# PROBABILISTIC NEURONAL DYNAMICS: EXPANDING SPIKING TEMPORAL DYNAMICS FOR PROBABILISTIC SPIKE GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The temporal modeling capability of a spiking neuron depends on both how its dynamics weight historical inputs and how the resulting membrane state is converted into binary spikes. In a primitive reset-free leaky integrate-and-fire (LIF) neuron, these two stages are respectively constrained by a single geometric temporal response and a deterministic threshold that maps membrane values on the same side of the threshold to the same instantaneous event. These coupled restrictions motivate a neuronal mechanism that enriches the temporal response space before firing while allowing the magnitude of the resulting membrane trajectory to modulate spike generation without abandoning binary communication. We propose Probabilistic Neuronal Dynamics (PND), which expands this response space through multiple learnable temporal modes. A learned readout combines these modes into a scalar membrane trajectory, allowing the neuron to shape its response across delays beyond a single geometric decay. An exponential event model maps the readout to a firing probability, so membrane magnitude controls firing tendency while each emitted spike remains hard and binary. We characterize the response space through basis-rank analysis and non-geometric mode composition. Learned-mode measurements reveal diverse response shapes with low redundancy, and a capacity sweep links state expansion to improved task performance before saturation. Component comparisons identify the task-level role of expanded dynamics and examine probabilistic firing with its associated gradient rule. Repeated inference shows stable aggregate predictions under stochastic sampling. Across temporal modeling benchmarks, PND achieves competitive performance while retaining parallelizable pre-spike computation and sparse binary communication.

## 1 INTRODUCTION

Temporal modeling in spiking neural networks (SNNs) is realized through two coupled neuronal operations: *temporal integration* and *event generation*. During temporal integration, neuronal dynamics accumulate input history into a continuous membrane state, determining how inputs arriving at different delays contribute to the current membrane trajectory. Event generation then converts this membrane state into binary spikes for downstream communication. The temporal modeling capability of an SNN therefore depends on both how temporal information is integrated into the membrane state and how that state is expressed through spike events.

For temporal integration, the primitive leaky integrate-and-fire (LIF) neuron provides a compact but restricted mechanism. Its reset-free subthreshold recurrence weights historical inputs through a single geometric temporal response controlled by one decay factor. Learning the membrane time constant changes the effective temporal scale, but the relative weighting across delays still follows the same geometric pattern (Fang et al., 2021). Gating mechanisms and heterogeneous or multi-state neuronal dynamics introduce additional flexibility and richer temporal behavior (Yao et al., 2022; Zheng et al., 2024; Hao et al., 2023). However, introducing more internal variables does not necessarily imply that the resulting temporal responses are distinct or non-redundant. The structure of the response space, and how multiple internal responses combine to form the final membrane trajectory, therefore remain important for understanding whether expanded neuronal dynamics actually provide richer temporal integration.



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.

For event generation, conventional LIF neurons use a deterministic threshold to convert the membrane state into a binary spike. Membrane values on the same side of the threshold produce the same instantaneous event, so differences in membrane magnitude are not reflected in the instantaneous spike output. Probabilistic neurons instead map continuous internal variables to firing probabilities while keeping the realized events binary (Fang et al., 2024; Chakraborty et al., 2024). Related designs separate continuous integration from stochastic firing (Yarga & Wood, 2023) or combine probabilistic events with state-space neuronal models (Bal & Sengupta, 2025). A useful probabilistic event rule should provide a monotonic and bounded membrane-to-probability mapping, preserve hard binary spikes, and remain stable under repeated stochastic inference. Exponential survival mappings provide a natural form for this purpose: increasing membrane drive monotonically raises firing probability, while multiplicative silence probabilities give the mapping a simple compositional interpretation.

