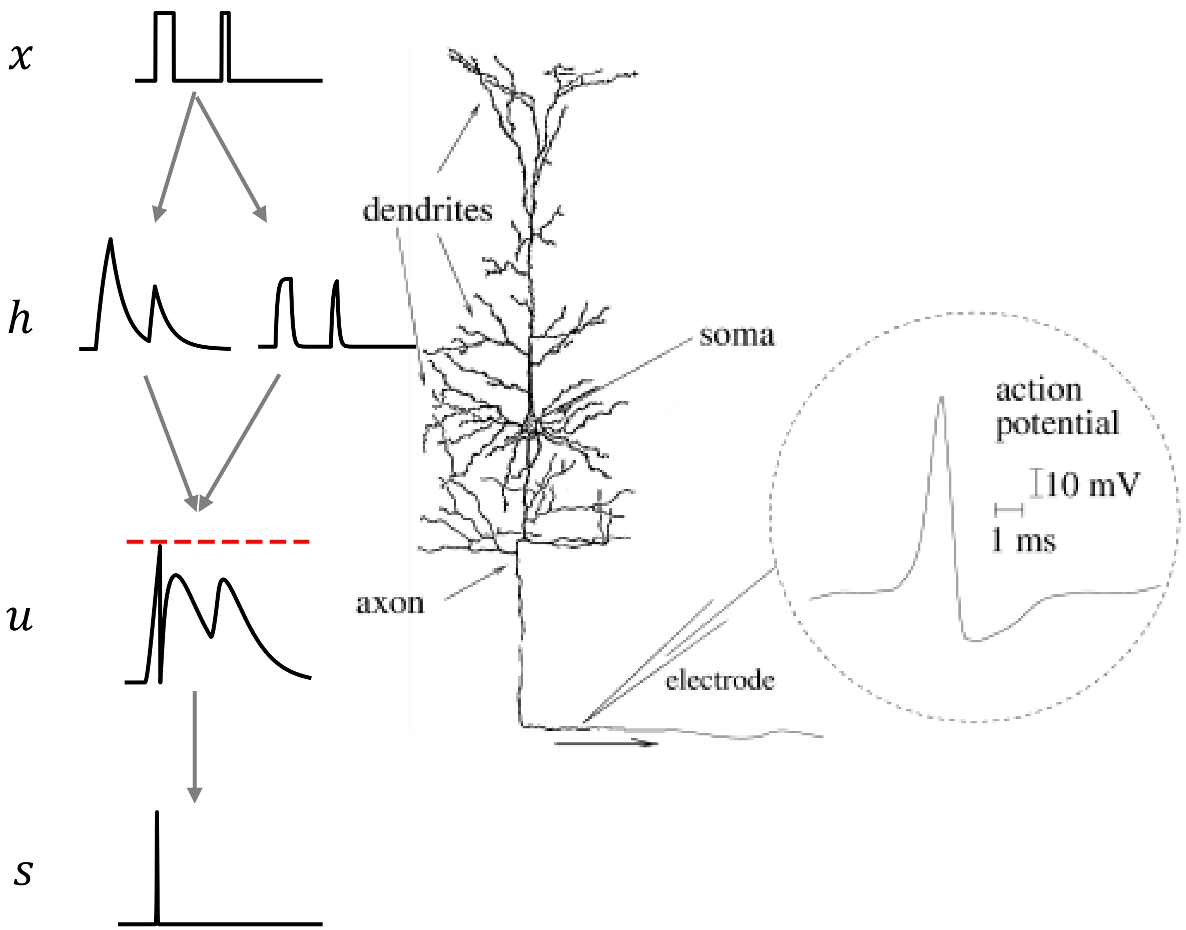
Parallel via FFT

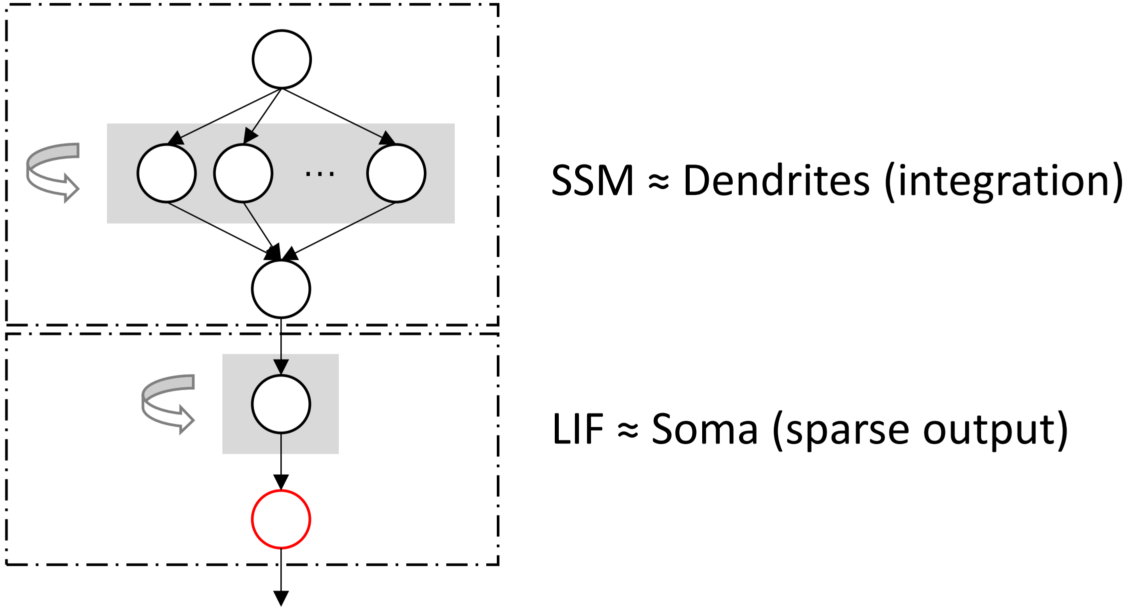
e.g., S4、S4D

Long-Range Dependency Modeling

SSMs Meet Dendritic Computation

Biological Inspiration: Dendritic neurons process multi-timescale inputs.





SpikingSSMs Design:

SSM hidden states (h) model dendritic integration.

LIF neuron acts as soma for sparse spiking.

Parallel Challenge of Spiking Neurons

$$u'_t = \tau u_{t-1} + y_t$$

