RELIF: EXPLOITING TEMPORAL PARALLELISM THROUGH FINITE-TIME COALESCENCE IN SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks carry recurrent neuronal states across time, making exact temporal execution serial even when the surrounding network is highly parallel. We introduce ReLIF, which rectifies the recurrent state before integration. Under a shared input sequence, sufficiently negative inputs erase the carried state through rectification, while sufficiently positive inputs trigger a common hard reset, allowing executions from different incoming membrane states to converge to the same zero state. We formalize this event as finite-time state coalescence and develop Coalescent Chunking, an exact execution scheme that precomputes boundary-independent suffixes in the forward pass and composes independent gradient intervals in the backward pass. ReLIF matches LIF accuracy across vision backbones and shows better performance in language modeling. In kernel-level measurements, parallel ReLIF reaches up to $38.07\times$ forward and $17.71\times$ backward speedup over SpikingJelly CuPy LIF. Across three SpikeTransformer++ scales, the complete stack reaches $2.92\times$–$3.20\times$ inference and $1.95\times$–$2.14\times$ training throughput over paired SpikingJelly CuPy LIF.

1 INTRODUCTION

Spiking neural networks (SNNs) have been applied to a range of computer vision and language modeling tasks (Fang et al., 2021a; Zhou et al., 2023; Yao et al., 2023; Shen et al., 2024; Zhu et al., 2023). SNNs are also motivated by event-driven computation and their potential for energy-efficient neuromorphic processing (Roy et al., 2019). As these models scale, their convolutional layers, attention, and feed-forward projections expose substantial parallelism to modern accelerators. The neuronal state, however, continues to advance one time step at a time, making temporal execution a serial bottleneck. For a sequence of length T, hard-reset LIF execution contains a dependency chain of length T, which limits accelerator utilization as SNNs move toward longer sequences and larger backbones.

The difficulty comes from the state carried across time steps. At each step, the neuron receives an input current while carrying forward the membrane state produced by the preceding step. Parallel execution becomes possible when a contiguous time segment can identify the point after which its outputs no longer depend on the incoming membrane state. We refer to such a contiguous time segment as a temporal chunk and to this point as a coalescence point. The desired dynamics preserve integration, leakage, threshold firing, and reset while confining the influence of the boundary state to a short, identifiable interval.

ReLIF exposes this structure by rectifying the carried state before integration:

$$h_t = \left(1 - \frac{1}{\tau_m}\right) \mathrm{ReLU}(v_{t-1}) + \frac{1}{\tau_m} x_t, \qquad s_t = \mathbf{1}\{h_t \geq \theta\}, \qquad v_t = (1 - s_t) h_t, \tag{1}$$

where x_t is the input current, $\tau_m > 1$ is the membrane time constant, and $\theta > 0$ is the firing threshold. We write

$$\gamma = 1 - \frac{1}{\tau_m}, \qquad \eta = \frac{1}{\tau_m}, \qquad \gamma + \eta = 1. \tag{2}$$

The effective state $z_t = \mathrm{ReLU}(v_t)$ always lies in $[0, \theta)$. If we partition the sequential time steps into contiguous temporal chunks, each chunk can start from the same bounded uncertainty set. Under a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

shared input sequence, its reachable trajectories can merge exactly at zero through rectification or hard reset. Once they merge, the remaining states and spikes are independent of the incoming boundary. We call the first such event finite-time state coalescence. The implementation and experiments use $\tau_m = 2$, for which $\gamma = \eta = 0.5$.

We propose Coalescent Chunking, the approach that turns this finite dependency into parallel execution. Each chunk is first evaluated from a provisional boundary state. Once its possible executions have merged, the remaining time steps produce the same outputs for every boundary state. When the true incoming state becomes available, only the earlier part of the chunk is recomputed. During the backward pass, zero local Jacobians divide the gradient recurrence into independent intervals that admit associative composition. The resulting forward states, spikes, and gradients agree with stepwise ReLIF execution. Figure 1 summarizes the execution pattern. Figure 1(a) shows that each chunk computes a provisional trajectory and identifies the earliest coalescence point, after which all reachable states share the same effective state and only the unresolved prefix requires exact replay. Figure 1(b) shows how zero-Jacobian anchors split the reverse recurrence into independent prefix intervals, leaving only the boundary-dependent suffixes to propagate gradients across chunks.

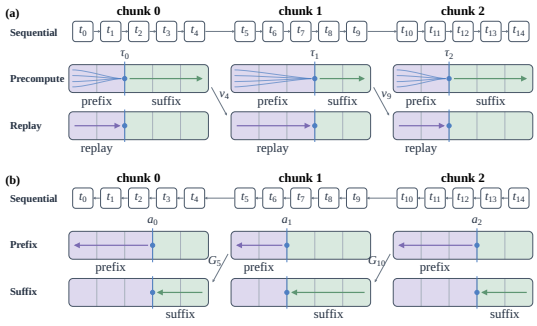

Figure 1: Coalescent Chunking: (a) forward pass and (b) backward pass.

Controlled replacements in SEW-ResNet and Spikformer retain LIF-level accuracy on CIFAR-10, CIFAR-100, and CIFAR10-DVS. On WikiText-103, 57M-, 102M-, and 191M-parameter SpikeTransformer++ models achieve lower training loss and validation perplexity under the same token budget and 4096-token context. Currents from these trained networks exhibit rapid coalescence and long boundary-independent suffixes. Coalescent Chunking exploits this structure to accelerate forward and backward execution, yielding higher full-model inference and training throughput than both serial ReLIF and SpikingJelly CuPy LIF.

Our contributions are threefold:

- We introduce ReLIF and finite-time state coalescence. The bounded effective state and two exact zero-producing transitions yield a finite expected coalescence time and a direct measure of temporal dependency.
- We develop exact forward and backward chunked execution. Boundary-independent suffixes expose parallel forward work, while zero-Jacobian anchors expose independent reverse intervals. Custom CUDA kernels realize both forms of parallelism.
- We establish a unified empirical account spanning task quality, finite-time coalescence in trained networks, and the resulting operator and end-to-end acceleration. Our results show that ReLIF preserves or improves task performance, that finite-time coalescence is present in currents generated by trained networks, and that this shortened dependence translates into measurable forward, backward, and end-to-end acceleration.

2 Related Work

Spiking neurons and scalable backbones. Recent neuron models improve temporal representation while retaining the familiar stepwise SNN interface. Direct training of SNNs commonly relies on surrogate-gradient methods to handle the non-differentiable spike function (Zenke & Ganguli, 2018; Neftci et al., 2019; Wu et al., 2018; Shrestha & Orchard, 2018; Eshraghian et al., 2023). PLIF learns the membrane time constant; GLIF combines integration, leakage, thresholding, and reset through learned gates; KLIF transforms the membrane state to coordinate activation with surrogate-gradient training; and CLIF adds a complementary state path for temporal credit assignment (Fang et al., 2021b; Yao et al., 2022; Jiang & Zhang, 2023; Huang et al., 2024a). Longer-timescale state retention

in recurrent SNNs has also been studied using adaptive neuronal dynamics (Bellec et al., 2018). Complementary work has improved the optimization and scaling of directly trained deep SNNs through normalization and architecture design (Zheng et al., 2021; Deng et al., 2022). In parallel, SNN backbones have expanded from SEW-ResNet to spike-based Transformers, state-space models, and autoregressive language models (Fang et al., 2021a; Zhou et al., 2023; Yao et al., 2023; 2024; Shen et al., 2024; Bal & Sengupta, 2025; Zhu et al., 2023; Bal & Sengupta, 2024). We evaluate neuron-level compatibility in convolutional and Transformer vision backbones, including event-stream classification (Yao et al., 2021), then use SpikeTransformer++ to examine long-sequence training and inference. The rectified recurrent state bounds the effective boundary-state domain in each setting, linking neuron design to temporal execution.

Temporal parallelism in SNNs. Temporal-parallel methods expose different structures in the neuronal recurrence. PSN removes reset recurrence and evaluates a learned temporal map, with masked and sliding variants for causal and variable-length deployment (Fang et al., 2023b). PMSN uses multi-compartment linear dynamics and separates parallel compartment evolution from spike-dependent recurrence (Chen et al., 2024). PSU decouples leaky integration from firing and handles reset through parallel units, allowing membrane values across time to be generated concurrently (Li et al., 2024b). PRF combines resonate-and-fire dynamics with a differentiable decoupled reset and frequency-domain computation for long sequences (Huang et al., 2024b). Dynamic-decay neurons express integration as causal convolution (Huang et al., 2026). FPT keeps the LIF recurrence and updates the trajectory through fixed-point iterations (Feng et al., 2025), while GPU-oriented work parallelizes specific propagation patterns at the operator level (Li et al., 2024a). ReLIF retains hard reset and exact stepwise dynamics, and finite-time coalescence identifies the chunk suffix whose states and spikes have become independent of the incoming boundary.