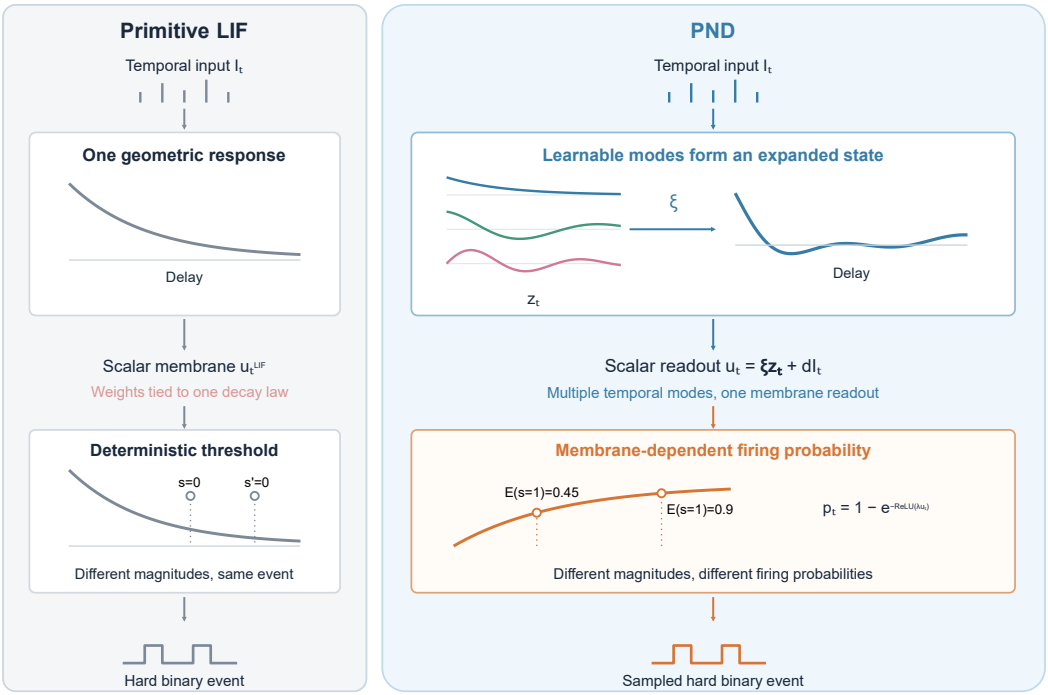



Figure 1: Primitive LIF and PND across temporal integration and event generation. During temporal integration, primitive LIF uses a single geometric response, whereas PND combines multiple learnable modes into a non-geometric scalar membrane trajectory. During event generation, deterministic LIF maps same-side membrane values to the same instantaneous event, whereas PND maps the scalar membrane readout to a firing probability and samples a hard binary spike. Curves and events are schematic; the direct input path is included in the readout equation.

These two directions motivate a unified neuron design for temporal modeling. On the integration side, the neuron should provide multiple distinct temporal responses whose combination extends beyond the single geometric weighting of primitive LIF dynamics. On the event side, the resulting scalar membrane trajectory should modulate firing probability in a graded manner while every emitted event remains hard and binary. Our motivation is therefore not simply to enlarge the neuronal state or to introduce stochastic firing independently, but to connect an explicitly expanded and analyzable temporal response space with a stable membrane-dependent probabilistic event mechanism.

We introduce *Probabilistic Neuronal Dynamics* (PND) following this principle. PND expands primitive LIF dynamics into learnable temporal modes that respond differently to input history. A learned readout compresses these modes into a scalar membrane trajectory whose composite response extends beyond a single geometric decay, preserving the neuron-level communication interface despite the expanded internal state. The linear pre-spike recurrence also admits equivalent temporal-kernel evaluation in parallel across the sequence. An exponential event model maps the scalar readout to a

firing probability and samples a hard binary spike. Expanded dynamics therefore shapes how history contributes to the membrane trajectory, while probabilistic event generation maps its magnitude to spike emission. We characterize this response space through basis rank, non-geometric composition, learned redundancy, and effective dimensionality. Experiments connect expansion capacity to performance, separate the contributions of temporal integration and event generation, and verify stable stochastic inference. PND achieves competitive temporal modeling performance while retaining parallel pre-spike computation and sparse hard binary communication.

Our contributions are threefold.

1. We introduce PND, which expands primitive LIF temporal integration into learnable modes and compresses them into a scalar membrane trajectory for probabilistic hard spikes. Its linear pre-spike dynamics admits parallel kernel evaluation while preserving scalar and binary communication.
2. We characterize the expanded temporal response space analytically and empirically. Distinct modes provide independent temporal response directions under the stated conditions, trained modes retain low redundancy and high effective dimensionality, and increasing expansion capacity improves task performance before saturation.
3. We validate PND through component comparisons, repeated inference, and temporal benchmarks. Timing and firing-rate measurements further verify parallel pre-spike evaluation and sparse binary communication.

## 2 Related Work

Learnable membrane constants, adaptive thresholds, and gating adapt neuronal temporal integration (Yin et al., 2021; Fang et al., 2021; Yao et al., 2022). Contextual modulation and heterogeneous or multi-compartment states support memory across interacting timescales (Chen et al., 2023; Zheng et al., 2024; Chen et al., 2024). These mechanisms enrich temporal behavior, but state dimension alone does not establish the independence of the resulting responses or how they combine into a scalar membrane trajectory.

Probabilistic spiking models connect continuous internal variables to distributional inference and stochastic binary events (Shen et al., 2023; Chakraborty, 2024; Fang et al., 2024; Chakraborty et al., 2024). P-SpikeSSM combines expanded neuronal states with a clipped affine firing probability and a stochastic surrogate (Bal & Sengupta, 2025). These interfaces preserve binary communication while allowing graded firing probabilities. Their conditional event distributions alone do not establish prediction consistency under repeated sampling, which requires empirical evaluation.

Parallel spiking models reorganize temporal integration for sequence evaluation. SPSN separates linear integration from stochastic firing (Yarga & Wood, 2023), while decoupled LIF updates and temporal kernels expose parallel computation across time (Li et al., 2024; Zhong et al., 2024). Richer internal dynamics extend this approach to long-sequence processing (Shen et al., 2025). Parallelism depends on the dependencies retained by the recurrence and event mechanism; it does not by itself characterize temporal response diversity.

## 3 Method

### 3.1 Overview

PND couples temporal integration and event generation through a scalar membrane readout. The design goal is to combine distinct responses to historical inputs and let the resulting membrane magnitude control firing tendency while preserving hard binary communication. A primitive threshold neuron follows $I_{1:t} \to u_t^{\mathrm{LIF}} \to s_t$. PND uses

$$I_{1:t} \longrightarrow \boldsymbol{z}_t \longrightarrow u_t \longrightarrow p_t \longrightarrow s_t, \tag{1}$$

where $\boldsymbol{z}_t \in \mathbb{R}^m$ is the expanded temporal state, $u_t$ is the scalar membrane readout, $p_t$ is the firing probability, and $s_t \in \{0, 1\}$ is the hard spike.