$$\boxed{s_t = \begin{cases} 1, & u'_t \geq v_{th} \\ 0, & u'_t < v_{th} \end{cases}}$$

$$u_t = (1 - s_t)u'_t + s_t v_r$$

Nonlinear !

LIF (Leaky Integrate and Fire) neuron with hard reset mechanism

Problem: LIF neurons require iterative reset, limits parallel training.

Note: $s_{1:T} = f_\theta(y_{1:T}; v_{th})$

$\downarrow$

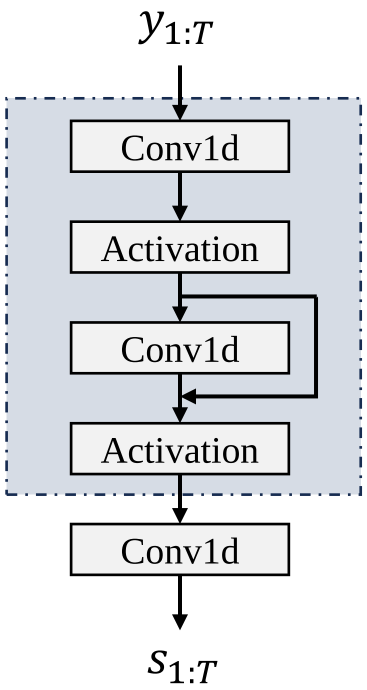
A model?

Surrogate Dynamic Network (SDN)