3 ReLIF and Finite-Time Coalescence

3.1 Recurrent State Rectification

The standard hard-reset LIF neuron follows

$$h_t = \gamma v_{t-1} + \eta x_t, \qquad s_t = \mathbf{1}\{h_t \geq \theta\}, \qquad v_t = (1 - s_t)h_t. \tag{3}$$

The reset bounds the state from above, while negative membrane values can be carried into the next block. ReLIF applies rectification to this recurrent state:

$$h_t = \gamma\,\mathrm{ReLU}(v_{t-1}) + \eta x_t, \qquad s_t = \mathbf{1}\{h_t \geq \theta\}, \qquad v_t = (1 - s_t)h_t. \tag{4}$$

The two updates coincide whenever $v_{t-1} \geq 0$. ReLIF retains the signed input current, leakage, threshold, and hard reset; rectification acts on the state carried from the preceding step. A negative current can therefore affect the present output, while a negative resulting membrane contributes zero to the next integration step. This local change gives the recurrent computation a lower bound without adding a trainable parameter.

For $z_t = \mathrm{ReLU}(v_t)$, the one-step map is

$$F_x(z) = \begin{cases} \gamma z + \eta x, & 0 < \gamma z + \eta x < \theta, \\ 0, & \gamma z + \eta x \leq 0 \text{ or } \gamma z + \eta x \geq \theta. \end{cases} \tag{5}$$

Every effective state lies in $[0, \theta)$. This interval supplies a common entry domain for all chunks. In contrast, the natural LIF domain $(-\infty, \theta)$ retains an unbounded negative tail after any finite input block, so it cannot provide the same full-domain coalescence guarantee. Figure 2(a) and Figure 2(b) compare the computational graphs of ReLIF and hard-reset LIF. ReLIF modifies only the recurrent state pathway by rectifying the carried membrane before the next integration step, while preserving leakage, thresholding, and reset dynamics. The domain argument is given in Appendix A.4.

3.2 Finite-Time Coalescence

For a shared input sequence, define

$$\mathcal{A}_t = \{z_t(z_0) : z_0 \in [0, \theta)\}, \qquad \mathcal{A}_t = \mathcal{F}_{x_t}(\mathcal{A}_{t-1}), \tag{6}$$

where $\mathcal{F}_x(A) = \{F_x(z) : z \in A\}$. The first coalescence time is

$$T = \inf\{t \geq 1 : \mathcal{A}_t \text{ is a singleton}\}. \tag{7}$$

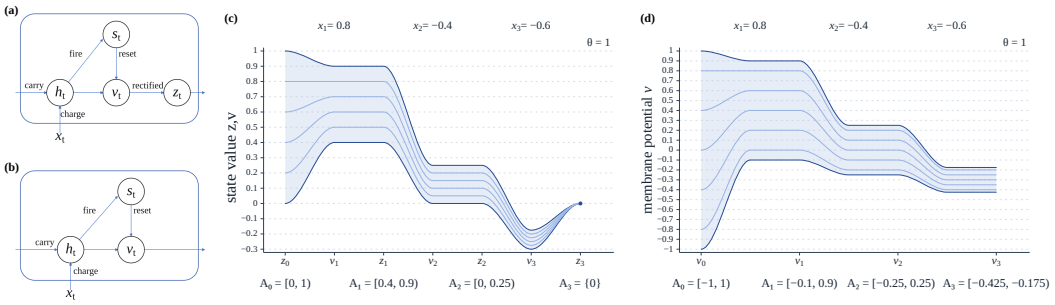

Figure 2: Comparison of ReLIF and LIF recurrence and reachable-state evolution. Panels (a–b) show their recurrent structures, and panels (c–d) show that under the same input sequence, ReLIF coalesces to zero while LIF remains dependent on the initial state.

Lemma 1 (First coalescence occurs at zero)**.** *If $A \subset [0, \theta)$ contains at least two states and $\mathcal{F}_x(A)$ is a singleton, then $\mathcal{F}_x(A) = \{0\}$.*

The active branch of eq. (5) is strictly injective, so a shared positive image can arise only after the input set has already collapsed. Hence

$$T = \inf\{t \geq 1 : \mathcal{A}_t = \{0\}\}. \tag{8}$$

Let $m(A) = \inf A$ and $M(A) = \sup A$. The preactivation set $\gamma A + \eta x$ has diameter at most $\gamma\theta < \theta$, which separates the two zero-producing branches of eq. (5).

Theorem 2 (One-step coalescence)**.** *For every nonempty $A \subset [0, \theta)$,*

$$\mathcal{F}_x(A) = \{0\} \quad \Longleftrightarrow \quad x \leq -(\tau_m - 1)M(A) \text{ or } x \geq \tau_m\theta - (\tau_m - 1)m(A). \tag{9}$$

The first event is ReLU coalescence and the second is reset coalescence.

The negative event sends the complete set through rectification, while the positive event sends it through hard reset. Figure 2(c) visualizes the reachable-state evolution of ReLIF under a shared input sequence. Starting from different initial states, the reachable set progressively contracts and eventually collapses to zero, showing that ReLIF removes the dependence on the incoming state after finite time. Figure 2(d) provides the corresponding hard-reset LIF comparison under the same input sequence. Unlike ReLIF, LIF trajectories originating from different initial states remain separated and preserve information about the initial condition, preventing finite-time state coalescence.

Corollary 3 (Finite expected coalescence)**.** *If the inputs are independent and identically distributed with*

$$p_0 = \mathbb{P}(x_t \leq -(\tau_m - 1)\theta) + \mathbb{P}(x_t \geq \tau_m\theta) > 0, \tag{10}$$

then $\mathbb{P}(T > n) \leq (1 - p_0)^n$ and

$$\mathbb{E}[T] \leq \frac{1}{p_0}. \tag{11}$$

The tail bound gives a finite mean whenever the input distribution assigns positive probability to either full-domain reset event. Reachable-set contraction also permits coalescence under less extreme inputs, as illustrated in Figure 2. Under $x_t \sim \mathcal{N}(0, 1)$, $\tau_m = 2$, and $\theta = 1$, numerical evaluation of the exact set recursion gives $\mathbb{E}[T] \approx 3.2918$. The full derivation and numerical procedure appear in Appendix A.3 and Appendix A.5. The real-current experiment in section 5.3 measures the same property under inputs generated by a trained network.

4 Parallel Forward and Backward Execution

Coalescent Chunking partitions $x_{1:N}$ into $K = \lceil N/L \rceil$ temporal chunks of length at most L. Before the exact boundary values are available, every chunk constructs a local summary in parallel. In the forward pass, the summary certifies when the remaining suffix no longer depends on the incoming

membrane state. In the backward pass, a local affine scan identifies zero-Jacobian anchors and the intervals that no longer depend on the following chunk. The remaining boundary quantities are carried only across the unresolved intervals, after which local computation proceeds concurrently in both directions.

4.1 Forward Coalescent Chunking

Chunk k covers positions $\ell_k = (k-1)L+1$ through $r_k = \min(kL, N)$. Its reachable set starts from the complete ReLIF state domain and evolves under the local inputs:

$$\mathcal{A}_0^{(k)} = [0, \theta), \qquad \mathcal{A}_j^{(k)} = \mathcal{F}_{x_{\ell_k+j-1}}\big(\mathcal{A}_{j-1}^{(k)}\big), \qquad \tau_k = \min\{j : \mathcal{A}_j^{(k)} = \{0\}\}. \tag{12}$$

A chunk without coalescence sets $\tau_k = r_k - \ell_k + 2$. Alongside the set propagation, a nominal trajectory evaluates the local recurrence. At $j = \tau_k$, all legal trajectories share the effective state $z = 0$. Every later preactivation, membrane value, and spike is therefore independent of the entry state. Certification is performed separately for each sample–neuron trajectory, allowing early-coalescing neurons to retain longer suffixes.

Chunk precomputation is independent across the temporal axis. A reconciliation pass follows the unresolved boundaries and evaluates each prefix from its true incoming state. To recover the complete output tensors, this prefix includes the coalescing step: its preactivation and signed membrane can still depend on the entry state even when its effective state is zero. The suffix after τ_k is retained. A chunk without a certificate is propagated in full. This boundary convention preserves the quantities required by both the forward output and surrogate backpropagation.

For short sequences, the fixed cost of multiple kernel launches is significant relative to the temporal workload. We therefore use a single-kernel anchor-segmented implementation. Because the effective entry state lies in $[0, \theta)$, an input satisfying $x_t \le -(\tau_m - 1)\theta$ maps the entire reachable set through rectification to zero, while $x_t \ge \tau_m\theta$ maps it to zero through hard reset. Equivalently, $\mathcal{F}_{x_t}([0,\theta)) = \{0\}$ at either event. This equality removes the dependence of all subsequent effective states on the unresolved incoming boundary and partitions the trajectory into independent segments. With $\tau_m = 2$, the two anchor conditions reduce to $x_t \le -\theta$ and $x_t \ge 2\theta$.