### 3.2 Expanded Dynamics

**Primitive temporal response.** To identify the constraint on temporal integration, consider the reset-free subthreshold LIF recurrence

$$u(t) = \alpha u(t-1) + I(t), \qquad u(0) = 0. \tag{2}$$

Here, $I(t)$ denotes the effective discrete-time input, with the input gain absorbed into its definition (Appendix A). At the last step of a length-$L$ sequence, its unrolled form is

$$u(L) = \sum_{i=1}^{L} \alpha^{L-i} I(i), \qquad K_i = \alpha^{L-i}. \tag{3}$$

Thus, a primitive LIF neuron integrates temporal inputs through a single geometric response characterized by one decay mode.

**Expanded temporal responses.** To shape the relative contributions of inputs across delays beyond one geometric decay, PND expands the internal state of each neuron to $m$ coordinates,

$$\boldsymbol{z}(t) = \boldsymbol{T}\boldsymbol{z}(t-1) + \boldsymbol{\zeta} I(t), \tag{4}$$

where $\boldsymbol{z}(t) \in \mathbb{R}^m$, $\boldsymbol{T} \in \mathbb{R}^{m\times m}$ contains the temporal modes, and $\boldsymbol{\zeta} \in \mathbb{R}^m$ maps the scalar input into the expanded state. A learned row vector $\boldsymbol{\xi} \in \mathbb{R}^{1\times m}$ combines the state with a direct input term to form a scalar membrane readout,

$$u(t) = \boldsymbol{\xi}\boldsymbol{z}(t) + dI(t). \tag{5}$$

Unrolling Eq. (4) and applying Eq. (5) yields

$$u(L) = \sum_{i=1}^{L} K'_i I(i) + dI(L), \tag{6}$$

$$K'_i = \boldsymbol{\xi}\boldsymbol{T}^{L-i}\boldsymbol{\zeta}. \tag{7}$$

The readout weights $\boldsymbol{\xi}$ reinforce or offset temporal-mode contributions across delays, combining them into the scalar membrane trajectory $u(t)$. The term $dI(t)$ provides a direct current-input path. The linear pre-spike dynamics admits the equivalent kernel form $u = K' * I + dI$ for parallel evaluation. Appendix A gives the modal parameterization and its real-valued realization.

### 3.3 Response Space

The purpose of this analysis is to establish when state expansion enriches temporal integration. Basis rank measures the independent response directions available before readout; non-geometric composition characterizes the scalar temporal response after readout. For scalar decay modes, $\boldsymbol{T} = \mathrm{diag}(\alpha_1, \ldots, \alpha_m)$ and $K'_i = \sum_r \xi_r \zeta_r \alpha_r^{L-i}$. The following result states the capacity gained from distinct modes. Appendix B extends the argument to oscillatory modes.

**Proposition 1** (Temporal basis expansion)**.** *For the $r$-th expanded coordinate, define $\boldsymbol{k}_r = [1, \alpha_r, \ldots, \alpha_r^{L-1}]^\top$ and $\boldsymbol{K} = [\boldsymbol{k}_1, \ldots, \boldsymbol{k}_m]$. If $L \geq m$ and the coefficients $\{\alpha_r\}_{r=1}^m$ are pairwise distinct, then*

$$\mathrm{rank}(\boldsymbol{K}) = m. \tag{8}$$

A primitive LIF neuron provides one temporal basis. In contrast, an $m$-dimensional expanded neuron can span up to $m$ independent temporal-response directions. The first $m$ rows form a nonsingular Vandermonde submatrix. Appendix B gives the proof.

Expansion also changes the shape of the response. The primitive kernel has a constant adjacent ratio,

$$\frac{K_{i-1}}{K_i} = \alpha. \tag{9}$$

Writing $w_r = \xi_r \zeta_r$, the expanded kernel is $K'_i = \sum_r w_r \alpha_r^{L-i}$. With multiple active, distinct modes, its adjacent ratio generally varies with $i$. The compressed response is therefore no longer restricted to a single geometric decay. Proposition 1 characterizes the basis before readout, whereas the varying adjacent ratio characterizes the scalar response after mode combination. Appendix B analyzes mode correlation. Section 4.1 tests whether trained modes remain distinct and whether additional response capacity improves prediction.

### 3.4 PROBABILISTIC SPIKING

**Membrane-dependent firing probability.** Event generation should translate the scalar membrane trajectory into a bounded firing probability. The exponential mapping makes this probability increase monotonically with the nonnegative membrane drive and gives silence a multiplicative composition rule. For neuron $j$, the scalar membrane readout parameterizes

$$p_j(t) = 1 - \exp[-\operatorname{ReLU}(\lambda_j u_j(t))]\,, \tag{10}$$

where $\lambda_j$ is a learned scale. The silence probability is $1-p_j(t)$, and $\mathbb{E}[s_j(t) \mid u_j(t)] = p_j(t)$. For additive nonnegative increments of the rectified drive, the silence probability factorizes into the product of the corresponding exponential survival terms. On the active branch, the scaled membrane readout continuously modulates firing tendency. Appendix C gives the exponential random-threshold construction and multiplicative survival identity.