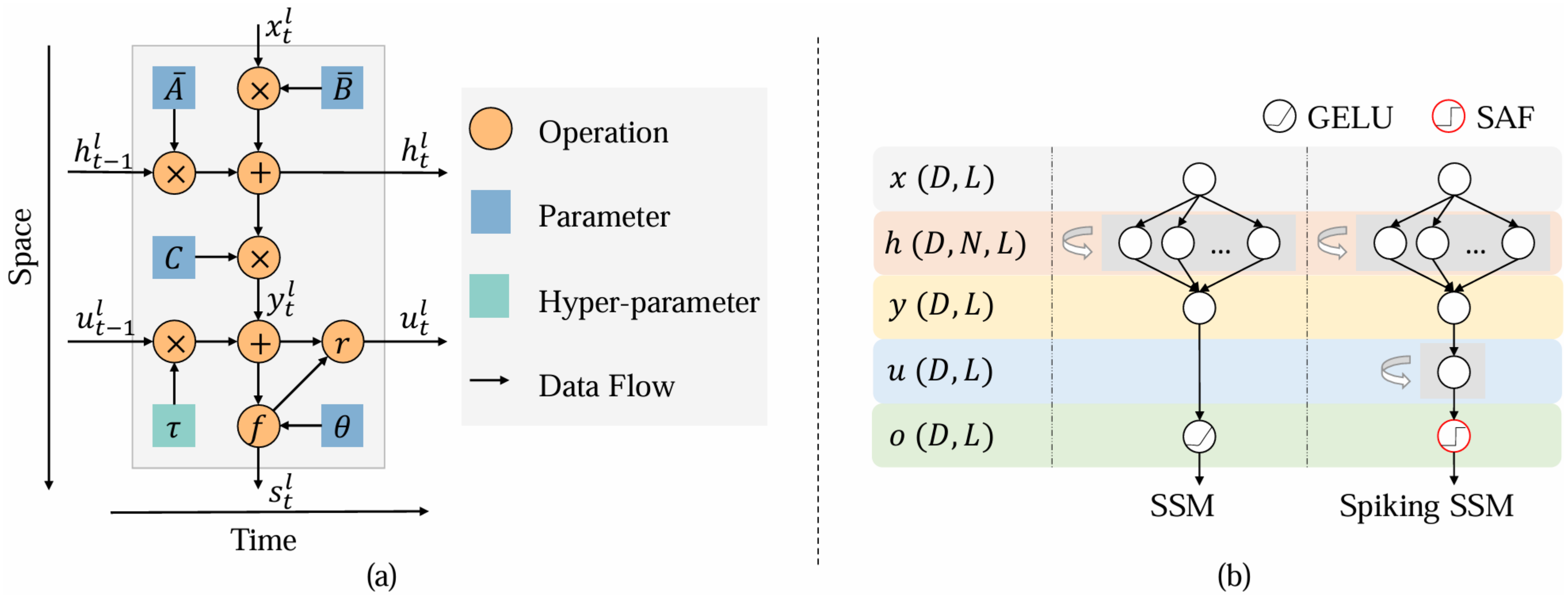
- Lightweight CNN predicts spikes in parallel.
- Architecture: 3-layer 1D-CNN (<200 params).

Advantage:

- Parallel training
- Can be Removed in iterative inference



Architecture



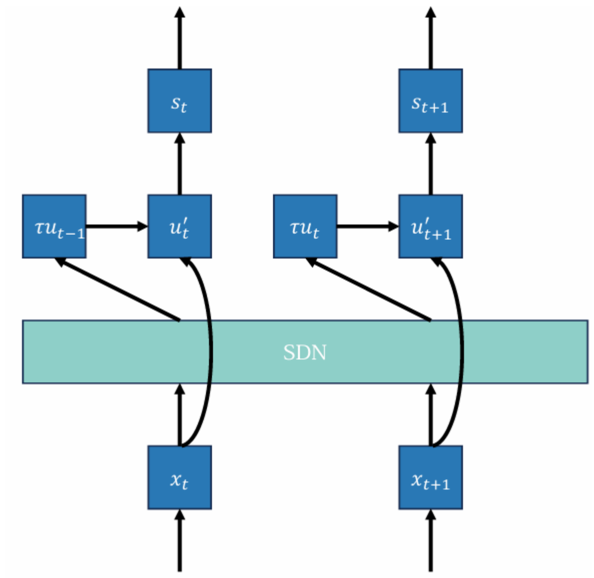
Computational Graph

Complex computational graph for SDN in training task models

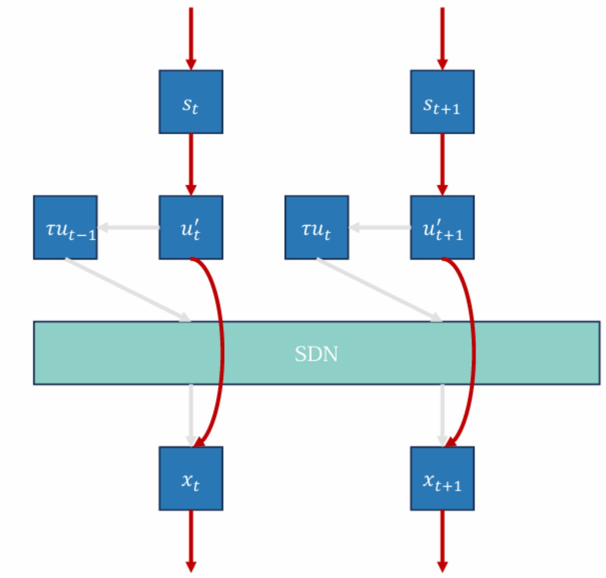
Solution:

$$s_{1:T} = f_\theta(y_{1:T}; v_{th}) \longrightarrow \begin{array}{l} \tau u_{0:T-1} = f_\theta(y_{1:T}; v_{th}) \\ s_{1:T} = \mathrm{H}(\tau u'_{0:T-1} + x_{1:T} - v_{th}) \end{array}$$

without gradient propagation



Forward of SDN



Backward of SDN

Learnable Threshold

Optimizing the threshold can improve network performance.