4.2 Backward Segmentation

Let G_t denote the total gradient with respect to the preactivation h_t, including the contribution from later time steps. With direct spike and membrane gradients a_t and b_t, surrogate derivative ϕ_t, and reset derivative ρ_t, the reverse recurrence is

$$G_t = p_t G_{t+1} + q_t, \qquad p_t = \gamma\rho_t \mathbf{1}\{v_t > 0\}, \qquad q_t = a_t\phi_t + \rho_t b_t. \tag{13}$$

Here $\rho_t = 1 - s_t - h_t\phi_t$ for differentiable reset and $\rho_t = 1 - s_t$ for detached reset. The corresponding input gradient is $\partial\mathcal{L}/\partial x_t = \eta G_t$, with terminal condition $G_{N+1} = 0$. Each token defines an affine map $g \mapsto p_t g + q_t$, and adjacent maps can be composed associatively (Martin & Cundy, 2018; Smith et al., 2023):

$$(A_1, B_1) \circ (A_2, B_2) = \big(A_1A_2,\ A_1B_2 + B_1\big). \tag{14}$$

For ReLIF, $p_t = 0$ whenever the recurrent state is rectified to zero or hard reset produces zero. At such a token, $G_t = q_t$ no longer depends on the right-boundary carrier. We call these tokens backward anchors. The anchors divide the reverse recurrence into independent local intervals while preserving the exact surrogate gradient. Within each chunk, a parallel reverse scan locates the rightmost anchor, evaluates the anchor and its left prefix immediately, and records the resulting left-boundary gradient. A second pass carries the exact boundary gradient only through the right-dependent suffix and through the uncommon run of chunks without an anchor.

4.3 Correctness and Computational Cost

Forward reconciliation reproduces the stepwise trajectory through the coalescing step. Both executions then share the effective state and all subsequent inputs, which establishes agreement throughout the retained suffix. Backward chunking composes the local maps in eq. (13); a zero coefficient removes the right-boundary contribution from the corresponding prefix. This gives the same surrogate-gradient recurrence.

Sequential execution has temporal depth N. Let Δ_n denote the longest interval without a backward anchor for neuron n. Coalescent precomputation has local depth at most L, while a warp scan composes the backward maps in $\lceil \log_2 L \rceil$ steps and the remaining carrier traverses at most Δ_n tokens. The resulting bounds are

$$D_{\text{fwd}} \le L + \sum_{k=1}^{K} \min(\tau_k, L), \qquad D_{\text{bwd}} \le \lceil \log_2 L \rceil + \max_n \Delta_n. \tag{15}$$

Here the forward expression counts reconciliation through the coalescing step. Its cost is governed by the short observed prefixes. Backward anchors bound the remaining carrier interval and supply independent chunk work. These depth bounds describe temporal dependence; actual runtime also includes certificate evaluation, tensor traffic, and finite GPU resources.

The CUDA implementation assigns chunk–neuron tiles to thread blocks and combines precision-aware hull propagation with nominal trajectory evaluation. The hull contains every reachable state in the chosen dtype and certifies a reusable suffix when all of it maps to zero. Backward uses warp shuffles for map composition and ballots for anchor locations. Implementation and certification details appear in Appendix B.1–B.3.

5 Experiments

The evaluation connects model behavior to execution through controlled LIF-to-ReLIF replacements, coalescence under trained currents, CUDA operator measurements, and end-to-end inference and training throughput.

5.1 Experimental Setup

The evaluation holds the surrounding network fixed while changing its recurrent neuron. Vision experiments use paired SEW-ResNet and Spikformer runs on CIFAR-10, CIFAR-100, and CIFAR10-DVS (Fang et al., 2021a; Zhou et al., 2023; Krizhevsky, 2009; Li et al., 2017). Each pair shares its architecture, initialization, optimizer, learning-rate schedule, data processing, surrogate derivative, and membrane time constant $\tau_m = 2$. Static models run for four steps and event models for 16. We report the mean and sample standard deviation across seeds 42, 123, and 2026 of each run's best Top-1 accuracy over its complete schedule.

SpikeTransformer++ extends the comparison to language modeling with 57M-, 102M-, and 191M-parameter models and a context length of 4096. Except for the final language modeling head, each spiking projection adopts a unified RMSNorm $\rightarrow$ LIF/ReLIF $\rightarrow$ Linear structure, with the original external normalization absorbed into the projection module to avoid redundant normalization. Every LIF/ReLIF pair is trained with seeds 42, 123, and 2026; paired runs begin from the same parameters and process the same 1.18 billion WikiText-103 (Merity et al., 2017) tokens over ten epochs. Kernel measurements use FP32 tensors, hidden size 1024, and chunk size 32. The matched serial CUDA ReLIF implementation isolates temporal scheduling, while SpikingJelly CuPy LIF (Fang et al., 2023a) represents the standard execution stack. All experiments were conducted on NVIDIA RTX 5090D GPUs. Full experimental protocols appear in Appendix C.

5.2 Model Quality

The vision study spans two architectures and two input modalities. SEW-ResNet contributes a convolutional backbone, while Spikformer exercises spike-based attention. CIFAR10-DVS adds a genuine temporal event stream to the static-image tasks (Li et al., 2017; Yao et al., 2021). The detailed vision model structures are given in Appendix C.1. Table 1 abbreviates the datasets as C10, C100, and C10-DVS.

The language experiment places the same neuron replacement inside a longer dependency horizon. All paired SpikeTransformer++ runs use 4096-token sequences throughout training, so the recurrent state evolves across the regime targeted by Coalescent Chunking.

Across the six controlled replacements, the mean Top-1 difference averages $+0.24$ percentage points. Accuracy remains comparable on the static datasets, and both backbones improve on CIFAR10-DVS, including a 1.13-point gain for SEW-ResNet.

Table 1: Vision Top-1 accuracy (%, three-seed mean ± s.d.).

Backbone	Dataset	T	LIF	ReLIF
SEW-ResNet	C10	4	**95.04**±0.18	95.02±0.04
	C100	4	**77.95**±0.28	**77.95**±0.21
	C10-DVS	16	79.00±0.56	**80.13**±0.38
Spikformer	C10	4	**94.83**±0.25	94.77±0.15
	C100	4	78.43±0.27	**78.58**±0.21
	C10-DVS	16	79.93±0.15	**80.17**±0.64

Table 2: SpikeTransformer++ on WikiText-103 at context 4096 (three-seed mean ± s.d.).

Model	Neuron	Train loss ↓	Val. PPL ↓
57M	LIF	4.150±0.020	62.74±0.89
	ReLIF	**4.112±0.008**	**60.55±0.54**
102M	LIF	3.992±0.009	54.02±0.27
	ReLIF	**3.890±0.004**	**49.29±0.08**
191M	LIF	3.779±0.040	45.89±1.59
	ReLIF	**3.658±0.013**	**40.81±0.29**

ReLIF achieves lower final-window training loss and validation perplexity for every seed at all three scales. Table 2 shows that the improvement grows with model size under the same token budget and 4096-token context, reaching a reduction in mean validation perplexity from 45.89 to 40.81 at 191M parameters. These results support the neuron replacement across convolutional and attention-based models and both input modalities.

5.3 Coalescence in Trained Networks

The theoretical result begins each chunk from the complete state set $[0, \theta)$. To determine how quickly this uncertainty disappears after training, we record currents from the 57M, 102M, and 191M ReLIF SpikeTransformer++ models. For each seed, we use eight complete WikiText-103 validation sequences, sample 16 channels at every spiking site, and partition the 4096-token currents into chunks of length 32. We propagate the full state set for every chunk–neuron observation and measure τ together with the deterministic suffix ratio

$$r_{\text{suffix}} = \mathbb{E}\left[\frac{(L-\tau+1)_+}{L}\right]. \tag{16}$$

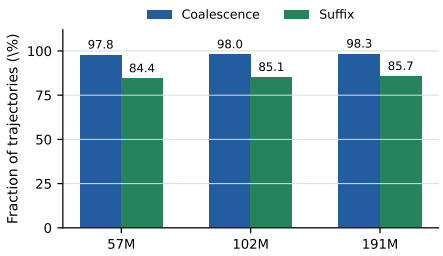

Figure 3: Coalescence and deterministic-suffix ratios across scales.

The within-chunk coalescence rate is 97.80%, 97.98%, and 98.33% for the 57M, 102M, and 191M models. Their conditional mean coalescence times are 5.38, 5.19, and 5.12 steps, while the deterministic suffix occupies 84.41%, 85.13%, and 85.67% of each chunk. All three survival curves fall below 10% by step 12. Figure 3 shows that most boundary information disappears near the beginning of each chunk, leaving a stable supply of independent work across model scales.

The effect appears throughout the attention and feed-forward projection paths and strengthens slightly from 57M to 191M parameters. The Gaussian analysis predicts the mechanism from the neuron dynamics; these measurements establish its presence under structured currents generated by trained networks.