**Hard binary sampling.** Sampling expresses this graded firing tendency through hard binary communication. Let $p_{j,0}(t) = 1 - p_j(t)$ and $p_{j,1}(t) = p_j(t)$. With independent noise $g_{j,c}(t) \sim \mathrm{Gumbel}(0, 1)$, PND samples

$$k_j(t) = \underset{c\in\{0,1\}}{\arg\max}\,[\log p_{j,c}(t) + g_{j,c}(t)], \qquad s_j(t) = \mathbb{I}[k_j(t) = 1]. \tag{11}$$

**Probability-path gradient.** To train the membrane-to-probability path through discrete sampled events, PND uses the straight-through variable

$$\widetilde{s}_j(t) = s_j(t) - \mathrm{sg}(p_j(t)) + p_j(t), \qquad \frac{\partial \widetilde{s}_j(t)}{\partial p_j(t)} = 1, \tag{12}$$

where $\mathrm{sg}(\cdot)$ stops the gradient. The forward pass retains the sampled hard event, whereas the backward pass follows the analytical membrane-to-probability path directly. Sampling equivalence and gradient details are given in Appendix D.

## 4 EXPERIMENTS

We first examine whether expanded dynamics learns distinct temporal responses and whether increasing state size improves prediction. Temporal benchmarks and confusion patterns then evaluate overall sequence-modeling performance. Component comparisons and repeated inference subsequently assess the roles of temporal integration and event generation, and the stability of stochastic predictions. The final analysis reports parallel execution and sparse communication.

### 4.1 TEMPORAL RESPONSE SPACE AND EXPANSION CAPACITY

**Expansion creates a diverse temporal response space.** We inspect PND on sequential CIFAR-10 with 1,024 time steps (sCIFAR-10) using the analysis configuration in Appendix F. The analysis model uses $d_{\mathrm{state}} = 128$, represented by 64 conjugate-paired temporal contributions; mode-level correlation and effective-rank statistics therefore operate on 64 response directions. Figure 2 connects the response basis in Proposition 1 to learned mode shapes and model-wide measurements. The selected Layer-3 neuron contains modes with different decay rates and oscillation frequencies. Their learned combination produces a compressed kernel with sign changes and a nonuniform decay profile. The layer-mean residual of the regularized geometric fit ranges from $0.425$ to $0.816$, complementing the observed response-shape diversity; Appendix F defines the fit.

**Learned temporal modes remain distinct after training.** Mean off-diagonal absolute correlations range from $0.0528$ to $0.0656$ across the four layers. Figure 2(b) shows the Layer-3 matrix, where most mode pairs remain weakly correlated. After normalizing each temporal-mode column to unit norm, the mean effective ranks range from $54.43$ to $57.11$ out of 64. This normalization measures response-shape diversity independently of amplitude. Together, the pairwise and spectral measurements show that training retains a broad set of temporal directions. Appendix F reports amplitude-sensitive ranks and complete statistics.