Can SDN approximate neuron dynamics with different threshold?

If both the initial membrane potential and the reset potential are 0:

Property 1. The ratio of inputs and threshold determines the dynamic process of the neuron.

$$s_{1:T} = f_\theta(y_{1:T}; v_{th}) = f_\theta(\alpha y_{1:T}; \alpha v_{th})$$

Property 2. The threshold scales the distribution of the input.

$$s_{1:T} = f_\theta(y_{1:T}; v_{th}) = f_\theta\left(\frac{y_{1:T}}{v_{th}}; 1\right)$$

SDN Accelerates Training by 100×

Method	Speed			
	L = 1K	L = 2K	L = 4K	L = 8K
BPTT	1370	2900	8040	25600
SLTT	1210	2720	7740	25600
Ours	183	196	200	253
Ratio	7.5 ×	15.0 ×	40.2 ×	101.2 ×

101× faster than BPTT.

Table1: Comparison on training speed of different methods. The input has a batch size of 64. Training with SDN achieves significant acceleration, the speed up ratio amplifies with increasing sequence length.

	Model	Accuracy (%)	Sparsity (%)	Speed (ms)
	LIF	85.45	12.08	1480
Fixed	SDN	85.57	11.92	230
Train from scratch	SDN-S	81.52	18.30	285

SDN matches iterative LIF (85.6% vs. 85.5% accuracy).

Table2: Performance comparison on the sCIFAR10 dataset.

Results

	Model	SNN	LISTOPS	TEXT	RETRIEVAL	IMAGE	PATHFINDER	Path-X	AVG
	Transformer	No	36.37	64.27	57.46	42.44	71.40	—	53.66
	LMUFormer	No	34.43	68.27	78.65	54.16	69.90	—	59.24
	S4D-Lin	No	**60.52**	**86.97**	**90.96**	**87.93**	**93.96**	**92.80***	**85.52**
	Spiking LMUFormer	Yes	37.30	65.80	79.76	55.65	72.68	—	60.20
	Binary S4D	Yes	54.80	**82.50**	85.03	82.00	82.60	61.20	74.69
	S6-based SNN	Yes	55.70	77.62	88.48	80.10	83.41	—	72.55
Fixed threshold	SpikingSSM-VF	Yes	59.93	82.35	88.20	86.81	**93.68**	94.80	84.30
	(spiking rate)		(0.13)	(0.10)	(0.06)	(0.22)	(0.07)	(0.10)	(0.11)
Learnable threshold	SpikingSSM-VT	Yes	**60.23**	80.41	**88.77**	**88.21**	93.51	**94.82**	**84.33**
	(spiking rate)		(0.14)	(0.06)	(0.06)	(0.15)	(0.08)	(0.10)	(0.10)

Table 3: Performance comparison of SpikingSSM and previous works on the LRA dataset. *Since the original S4D-Lin failed in the Path-X task, we instead present the result of another close variant S4D-Inv. -VF and -VT denote fixed and trainable threshold, respectively. Furthermore, we take the 50% accuracy for the absence of Path-X accuracy as did in the work of S4D, then compute the overall average metrics across all tasks as AVG. The spiking rate for each task have also been calculated, which is indicated by blue font.

Model	SNN	PPL	Parameters
Transformer	No	20.51	231M
S4	No	20.95	249M
SpikeGPT	Yes	39.75	213M
SpikingSSM	Yes	33.94	75M

Table 4: Performance comparison of SpikingSSMs with previous works on WikiText-103 dataset.

Energy Efficiency

Spikes × Weights (in SNN): synaptic Accumulation(AC)

1 × float : + weight
0 × float : no operation

Activations × Weights (in ANN): Multiply-and-Accumulate(MAC)

Major Computation: feature-mix layers (after SSMs)

For Wiki-Text-103 with $L = 8\mathrm{K}$ and 16 layers,
projecting inputs from $d = 1024$ to $d = 2048$
sparsity: 73.6 %

Done via multiplication(ANN):	$275.2G$ MAC	$1.265J$
Done via accumulation(SNN):	$72.66G$ AC	$65.40\ mJ$

Theoretical Saving: 95% energy reduction.

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

[1] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, San Francisco, CA, USA, 2014, pp. 10-14, doi: 10.1109/ISSCC.2014.6757323.

Conclusion & Future Work

- Key Takeaways:
  - SpikingSSMs: Combines SSMs’ performance with SNNs’ sparsity.
  - SDN: Enables 100× faster training without sacrificing accuracy.
- Future Work:
  - Scaling to billion-parameter models.
  - **Exploring Brain-Inspired Large Language Models.**

Thank You! Questions?