5.4 Comparison with Temporal-Parallel Neurons

On identical FP32 Gaussian currents of shape $[T, 1, 8192]$, we compare parallel ReLIF with released FPT (Feng et al., 2025), sliding PSN (Fang et al., 2023b), and PRF (Huang et al., 2024b) implementations. Inference times a no-grad forward; training times a fresh forward and input- and parameter-gradient backward together, excluding the optimizer. Each CUDA-event result is the median of 20 calls after 10 warm-ups.

At $T = 4096$, Figure 4 shows ReLIF forward gains of $16.23\times$, $1.61\times$, and $10.09\times$ over FPT, sliding PSN, and PRF. Training forward-and-backward gains reach $29.09\times$, $7.41\times$, and $14.34\times$, with sliding PSN's larger training gap arising from gradient computation.

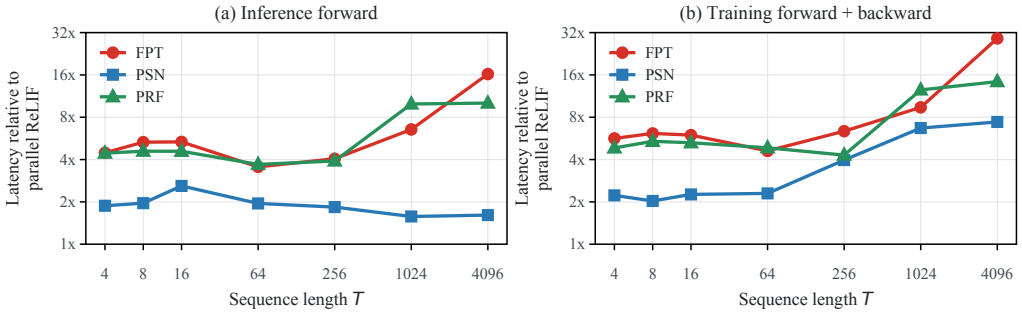

Figure 4: Latency relative to parallel ReLIF: (a) inference forward and (b) neuron-level training forward and backward. Values above one favor ReLIF; absolute times appear in table 6.

5.5 Kernel-Level Acceleration

The five current distributions vary rectification, reset, and tail behavior: negative shifts favor rectification, positive shifts alter the reset balance, a narrow Gaussian reduces extremes, and Laplace inputs supply heavier tails. Serial and parallel ReLIF receive identical inputs. The serial baseline uses a fused CUDA recurrence; parallel timing includes interval discovery and reconciliation.

Table 3: Finite dependence and kernel-level speedups at $T = 4096$. Each speedup is the baseline latency divided by the Coalescent Chunking ReLIF latency. The first three speedup columns use matched serial CUDA ReLIF as the baseline; the final three use SpikingJelly CuPy LIF.

Input current	Mean τ	Suffix (%)	vs. serial ReLIF			vs. CuPy LIF		
			Fwd.	Bwd.	F+B	Fwd.	Bwd.	F+B
$\mathcal{N}(0, 1)$	3.30	92.81	6.70×	17.81×	12.09×	32.78×	16.37×	24.82×
$\mathcal{N}(0, 0.5^2)$	4.72	88.36	5.41×	16.33×	10.35×	26.37×	14.91×	21.18×
$\mathcal{N}(-0.5, 1)$	2.27	96.02	8.04×	22.07×	15.17×	38.07×	17.71×	27.72×
$\mathcal{N}(0.5, 1)$	4.26	89.82	5.85×	17.39×	11.14×	27.89×	15.76×	22.33×
Laplace, unit variance	3.58	91.92	6.43×	18.67×	12.22×	31.04×	16.51×	24.17×

Table 3 shows the earliest coalescence, longest suffix, and highest forward speedup under the negatively shifted Gaussian, whereas the narrow Gaussian retains more boundary-dependent work. Across the five distributions at $T = 4096$, suffix ratio and forward speedup have a Pearson correlation of $r = 0.994$, linking coalescence to runtime. Backward anchors bound the independent intervals to a p95 length of 9.0–20.85 steps. At $T = 4096$, parallel ReLIF achieves $5.41\times$–$8.04\times$ forward and $16.33\times$–$22.07\times$ backward speedup over matched serial CUDA ReLIF; relative to SpikingJelly CuPy LIF, the corresponding ranges are $26.37\times$–$38.07\times$ and $14.91\times$–$17.71\times$ across the five distributions. Figure 5 shows that forward speedup grows with sequence length as more independent chunks become available, while each local recurrence remains bounded by $L = 32$. Appendix D.2 reports zero spike mismatches against the Torch reference and maximum state or gradient error below 2.4×10^{-7} for parallel CUDA ReLIF.

NCU measurements show distinct hardware regimes. At $T \leq 256$, DRAM throughput is at most 1.24% of peak, with launch and local scan costs dominating. At $T = 4096$, backward sustains 1.09–1.20 TB/s, or 61.8%–68.0% of peak, making tensor traffic the principal cost. Full measurements appear in Appendix D.3.

5.6 End-to-End Throughput

We measure full-validation inference and complete training at all three model scales using seed 42. The serial and parallel ReLIF paths share the same neuron recurrence, CUDA extension, checkpoints, and tensor interface; their temporal schedules are the only execution difference. Inference runs one timed pass over the WikiText-103 validation split after warm-up. The LIF path uses the official SpikingJelly CuPy multi-step kernel, whose output matches the stepwise reference exactly in a

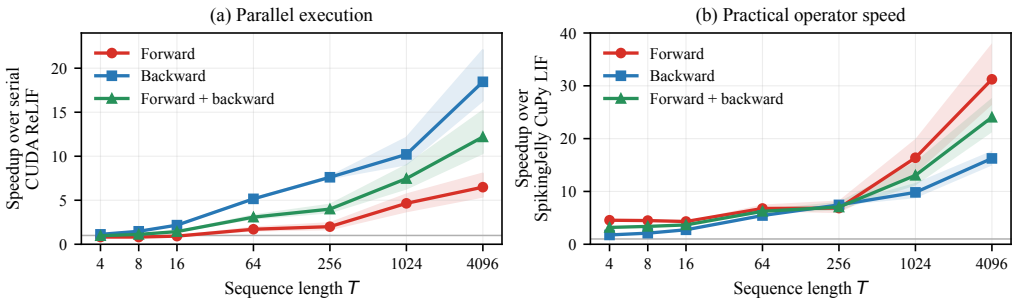

Figure 5: NCU kernel acceleration across sequence lengths. (a) Forward, backward, and combined speedup over matched serial ReLIF. (b) Operator speedup over SpikingJelly CuPy LIF. Shading gives the range across five current distributions.

pre-run probe. Training throughput averages the final 100 logged intervals and covers attention, feed-forward computation, loss evaluation, backward propagation, gradient clipping, and parameter updates.

Table 4: End-to-end SpikeTransformer++ throughput using seed 42. Values are thousands of tokens per second.

Model	Workload	CuPy LIF	Serial ReLIF	Parallel ReLIF	P/S	P/CuPy
57M	Inference	96.64	272.00	289.88	1.07×	3.00×
	Training	27.71	44.27	53.98	1.22×	1.95×
102M	Inference	62.00	152.72	198.11	1.30×	3.20×
	Training	17.83	26.57	36.16	1.36×	2.03×
191M	Inference	56.51	127.03	165.22	1.30×	2.92×
	Training	16.09	22.90	34.37	1.50×	2.14×

Table 4 shows that parallel ReLIF reaches $1.07\times$–$1.30\times$ the inference throughput and $1.22\times$–$1.50\times$ the training throughput of matched serial CUDA ReLIF. Against SpikingJelly CuPy LIF, the corresponding gains are $2.92\times$–$3.20\times$ and $1.95\times$–$2.14\times$. The matched comparison isolates temporal scheduling inside the complete model, while the CuPy comparison measures the gain of deploying ReLIF and its backend together. The gains persist from 57M to 191M parameters across both forward-only evaluation and complete optimization.

6 Conclusion

ReLIF bounds the recurrent state and enables finite-time coalescence while retaining hard reset. Coalescent Chunking uses boundary-independent suffixes and zero-Jacobian intervals for exact parallel forward and backward execution. Experiments connect this mechanism to preserved vision accuracy, improved language modeling, coalescence in trained networks, and acceleration at both operator and model levels. In kernel-level measurements at $T = 4096$, parallel ReLIF reaches up to $38.07\times$ forward and $17.71\times$ backward speedup over SpikingJelly CuPy LIF, and $8.04\times$ forward and $22.07\times$ backward speedup over matched serial CUDA ReLIF. Across three SpikeTransformer++ scales, the complete stack reaches $2.92\times$–$3.20\times$ inference and $1.95\times$–$2.14\times$ training throughput over paired SpikingJelly CuPy LIF, together with $1.07\times$–$1.30\times$ inference and $1.22\times$–$1.50\times$ training throughput over matched serial CUDA ReLIF. These results establish finite temporal dependence as a practical basis for parallel SNN execution.