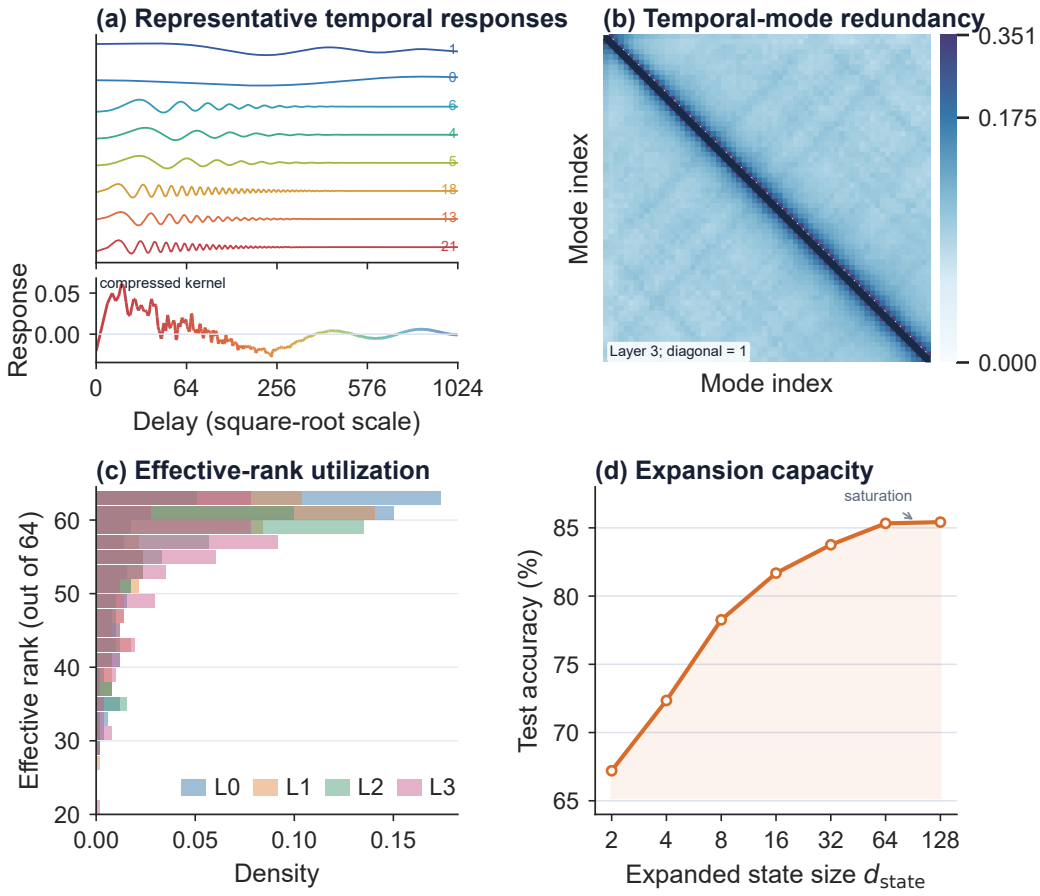



Figure 2: Expanded temporal dynamics on sCIFAR-1024. (a) Eight modes from Layer 3, channel 192, normalized individually and shown on a square-root delay axis to resolve early oscillations; the compressed kernel retains its original amplitude. (b) Mean absolute mode correlation in Layer 3. The blue-black diagonal denotes unit self-correlation, while a power-scaled color map resolves low off-diagonal values. (c) Distributions of unit-column-normalized effective rank, reported as the actual rank out of 64 modes. (d) Test accuracy versus expanded state size.

**Additional expansion capacity improves performance before saturation.** Increasing $d_{\text{state}}$ from 2 to 64 raises sCIFAR-1024 test accuracy from $67.20\%$ to $85.33\%$; increasing it further to 128 gives $85.42\%$. The sweep links expansion to improved prediction, while the diminishing gain beyond 64 states indicates saturation. Together with the low correlations and high effective rank, it shows that added coordinates preserve distinct response shapes and provide usable temporal capacity rather than redundant state.

### 4.2 Temporal Modeling Performance

**PND maintains competitive temporal modeling performance.** We evaluate PND on the five Long Range Arena tasks, which test temporal modeling over hierarchical, textual, document-matching, and sequentialized visual inputs. Table 1 examines whether the same neuronal mechanism transfers across these distinct dependency structures.

PND obtains an average accuracy of $83.87\%$, compared with $82.39\%$ for PRF and $82.23\%$ for SpikingSSM-Vt. It leads the listed SNNs on sCIFAR and Pathfinder, and reaches $59.76\%$ on ListOps and $89.50\%$ on AAN. These tasks require the model to combine information across nested operations, paired documents, and long sequentialized images. Performance across these input structures shows that the expanded temporal integration is useful beyond a single sequence type. The com-

Table 1: Reported test accuracy (%) on Long Range Arena tasks. The average covers the five displayed tasks. Bold marks the leading result among the listed SNNs. PND evaluation and checkpoint selection are specified in Appendix E.

| Method | SNN | ListOps | IMDB | AAN | sCIFAR | Path. | Avg. |
|---|---|---|---|---|---|---|---|
| Transformer (Tay et al., 2020) | No | 36.37 | 64.27 | 57.46 | 42.44 | 71.40 | 54.39 |
| S4D-Lin (Gu et al., 2022) | No | 60.52 | 86.97 | 90.96 | 87.93 | 93.96 | 84.07 |
| SpikeLMUFormer (Liu et al., 2024) | Yes | 37.30 | 65.80 | 79.76 | 55.65 | 72.68 | 62.24 |
| SpikingSSM-VF (Shen et al., 2025) | Yes | 59.93 | 82.35 | 88.20 | 86.81 | 93.68 | 82.19 |
| SpikingSSM-Vt (Shen et al., 2025) | Yes | **60.23** | 80.41 | 88.77 | 88.21 | 93.51 | 82.23 |
| PRF (Huang et al., 2024) | Yes | 59.20 | **86.33** | **89.88** | 84.77 | 91.76 | 82.39 |
| P-SpikeSSM (Bal & Sengupta, 2025) | Yes | 58.20 | 81.20 | 88.53 | 82.40 | 84.80 | 79.03 |
| BinaryS4D (Stan & Rhodes, 2024) | Yes | 54.80 | 82.50 | 85.03 | 82.00 | 82.60 | 77.39 |
| S6-based SNN (Bal & Sengupta, 2024) | Yes | 55.70 | 77.62 | 88.48 | 80.10 | 83.41 | 77.06 |
| **PND** | Yes | 59.76 | 84.78 | 89.50 | **88.48** | **96.84** | **83.87** |

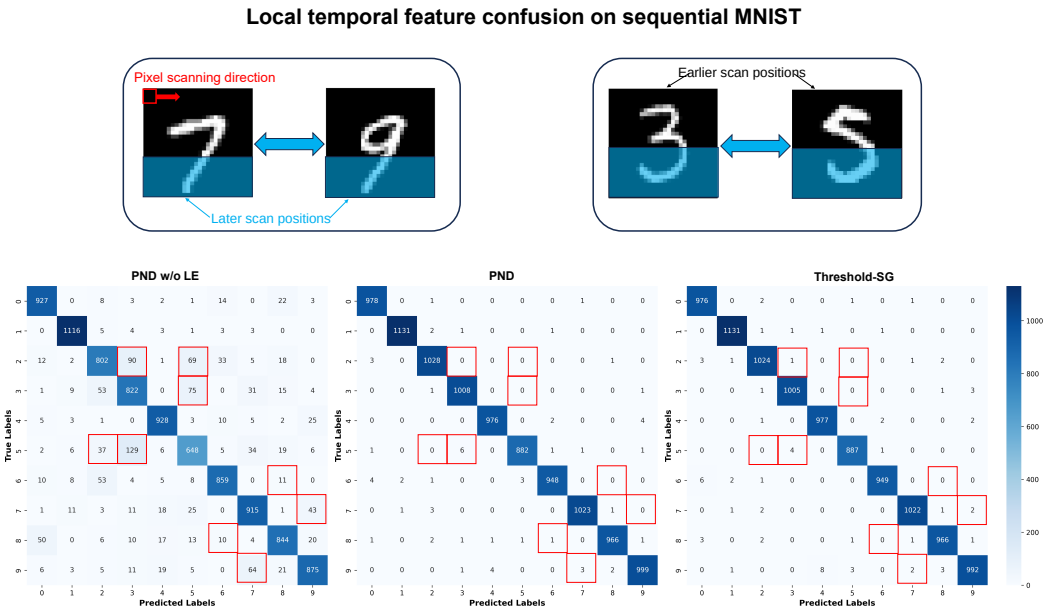



Figure 3: Structural confusions on sequential MNIST. Each image is flattened into a 784-step sequence. Outlined cells mark the 2/3, 2/5, 3/5, and 7/9 digit pairs. PND w/o LE produces 1,264 total errors, whereas Threshold-SG and PND produce 71 and 61, respectively. Comparing PND with PND w/o LE isolates the effect of expanded temporal dynamics under the same probabilistic event rule.

ponent results in Table 2(a) further connect this performance to the two parts of PND: dimension expansion improves over the single-mode variant on four tasks, while the complete probabilistic mechanism provides its largest additional gain on Pathfinder. Appendix H extends the evaluation to sequential static images, neuromorphic visual inputs, and raw speech.

**Confusion patterns reveal how temporal integration affects prediction.** Sequential MNIST presents each image as a 784-step pixel stream, so strokes that are adjacent in space can appear at distant sequence positions. PND w/o LE concentrates errors among digits sharing local structures: its bidirectional 3/5, 2/3, 2/5, and 7/9 confusions total 204, 143, 106, and 107, respectively. Under the same probabilistic event rule, PND reduces the total error count from 1,264 to 61, directly linking expanded temporal integration to fewer structural confusions. Threshold-SG produces 71 errors, showing that expanded dynamics also supports accurate classification with deterministic events.

PND gives the lowest total error, consistent with the complementary contribution of its temporal integration and event-generation mechanisms.

### 4.3 Component Analysis

**The two substitutions expose complementary contributions to PND.** PND w/o LE (linear expansion) retains probabilistic event generation but replaces the expanded dynamics with a primitive single-mode LIF trajectory, isolating the effect of dimension-expanded temporal integration. Threshold-SG retains the expanded dynamics but replaces probabilistic firing and its probability-path gradient with deterministic threshold firing and a surrogate gradient, evaluating the complete probabilistic event mechanism.

Table 2: Component contributions and stochastic-inference stability. (a) Test accuracy (%) under substitutions of temporal integration or event generation. (b) Variation across ten stochastic test passes of a fixed checkpoint; standard deviation is in percentage points and agreement is the mean fraction of runs supporting the majority prediction.

| | Model | sCIFAR | AAN | ListOps | IMDB | Path. |
|---|---|---|---|---|---|---|
| **(a) Component analysis** | PND | 88.48 | 89.50 | 59.76 | 84.78 | 96.84 |
| | Threshold-SG | 89.50 | 88.80 | 60.20 | 84.05 | 81.89 |
| | PND w/o LE | 68.53 | 83.20 | 46.25 | 87.17 | 50.76 |
| **(b) Stochastic inference** | **Metric** | **sCIFAR** | **AAN** | **ListOps** | **IMDB** | **Path.** |
| | Accuracy std. (pp) | 0.148 | 0.107 | 0.503 | 0.162 | 0.062 |
| | Agreement (%) | 96.54 | 97.12 | 84.27 | 92.48 | 96.68 |

Table 2(a) first isolates dimension expansion under the same probabilistic event rule. Relative to PND w/o LE, PND improves accuracy by $19.95$, $6.30$, $13.51$, and $46.08$ percentage points on sCIFAR, AAN, ListOps, and Pathfinder, respectively, while the single-mode variant performs better on IMDB. The gains across the other four tasks show that multiple temporal modes can provide more effective integration than a primitive single-mode trajectory. The second comparison keeps the expanded dynamics and changes the event mechanism. PND remains close to Threshold-SG on sCIFAR, ListOps, and IMDB, improves AAN by $0.70$ points, and yields its clearest gain on Pathfinder, from $81.89\%$ to $96.84\%$. The two comparisons therefore show broad contributions from dimension-expanded integration and a clear benefit from the complete probabilistic spiking mechanism on Pathfinder, while preserving comparable performance on the remaining tasks.