Reproducibility Statement

The neuron equations, reachable-set recursion, parallel forward and backward algorithms, and complexity bounds are specified in the main text. The appendix provides complete proofs, numerical

procedures, CUDA organization and kernel pseudocode, training configurations, correctness criteria, and full operator results.

AI USE STATEMENT

Generative AI tools were used on a limited basis to assist with mathematical derivations, code debugging, experiment scripting, figure preparation, and copyediting. The authors independently developed the core methods, conducted the experiments, and verified all mathematical arguments, source code, numerical results, and final text. The authors take full responsibility for the complete submission.

REFERENCES

Malyaban Bal and Abhronil Sengupta. SpikingBERT: Distilling BERT to train spiking language models using implicit differentiation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10998–11006, 2024. doi: 10.1609/aaai.v38i10.28975.

Malyaban Bal and Abhronil Sengupta. P-SpikeSSM: Harnessing probabilistic spiking state space models for long-range dependency tasks. In *International Conference on Learning Representations*, 2025.

Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

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:2408.14917v1*, 2024.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2022.

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. *Proceedings of the IEEE*, 111(9):1016–1054, 2023. doi: 10.1109/JPROC.2023.3308088.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 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. doi: 10.1126/sciadv.adi1480.

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 *Advances in Neural Information Processing Systems*, volume 36, 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 *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 16711–16726. PMLR, 2025.

Yanbin Huang, Man Yao, Yuqi Pan, Changze Lv, Siyuan Xu, Xiaoqing Zheng, Bo Xu, and Guoqi Li. Parallel training in spiking neural networks. *arXiv preprint arXiv:2602.01133*, 2026.

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 *Proceedings of the 41st International Conference on Machine Learning*, 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.

Chunming Jiang and Yilei Zhang. KLIF: An optimized spiking neuron unit for tuning surrogate gradient slope and membrane potential. *arXiv preprint arXiv:2302.09238*, 2023.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. CIFAR10-DVS: An event-stream dataset for object classification. *Frontiers in Neuroscience*, 11:309, 2017. doi: 10.3389/fnins.2017.00309.

Yanchen Li, Jiachun Li, Kebin Sun, Luziwei Leng, and Ran Cheng. Towards scalable GPU-accelerated SNN training via temporal fusion. *arXiv preprint arXiv:2408.00280*, 2024a.

Yang Li, Yinqian Sun, Xiang He, Yiting Dong, Dongcheng Zhao, and Yi Zeng. Parallel spiking unit for efficient training of spiking neural networks. *arXiv preprint arXiv:2402.00449*, 2024b.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

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. doi: 10.1109/MSP.2019.2931595.

Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575:607–617, 2019. doi: 10.1038/s41586-019-1677-2.

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. *arXiv preprint arXiv:2408.14909*, 2024.

Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

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. doi: 10.3389/fnins.2018.00331.

Man Yao, Huanhuan Gao, Guangshe Zhao, Dingheng Wang, Yihan Lin, Zhaoxu Yang, and Guoqi Li. Temporal-wise attention spiking neural networks for event streams classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10221–10230, 2021. doi: 10.1109/ICCV48922.2021.01006.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Man Yao, Jiakui Hu, Tianxiang Hu, Yifan Xu, Zhaokun Zhou, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer V2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips. In *International Conference on Learning Representations*, 2024.

Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. GLIF: A unified gated leaky integrate-and-fire neuron for spiking neural networks. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Computation*, 30(6):1514–1541, 2018. doi: 10.1162/neco_a_01086.

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. doi: 10.1609/aaai.v35i12.17320.

Zhaokun Zhou, Yuesheng Zhu, Chao He, Yaowei Wang, Shuicheng Yan, Yonghong Tian, and Li Yuan. Spikformer: When spiking neural network meets transformer. In *International Conference on Learning Representations*, 2023.

Rui-Jie Zhu, Qihang Zhao, Guoqi Li, and Jason K. Eshraghian. SpikeGPT: Generative pre-trained language model with spiking neural networks. *arXiv preprint arXiv:2302.13939*, 2023.

A Theoretical Proofs and Numerical Evaluation

A.1 Proof of Lemma 1

Assume $\mathcal{F}_x(A) = \{c\}$ for a non-singleton A and some $c > 0$. Every state that maps to c lies in the active region $0 < \gamma z + \eta x < \theta$, where $F_x(z) = \gamma z + \eta x$. This affine map is strictly injective because $\gamma > 0$. Hence all states in A are equal, contradicting the premise. The singleton image is therefore $\{0\}$.

A.2 Proof of Theorem 2

Let $m = \inf A$ and $M = \sup A$. If $x \le -(\gamma/\eta)M$, then $\gamma z + \eta x \le 0$ for every $z \in A$, so all states map to zero through rectification. If $x \ge (\theta - \gamma m)/\eta$, then $\gamma z + \eta x \ge \theta$ for every $z \in A$, so all states map to zero through hard reset.

For the converse, suppose both inequalities fail. Then

$$\gamma M + \eta x > 0, \qquad \gamma m + \eta x < \theta. \tag{17}$$

If $\gamma A + \eta x$ contained no value in $(0, \theta)$, it would contain values in both $(-\infty, 0]$ and $[\theta, \infty)$. Its diameter would then be at least θ. Yet

$$\operatorname{diam}(\gamma A + \eta x) = \gamma \operatorname{diam}(A) \le \gamma\theta < \theta, \tag{18}$$

a contradiction. Some state remains in the active region and maps to a positive value. The two coalescence inequalities are disjoint because

$$-\frac{\gamma}{\eta} M < \frac{\theta - \gamma m}{\eta}. \tag{19}$$

A.3 Expected Coalescence under Gaussian Inputs

For $x_t \sim \mathcal{N}(0, 1)$, $\tau_m = 2$, and $\theta = 1$, the conditional one-step coalescence probability from a reachable set A is

$$p(A) = \Phi\big(-M(A)\big) + 1 - \Phi\big(2 - m(A)\big). \tag{20}$$

The full-state lower bound is

$$p(A) \ge \Phi(-1) + 1 - \Phi(2) = 0.1814053859. \tag{21}$$

The exact expectation satisfies the Bellman equation

$$V(A) = 1 + \int_{-M(A)}^{2-m(A)} V(\mathcal{F}_x(A))\phi(x)\,\mathrm{d}x, \tag{22}$$

with $V(\{0\}) = 0$. Exact finite-union set propagation and numerical integration give

$$V([0, 1)) = 3.29177\ldots. \tag{23}$$

The first-step probability is 0.1814053859. Exact finite-union propagation gives the remaining hitting-time distribution and the expectation above.

A.4 LIF Coalescence and Bellman Equation

For LIF, define

$$G_x(v) = \begin{cases} \gamma v + \eta x, & \gamma v + \eta x < \theta, \\ 0, & \gamma v + \eta x \ge \theta. \end{cases} \tag{24}$$

Natural state domain. Fix a finite input prefix $x_1, \ldots, x_t$. If no reset occurs, then

$$v_k = \gamma^k v_0 + \eta \sum_{j=1}^{k} \gamma^{k-j} x_j. \tag{25}$$

For every finite t, one may choose v_0 sufficiently negative so that $v_k < \theta$ for all $k \le t$ and $v_t \ne 0$. Thus the image of $(-\infty, \theta)$ is non-singleton at every finite time, and

$$T_{\mathrm{LIF}} = \infty. \tag{26}$$

Bounded state interval. For $v_0 \in (-1, 1)$, $\tau_m = 2$, $\theta = 1$, and standard Gaussian inputs, the reachable set at every finite time is the union of at most one interval and finitely many isolated points. Let $W(A)$ denote the expected remaining coalescence time from a reachable set A. The absorbed set-valued Markov chain gives

$$W(\{0\}) = 0, \qquad W(A) = 1 + \int_{-\infty}^{\infty} W(G_x(A))\phi(x)\,\mathrm{d}x. \tag{27}$$

The equation follows from the pathwise first-step decomposition and the independence of the shifted input sequence. Value iteration

$$W_0(A) = 0, \qquad W_{n+1}(A) = 1 + \int W_n(G_x(A))\phi(x)\,\mathrm{d}x \tag{28}$$

satisfies $W_n(A) = \mathbb{E}[T_A \wedge n]$ and increases to the minimal nonnegative solution $W(A)$. Exact set propagation and Monte Carlo integration yield

$$W((-1, 1)) = 34.39\ldots. \tag{29}$$

This value is computed under the same $\tau_m = 2$ decay-input dynamics as ReLIF.

A.5 Exact Reachable-Set Propagation

For ReLIF, the active preimage is

$$I_x = \left(-\frac{\eta}{\gamma}x, \frac{\theta - \eta x}{\gamma}\right). \tag{30}$$

For a finite union of intervals A, the exact update is

$$\mathcal{F}_x(A) = \eta x + \gamma(A \cap I_x) \cup \begin{cases} \{0\}, & A \setminus I_x \neq \varnothing, \\ \varnothing, & A \setminus I_x = \varnothing. \end{cases} \tag{31}$$

Each interval is intersected with I_x, mapped affinely, and merged with neighboring output intervals. States outside the active preimage contribute zero. Starting from one interval, a single update adds at most one connected component, so a chunk of length L requires at most $L + 1$ interval records; singleton states use equal endpoints. Endpoint openness may be retained explicitly. Under an independent continuous input distribution, closed-set propagation gives the same coalescence time almost surely because threshold-equality events form a countable union of null events.