**Stochastic inference remains stable across repeated samples.** We repeatedly evaluate fixed checkpoints while changing only sampling randomness. Accuracy standard deviation measures aggregate performance variation; agreement measures consistency of individual predictions. This diagnostic uses checkpoints distinct from the benchmark runs in Table 1. Table 2(b) reports accuracy deviations below $0.15$ percentage points and agreement above $96\%$ on sCIFAR, AAN, and Pathfinder. ListOps shows the most inference variation, with $0.503$-point deviation and $84.27\%$ agreement. Thus, aggregate accuracy remains stable across repeated event samples, while individual predictions vary more on ListOps. Appendix G specifies this repeated-inference experiment and provides additional statistics and a separate sample-level diagnostic.

### 4.4 Parallel Evaluation and Sparse Communication

**Expanded dynamics retains parallel sequence evaluation.** The linear recurrence before spike generation admits kernel evaluation across the full sequence. Table 3(a) compares recorded iteration rates on IMDB using recurrent and parallel execution on an RTX 4090. The recorded acceleration grows with sequence length and reaches $154.68\times$ at 4K steps. The equivalent kernel removes the time-step dependency within the pre-spike expanded dynamics. These recorded loop rates do not measure end-to-end model acceleration. Appendix I describes the timing record.

**Sparse hard events expose an opportunity to reduce projection arithmetic.** Table 3(b) reports global firing rates below $10\%$ on all five tasks. Each event entering the $H \to 2H$ pointwise projection

Table 3: Parallel execution and sparse propagation. (a) Recurrent and parallel iteration times and recorded acceleration. (b) Global firing rate and event-driven projection energy under the 45 nm FP32 arithmetic model.

| | Metric | 1K | 2K | 3K | 4K | |
|---|---|---|---|---|---|---|
| **(a) Parallel execution** | Recurrent (s/it) | 4.65 | 9.25 | 14.32 | 17.36 | |
| | Parallel (s/it) | .0818 | .0877 | .0937 | .1122 | |
| | Speedup (×) | 56.82 | 105.45 | 152.79 | 154.68 | |
| **(b) Sparse propagation** | **Metric** | **ListOps** | **IMDB** | **AAN** | **sCIFAR** | **Path.** |
| | Global firing (%) | 8.056 | 1.128 | 1.645 | 9.403 | 5.571 |
| | Projection energy (mJ/example) | 0.107 | 0.026 | 0.093 | 0.273 | 0.040 |

induces $2H$ accumulations. Combining the measured event counts with $E_{\mathrm{AC}} = 0.9\,\mathrm{pJ}$ under the 45 nm FP32 arithmetic model gives $0.026$–$0.273\,\mathrm{mJ}$ per evaluated example (Horowitz, 2014; Zhou et al., 2023). The hard output supports sparse inter-layer projection under event-driven execution. These are operation-level estimates for post-spike pointwise projections, not measurements of end-to-end hardware energy. Appendix I gives the counts, dense reference, and estimate scope.

## 5 Conclusion

PND enriches the temporal computation of spiking neurons by expanding their internal response space and coupling a scalar membrane readout to probabilistic hard spike generation. Distinct modes provide independent response directions whose learned combination extends beyond a primitive geometric decay. Learned-mode analyses, component comparisons, and temporal modeling benchmarks connect this response space to task behavior. Repeated inference shows stable aggregate predictions under stochastic sampling. PND retains parallelizable pre-spike dynamics and sparse binary communication.

### AI use statement

Generative AI tools were used to assist with language editing and presentation, including improving grammar, readability, conciseness, terminology consistency, and the clarity of manuscript wording. They were also used to assist in identifying potentially relevant literature, which was subsequently checked and verified by the authors. The research questions, methodological design, theoretical arguments, experimental setup, result analysis, and scientific conclusions were determined by the authors. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content of this work.

### Reproducibility statement

The anonymized code repository is available at `https://anonymous.4open.science/r/PND-SNN-codes-A0FB`. The method assumptions and complete proofs are provided in Appendices A–D. Appendix E records dataset preprocessing, model configurations, optimization settings, software versions, and the firing-rate measurement protocol. Appendices F–I define the analysis metrics, give extended results, and state the operation-counting assumptions used for the timing and energy analyses.

## References


Malyaban Bal and Abhronil Sengupta. Rethinking spiking neural networks as state space models. *arXiv preprint arXiv:2406.02923*, 2024.

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*, 2025.

Abhisek Chakraborty. Probabilistic spike train inference. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5685–5689. IEEE, 2024.

Ayan Chakraborty, Archita Hore, Chittotosh Ganguly, and Saswat Chakrabarti. Probabilistic boolean operations in spiking neurons for neuromorphic systems. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(6):4250–4260, 2024.

Xinyi Chen, Jibin Wu, Huajin Tang, Qinyuan Ren, and Kay Chen Tan. Unleashing the potential of spiking neural networks for sequential modeling with contextual embedding. *arXiv preprint arXiv:2308.15150*, 2023.

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.

Senlin Fang, Yiwen Liu, Chengliang Liu, Jingnan Wang, Yuanzhe Su, Yupo Zhang, Hoiio Kong, Zhengkun Yi, and Xinyu Wu. Probabilistic spiking neural network for robotic tactile continual learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 530–536. IEEE, 2024.

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, 2021.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022.

Zecheng Hao, Xinyu Shi, Zihan Huang, Tong Bu, Zhaofei Yu, and Tiejun Huang. A progressive training framework for spiking neural networks with learnable multi-hierarchical model. In *The Twelfth International Conference on Learning Representations*, 2023.