For bounded-state LIF, the reachable set is represented as one interval plus a finite point set. The interval is clipped at the reset preimage $(\theta - \eta x)/\gamma$, the surviving part is mapped affinely, and every reset state contributes the point zero. Isolated points are updated individually, merged when equal, and removed when they fall inside the interval.

B Algorithm and CUDA Implementation Details

B.1 Forward Execution

The implementation receives the standard multi-step tensor layout, uses $\tau_m = 2$, and sets the chunk size to $L = 32$. For $T \leq 256$, a one-launch anchor-segmented kernel uses warp ballots to identify the direct anchors $x \leq -\theta$ and $x \geq 2\theta$, which are the $\tau_m = 2$ instances of $x \leq -(\tau_m - 1)\theta$ and $x \geq \tau_m\theta$. Segment leaders advance independent intervals from the true initial state or the shared zero state, while the exact terminal state is carried between consecutive 32-token tiles. The chunked path uses two kernels for longer sequences. Its precomputation kernel maps chunk–sample–neuron tiles to thread blocks, keeps the first four certificate steps in registers, compacts unresolved tasks into 32-lane tail waves, and processes at most 64 chunks per wave. Certificate propagation and deterministic-suffix evaluation share this kernel. All active lanes advance through the suffix at the same local time, giving contiguous accesses to the $[T, N]$ tensors. The segmented boundary kernel carries exact entry states across unresolved chunks and replays each prefix directly in the final tensors. Both execution paths support FP16, BF16, and FP32.

B.2 ROUND-SAFE CERTIFICATION

Let $D \in \{\mathrm{FP16}, \mathrm{BF16}, \mathrm{FP32}\}$ and let $\mathbb{D}$ denote its finite representable values. The operator Q_D rounds to D in round-to-nearest-even mode. The threshold $\theta_D > 0$ is representable in D, and θ_D^- denotes the largest value in $\mathbb{D}$ strictly below θ_D. For representable z and x, define the rounded preactivation

$$P_D(z, x) = Q_D(\mathrm{fma}(\gamma, z, \eta x)) \,. \tag{32}$$

The definition of P_D follows the operation order of the sequential CUDA kernel. Its finite-precision effective-state transition is

$$\widehat{F}_x^D(z) = \begin{cases} P_D(z, x), & 0 < P_D(z, x) < \theta_D, \\ 0, & P_D(z, x) \le 0 \text{ or } P_D(z, x) \ge \theta_D. \end{cases} \tag{33}$$

For $m, M \in \mathbb{D}$, write $[m, M]_D = \{z \in \mathbb{D} : m \le z \le M\}$. The certificate propagates this discrete hull with algorithm 1. Non-finite values invalidate the certificate and leave the complete chunk to reconciliation.

Algorithm 1 Round-safe reachable-hull update.

Require: Hull $H = [m, M]_D$, input x, threshold θ_D, predecessor θ_D^-
Ensure: Updated hull H' and certificate c

if x, m, or M is non-finite **then**
  return $(\bot, \mathrm{false})$
end if
$a \leftarrow P_D(m, x)$; $b \leftarrow P_D(M, x)$
if $b \le 0$ or $a \ge \theta_D$ **then**
  return $(\{0\}, \mathrm{true})$
end if
$m' \leftarrow a$; $M' \leftarrow b$
if $a \le 0$ or $b \ge \theta_D$ **then**
  $m' \leftarrow 0$
end if
if $b \ge \theta_D$ **then**
  $M' \leftarrow \theta_D^-$
end if
return $([m', M']_D, \mathrm{false})$

Proposition 4 (No false finite-precision certificate)**.** *Let $S \subseteq [m, M]_D$ be the effective states reachable at the entrance to one step, and let algorithm 1 return (H', c) on a finite input. If $c = \mathrm{false}$, then $\widehat{F}_x^D(S) \subseteq H'$. If $c = \mathrm{true}$, then $\widehat{F}_x^D(S) = \{0\}$.*

Proof. The affine preactivation has positive slope in z, and round-to-nearest-even is nondecreasing. Hence $P_D(z, x) \in [a, b]_D$ for every $z \in S$. If $b \le 0$, all rounded preactivations take the rectifying branch of eq. (33); if $a \ge \theta_D$, they all take the hard-reset branch. Both cases give the singleton image $\{0\}$.

In the remaining cases, rounded preactivations in $(0, \theta_D)$ pass through unchanged, while values at or below zero and values at or above θ_D contribute zero. When $[a, b]_D$ stays inside the active region, its image is contained in $[a, b]_D$. If it reaches either zero-producing region, the update includes zero by setting $m' = 0$. If it reaches the reset region, every surviving active value is at most θ_D^-, which gives the stated upper bound. Thus the returned hull contains the complete finite-precision image in every case. □

Starting from $[0, \theta_D^-]_D$, repeated application of the proposition preserves containment of the discrete reachable set. The hull may include states that are unreachable and can therefore delay certification; every reported certificate is sufficient for coalescence. The certified step itself is replayed from the true chunk entry because its preactivation, signed membrane, and spike can still depend on that entry. Reuse begins at the following step, where every trajectory has the common effective state zero and subsequently executes the same finite-precision recurrence.

B.3 Backward Execution

The custom autograd function saves the preactivation, membrane sequence, and initial state. For $T \leq 256$, a one-launch anchor-based reverse affine scan composes the temporal maps with warp shuffles and carries the exact boundary gradient between consecutive 32-token tiles. Zero-Jacobian anchors terminate the corresponding right-boundary carrier during the scan. Longer sequences use two kernels. The anchor-prefix kernel assigns one warp to each chunk–neuron pair, locates the rightmost zero-Jacobian anchor with a warp ballot, and emits the anchor together with its independent left prefix. Its shared-memory rows use a 33-element stride to avoid bank conflicts during transposed tensor access. The boundary-suffix kernel starts from the sequence end or immediately before a certified chunk, carries the exact boundary gradient through chunks without an anchor, and emits the remaining right-dependent suffix. Two FP32 arrays of shape $[\lceil T/L \rceil, N]$ store the anchor positions and left-boundary gradients. Both paths support ATan and sigmoid surrogate derivatives, with differentiable or detached hard reset.

B.4 Backward Recurrence

Let $a_t = \partial \mathcal{L}/\partial s_t$ and $b_t = \partial \mathcal{L}/\partial v_t$ denote the direct upstream gradients. For the ATan surrogate,

$$\phi_t = \frac{\alpha/2}{1 + [(\pi\alpha/2)(h_t - \theta)]^2}. \tag{34}$$

Differentiable reset uses $\rho_t = 1 - s_t - h_t\phi_t$, while detached reset uses $\rho_t = 1 - s_t$. The coefficients in eq. (13) are

$$p_t = \gamma\rho_t \mathbf{1}\{v_t > 0\}, \qquad q_t = a_t\phi_t + \rho_t b_t. \tag{35}$$

The decay-input gradients are

$$\frac{\partial \mathcal{L}}{\partial x_t} = \eta G_t, \qquad \frac{\partial \mathcal{L}}{\partial v_0} = \gamma \mathbf{1}\{v_0 > 0\} G_1. \tag{36}$$

Each token defines the map $g \mapsto p_t g + q_t$. Pair composition satisfies

$$(A_1, B_1) \circ (A_2, B_2) = \big(A_1 A_2,\ A_1 B_2 + B_1\big), \tag{37}$$

and is associative. Whenever $p_t = 0$, the left side of the recurrence no longer depends on gradients to the right of token t. Let Δ_n be the longest interval without such an anchor for neuron n. A warp composes one chunk in $\lceil \log_2 L \rceil$ steps, after which the remaining carrier path is bounded by Δ_n:

$$D_{\mathrm{bwd}} \leq \lceil \log_2 L \rceil + \max_n \Delta_n. \tag{38}$$

B.5 CUDA Kernel Pseudocode

Algorithms 2 and 3 summarize forward and backward CUDA execution. Here Step applies the rounded ReLIF update and writes the spike, signed membrane, and preactivation; HullUpdate is algorithm 1. Parallel loops assign independent neuron, chunk, or segment work to GPU lanes. Ordered loops carry the remaining boundary state or gradient. The short-sequence path uses 32-step tiles. Its negative-input anchor at the first sequence step is used only when $\mathrm{ReLU}(v_0) \leq \theta$. Non-finite inputs do not produce a forward certificate. The backward kernels compute each affine pair (p_t, q_t) when processing its token; no separate coefficient tensor is formed.

Algorithm 2 Parallel ReLIF forward CUDA execution.

Require: Currents $x_{1:T}$, initial membrane v_0, threshold θ; chunk length $L = 32$
Ensure: Spikes $s_{1:T}$, membranes $v_{1:T}$, preactivations $h_{1:T}$

if $T \le 256$ **then**
  for all neurons n in parallel **do**
    $u \leftarrow v_{0,n}$
    for 32-step tiles I in temporal order **do**
      $A \leftarrow$ warp ballot for valid $x_{t,n} \le -\theta$ or $x_{t,n} \ge 2\theta$ in I
      for all segments of I induced by A, in parallel **do**
        Start from u at the tile boundary, else zero; apply STEP and write (s, v, h)
      end for
      $u \leftarrow$ last membrane in I
    end for
  end for
else
  for waves W of at most 64 consecutive chunks **do**
    for all chunk–neuron pairs (C, n) in W, in parallel **do**
      $H \leftarrow [0, \theta_D^-]_D$; $\tau_{C,n} \leftarrow |C| + 1$
      Apply HULLUPDATE along C; set $\tau_{C,n}$ to the first certified step
      if $\tau_{C,n} \le |C|$ **then**
        Evaluate the suffix after $\tau_{C,n}$ from zero; store outputs and final state
      end if
    end for
    for all neurons n in parallel **do**
      $u \leftarrow$ initial membrane or saved final state from the preceding wave
      if u is non-finite or $\mathrm{ReLU}(u) \ge \theta$ **then**
        for chunks $C \in W$ in temporal order, in one thread **do**
          if $\tau_{C,n} \le |C|$ and u is finite and $\mathrm{ReLU}(u) < \theta$ **then**
            Replay through $\tau_{C,n}$; retain the suffix; $u \leftarrow$ stored final state
          else
            Replay all of C from u; $u \leftarrow$ its actual final membrane
          end if
        end for
      else
        for all segments starting at W or after a certified chunk, in parallel **do**
          Load the true segment entry; replay each uncertified chunk in full
          If one follows, replay the first certified chunk through $\tau_{C,n}$, inclusive
          Retain its suffix; the next segment loads its stored final state
          If none is certified, save the final membrane for the next wave
        end for
      end if
    end for
  end for
end if

Algorithm 3 Parallel ReLIF backward CUDA execution.

Require: Saved $h_{1:T}, v_{1:T}, v_0$ and direct gradients $a_{1:T}, b_{1:T}$; $L = 32$
Ensure: Input gradients $\nabla_x \mathcal{L}$ and initial-state gradient $\nabla_{v_0} \mathcal{L}$

if $T \leq 256$ **then**
  for all neurons n in parallel **do**
    $c \leftarrow 0$
    for 32-step tiles I in reverse temporal order **do**
      Warp-scan (p_t, q_t); use carrier c to write $\nabla_{x_t} \mathcal{L} = \eta G_t$
      $c \leftarrow G_{\min(I)}$
    end for
    $\nabla_{v_{0,n}} \mathcal{L} \leftarrow \gamma \mathbf{1}\{v_{0,n} > 0\} c$
  end for
else
  for all chunk–neuron pairs (C, n) in parallel **do**
    $j_{C,n} \leftarrow$ rightmost $t \in C$ with finite $h_{t,n}, v_{t,n}$ and $v_{t,n} \leq 0$, or -1
    Reverse-scan with zero carrier; write gradients through $j_{C,n}$
    Store $j_{C,n}$ and, if anchored, the independent left-boundary gradient
    if C is the first chunk and $j_{C,n} \geq 0$ **then**
      $\nabla_{v_{0,n}} \mathcal{L} \leftarrow \gamma \mathbf{1}\{v_{0,n} > 0\}$ times the stored left-boundary gradient
    end if
  end for
  for all neuron segments ending at T or before an anchored chunk, in parallel **do**
    Initialize the carrier to zero or the stored gradient of the next chunk
    Replay $G_t = p_t G_{t+1} + q_t$ right to left until an anchor; write $\nabla_{x_t} \mathcal{L}$
    If $t = 1$ is reached without an anchor, write $\nabla_{v_{0,n}} \mathcal{L} = \gamma \mathbf{1}\{v_{0,n} > 0\} G_1$
  end for
end if

C Experimental Protocols

C.1 Controlled Replacement

Within each LIF/ReLIF pair, the architecture, parameter count, initialization, data pipeline, optimizer, learning-rate schedule, surrogate derivative, reset-gradient convention, membrane time constant $\tau_m = 2$, and model-selection rule are shared. The recurrent charge equation is the sole model change. LIF uses the SpikingJelly CuPy backend and ReLIF uses the parallel CUDA backend.

The static-image SEW-ResNet entries use the released four-stage `sew_resnet18` body with element-wise ADD residual connections. Its ImageNet stem is replaced by a CIFAR stem with a 3×3 stride-one convolution, and the initial max-pooling layer is removed. The CIFAR10-DVS entry uses the dataset-specific `ResNetN` construction from the released event model: four 64-channel SEW-ADD blocks followed by three 128-channel SEW-ADD blocks, with a 2×2 pooling operation after each block and a final linear classifier. Each SEW block contains two convolution–BatchNorm–neuron paths joined by the ADD connection. The paired Spikformer models use the released Spikformer-4-384 configuration for static CIFAR and Spikformer-2-256 for CIFAR10-DVS. The surrounding backbone remains fixed within each pair; only the recurrent neuron changes from LIF to ReLIF.

C.2 Long-Sequence Language Modeling

The 57M SpikeTransformer++ uses eight layers, width 384, six attention heads, and a feed-forward width of 1536. The 102M model uses 12 layers, width 512, eight heads, and a feed-forward width of 2048. The 191M model uses 12 layers, width 768, 12 heads, and a feed-forward width of 3072. All three models use a GPT-2 vocabulary of 50,257 tokens. Each SpikeTransformer block contains four spiking sites, including a shared input for the Q/K/V projections, the attention output projection, a shared input for the gate/up projections, and the FFN hidden projection. Except for the final language modeling head, all spiking projections follow a unified RMSNorm $\rightarrow$ LIF/ReLIF $\rightarrow$ Linear structure, with the original external normalization absorbed into the projection module so that normalization is

Table 5: Training configurations for the controlled vision replacements. All runs use seeds 42, 123, and 2026. The CIFAR10-DVS logs include ten cooldown evaluations after the 96 scheduled epochs.

Backbone	Dataset	Optimizer	Epochs	LR schedule	Batch size
SEW-ResNet18	CIFAR-10	AdamW	310	warmup + cosine	128
SEW-ResNet18	CIFAR-100	AdamW	410	warmup + cosine	64
SEW-ResNetN	CIFAR10-DVS	AdamW	96	warmup + cosine	16
Spikformer-4-384	CIFAR-10	AdamW	310	warmup + cosine	128
Spikformer-4-384	CIFAR-100	AdamW	410	warmup + cosine	64
Spikformer-2-256	CIFAR10-DVS	AdamW	96	warmup + cosine	16

applied only once. The resulting block architecture is shown in Figure 6. Training uses WikiText-103 sequences of length 4096, BF16 arithmetic, micro-batch size one, 48-step gradient accumulation, AdamW, peak learning rate 4×10^{-4}, final learning rate 4×10^{-5}, weight decay 0.1, and seeds 42, 123, and 2026. Every paired run processes 1.18 billion tokens over ten epochs from the same seed-specific initialization. Each run's final-window training loss averages its last 100 logged measurements. Validation loss and perplexity use the complete WikiText-103 validation split. The table reports mean and sample standard deviation over the three seeds.

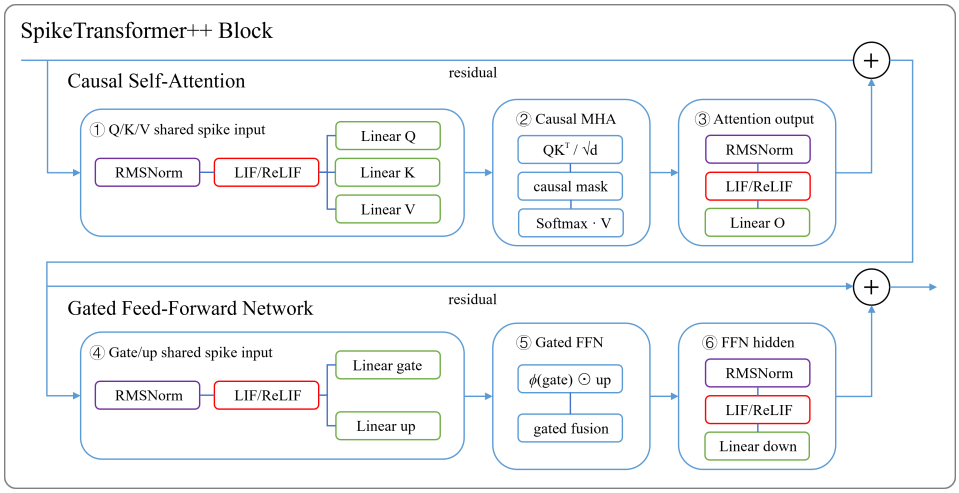

Figure 6: Architecture of a SpikeTransformer++ block. Each block contains four spiking sites corresponding to the shared Q/K/V input, attention output, shared gate/up input, and FFN hidden projection. All spiking projections follow the unified RMSNorm → LIF/ReLIF → Linear structure.

C.3 Speed Measurement

The forward and backward operator benchmarks use an RTX 5090D, CUDA 13.0, PyTorch 2.9.0, FP32 tensors, hidden size 1024, and chunk size 32. We measure $T \in \{4, 8, 16, 64, 256, 1024, 4096\}$. Kernel duration is collected with Nsight Compute using `gpu__time_duration.sum`. Each setting receives three warm-up calls followed by three profiled calls, and all CUDA kernels launched by the corresponding public operator are included. Forward and backward measurements cover standard Gaussian currents, Gaussian currents with standard deviation 0.5, Gaussian currents shifted by -0.5 and 0.5, and unit-variance Laplace currents. Correctness compares the Torch, CuPy, serial CUDA, and parallel CUDA paths using the same spike, membrane, and gradient tensors.

C.4 End-to-End Execution

End-to-end measurements use seed 42 for the 57M, 102M, and 191M SpikeTransformer++ models. Serial and parallel inference use the same ReLIF checkpoint, batch size one, two warm-up validation batches, and one timed pass over the complete WikiText-103 validation split. Before the suite begins,

a 16-token probe compares official SpikingJelly CuPy LIF against the stepwise LIF reference; the observed maximum absolute output difference is zero. Training throughput is taken from runs that share the seed-42 initialization, data order, model configuration, and optimizer settings. Each report covers ten optimizer intervals, and the table averages the final 100 reports from each run. The measured interval contains data transfer, model forward computation, loss evaluation, backward propagation, gradient clipping, and the optimizer update.

C.5 Real-Current Coalescence

The real-current study uses trained ReLIF checkpoints for all three SpikeTransformer++ scales and seeds 42, 123, and 2026. For each checkpoint, we take the first eight sequences of the WikiText-103 validation split, collect currents immediately before every ReLIF site, sample 16 evenly spaced channels per site, and partition each 4096-token sequence into 32-token chunks. The 57M model has 32 sites and yields 524,288 chunk–neuron observations per seed; the 102M and 191M models have 48 sites and each yield 786,432 observations per seed. Currents are collected in FP32 using the stepwise Torch ReLIF recurrence on CPU. Collection stops after the last recurrent block, before the vocabulary projection. Coalescence is evaluated by the real-valued reachable-set oracle. The mean time is conditional on coalescence within the observed chunk; uncoalesced chunks contribute zero to the suffix fraction. We average each statistic across the three seeds and report

$$\mathbb{E}[\tau \mid \tau \le L], \qquad \mathbb{P}(\tau > t), \qquad \mathbb{P}(\tau \le L), \qquad \mathbb{E}\left[\frac{(L-\tau+1)_+}{L}\right]. \tag{39}$$

D Supplementary Results

D.1 Temporal-Parallel Neuron Latencies

Table 6: Absolute operator latency (μs) underlying Figure 4. Inference forward runs without gradients; training F+B times a fresh autograd forward and backward in one call, without an optimizer update. All methods use FP32 inputs with 8192 neurons.

Method	Phase	Sequence length T						
		4	8	16	64	256	1024	4096
ReLIF	Inf. fwd.	58.14	56.56	56.38	70.02	75.81	178.80	799.12
	Train F+B	189.38	187.94	187.86	211.44	214.96	374.62	1465.79
FPT	Inf. fwd.	259.90	300.94	301.23	248.42	307.30	1171.04	12968.54
	Train F+B	1071.28	1153.36	1121.92	974.56	1368.46	3522.34	42645.46
PSN	Inf. fwd.	109.14	110.80	146.43	136.54	139.23	281.50	1287.34
	Train F+B	421.34	381.58	424.56	485.92	853.12	2510.82	10856.22
PRF	Inf. fwd.	258.37	259.47	258.35	258.67	296.21	1776.72	8065.12
	Train F+B	916.16	1013.55	989.47	1025.71	924.98	4687.90	21019.41

D.2 Complete Operator Results

Table 7: Complete FP32 NCU kernel scaling with hidden size 1024 and chunk size 32. Entries give the range across the five controlled current distributions.

T	Fwd./serial	Bwd./serial	F+B/serial	Fwd./CuPy LIF	Bwd./CuPy LIF	F+B/CuPy LIF
4	0.81–0.83×	1.11–1.14×	0.96–0.97×	4.52–4.59×	1.75–1.78×	3.17–3.21×
8	0.79–0.83×	1.46–1.49×	1.10–1.13×	4.46–4.52×	2.09–2.13×	3.37–3.39×
16	0.92–0.93×	2.16–2.18×	1.43–1.44×	4.29–4.33×	2.71–2.77×	3.64–3.68×
64	1.56–1.91×	5.12–5.21×	2.92–3.29×	6.19–7.52×	5.39–5.48×	5.89–6.64×
256	1.70–2.40×	7.49–7.70×	3.62–4.50×	5.90–8.29×	7.32–7.65×	6.38–8.04×
1024	3.74–5.66×	9.08–12.09×	6.33–8.97×	13.37–19.96×	8.72–11.62×	11.12–15.66×
4096	5.41–8.04×	16.33–22.07×	10.35–15.17×	26.37–38.07×	14.91–17.71×	21.18–27.72×

Across all controlled distributions, parallel CUDA ReLIF has zero spike mismatches against the Torch recurrence. The maximum absolute error among membrane states, input gradients, and initial-state gradients is 2.4×10^{-7}. The SpikingJelly CuPy LIF and ReLIF paths are checked against their corresponding Torch references under the same inputs.

D.3 Speedup Metrics and Hardware Analysis

The operator analysis uses the six speedup ratios reported in tables 3 and 7. Let t_P^ϕ denote parallel ReLIF latency for phase $\phi \in \{\mathrm{F}, \mathrm{B}\}$, and let t_S^ϕ and t_C^ϕ denote the corresponding matched serial ReLIF and SpikingJelly CuPy LIF latencies. The phase-wise and combined ratios are

$$S_{\phi/X} = \frac{t_X^\phi}{t_P^\phi}, \qquad S_{\mathrm{F+B}/X} = \frac{t_X^{\mathrm{F}} + t_X^{\mathrm{B}}}{t_P^{\mathrm{F}} + t_P^{\mathrm{B}}}, \qquad X \in \{S, C\}. \tag{40}$$

The three ratios against serial ReLIF isolate the gain from parallelizing the same recurrence with the same tensor outputs. The three ratios against CuPy LIF measure the operator gain available when ReLIF and its parallel backend replace the standard SpikingJelly execution path. Forward, backward, and their sum are reported separately because their parallel structures and hardware costs differ.

The dependency depth in eq. (15) describes the recurrence after independent chunk work is exposed. Measured latency also includes certificate evaluation, tensor traffic, and kernel launch. For required traffic volume $B_{\min}$, sustained bandwidth $\mathrm{BW}_{\mathrm{sust}}$, and launch cost t_{launch}, the parallel latency follows the hardware floor

$$t_P \geq \max\left(t_{\mathrm{depth}}, \frac{B_{\min}}{\mathrm{BW}_{\mathrm{sust}}}, t_{\mathrm{launch}}\right). \tag{41}$$

At short T, launch and local scan costs dominate, which yields forward ratios near unity while the associative backward scan already shortens the local recurrence. As T increases, independent chunks fill more of the GPU and the serial dependency chain grows, producing the scaling in table 7.

We profile the parallel ReLIF kernels with the same FP32 workloads used in table 7: 1024 neurons, chunk size 32, seven sequence lengths, and five current distributions. Nsight Compute reports DRAM bandwidth and the fraction of peak DRAM throughput for every forward and backward kernel. For paths containing multiple kernel launches, operator bandwidth is computed by weighting each kernel's throughput by its measured duration.

At $T = 4$, the warmed tensors reside in on-chip cache and NCU records no DRAM traffic. Through $T = 256$, both paths remain far below the DRAM ceiling, matching the regime where fixed launch and local scan work determine latency. The transition begins at $T = 1024$, although both paths still use less than 3.1% of peak DRAM throughput for the 1024-neuron workload.

At $T = 4096$, the backward anchor-prefix kernel reaches 1.37–1.41 TB/s and 77.6%–80.0% of peak DRAM throughput. After including the boundary-suffix kernel, the complete backward operator sustains 1.09–1.20 TB/s. This streaming traffic forms the practical floor beneath the measured $16.33\times$–$22.07\times$ backward speedup. Together with the short-sequence measurements, these results identify launch and local scan work at small T and sustained tensor traffic at large T as the principal hardware regimes surrounding the measured acceleration.