Mark Horowitz. Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pp. 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

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*, 2024.

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.

Zeyu Liu, Gourav Datta, Anni Li, and Peter Anthony Beerel. Lmuformer: Low complexity yet powerful spiking model with legendre memory units. *arXiv preprint arXiv:2402.04882*, 2024.

Dongrui Shen, Dan Yao, Stephen McLaughlin, and Yoann Altmann. Probabilistic spiking neural networks training with expectation-propagation. In *2023 IEEE 9th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 41–45. IEEE, 2023.

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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20380–20388, 2025. doi: 10.1609/aaai.v39i19.34245.

Matei-Ioan Stan and Oliver Rhodes. Learning long sequences in spiking neural networks. *Scientific Reports*, 14(1):21957, 2024.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35: 32160–32171, 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.

Hanle Zheng, Zhong Zheng, Rui Hu, Bo Xiao, Yujie Wu, Fangwen Yu, Xue Liu, Guoqi Li, and Lei Deng. Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics. *Nature Communications*, 15(1):277, 2024.

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 *International Conference on Learning Representations*, 2023.

# A Expanded LIF Dynamics and Kernel Formulation

## A.1 Primitive LIF kernel

With the resting potential set to zero, the continuous subthreshold equation is $\tau_m \dot{u} = -u + RI_{\text{phys}}$. Holding the physical input current constant over a step of duration $\Delta$ gives

$$u(t) = \alpha u(t-1) + \beta I_{\text{phys}}(t), \qquad \alpha = e^{-\Delta/\tau_m}, \qquad \beta = R(1-\alpha).$$

Thus, $R = 1$ yields the familiar input factor $1 - \alpha$. Equation (2) uses the effective input $I(t) = \beta I_{\text{phys}}(t)$. Relative to the physical current, the kernel is $K_\tau^{\text{phys}} = \beta\alpha^\tau$; relative to the effective input, it is $K_\tau = \alpha^\tau$. A nonzero constant gain changes the amplitude but preserves the single geometric response. In the expanded recurrence, the input gain is included in $\boldsymbol{\zeta}$.

Starting from the reset-free recurrence in Eq. (2), repeated substitution gives

$$u(t) = \alpha^t u(0) + \sum_{i=1}^{t} \alpha^{t-i} I(i). \tag{13}$$

With $u(0) = 0$, define the delay-indexed kernel $K_\tau = \alpha^\tau$ for $\tau = 0, \ldots, L-1$. The membrane sequence is then the causal convolution

$$u(t) = \sum_{\tau=0}^{t-1} K_\tau I(t-\tau), \qquad u = K * I. \tag{14}$$

At $t = L$, this recovers Eq. (3). A primitive neuron therefore uses one geometric kernel over all input delays.

## A.2 Expanded dynamics and compressed kernel

For the expanded recurrence in Eq. (4), repeated substitution yields

$$\boldsymbol{z}(t) = \boldsymbol{T}^t \boldsymbol{z}(0) + \sum_{i=1}^{t} \boldsymbol{T}^{t-i} \boldsymbol{\zeta} I(i). \tag{15}$$

Setting $\boldsymbol{z}(0) = \mathbf{0}$ and applying the learned compression with the direct input term gives

$$\begin{aligned} u(t) &= \sum_{i=1}^{t} \boldsymbol{\xi} \boldsymbol{T}^{t-i} \boldsymbol{\zeta} I(i) + dI(t) \\ &= \sum_{\tau=0}^{t-1} K'_\tau I(t-\tau) + dI(t), \qquad K'_\tau = \boldsymbol{\xi} \boldsymbol{T}^\tau \boldsymbol{\zeta}. \end{aligned} \tag{16}$$

For scalar modes with $\boldsymbol{T} = \text{diag}(\alpha_1, \ldots, \alpha_m)$,

$$K'_\tau = \sum_{r=1}^{m} \xi_r \zeta_r \alpha_r^\tau. \tag{17}$$

Thus, state expansion and compression are absorbed into one scalar temporal kernel. Parallel evaluation does not require materializing the $m$-dimensional state at every time step.

## A.3 Oscillatory modes and real-valued membrane readout

The implementation represents decaying oscillations by conjugate-paired modes. For each channel, let $N = d_{\text{state}}$ be even and $q = N/2$. The $q$ stored continuous-time coefficients and the positive discretization step are

$$a_r = -\exp(\eta_r) + \mathrm{i}\omega_r, \qquad \Delta = \exp(\delta) > 0, \qquad r = 0, \ldots, q-1. \tag{18}$$

The negative real part makes each continuous mode stable. Holding the scalar input constant over an interval of length $\Delta$ gives

$$\alpha_r = \exp(\Delta a_r), \qquad b_r = \int_0^\Delta \exp(a_r s)\, ds = \frac{\exp(\Delta a_r) - 1}{a_r}. \tag{19}$$

Writing $h_r(t) \in \mathbb{C}$ for the stored modal state, the corresponding recurrence and real-valued readout are

$$h_r(t) = \alpha_r h_r(t-1) + b_r I(t), \qquad u(t) = 2\,\mathrm{Re}\left[\sum_{r=0}^{q-1} c_r h_r(t)\right] + dI(t), \tag{20}$$

where $c_r$ is learned and $h_r(0) = 0$. Unrolling yields the kernel used in the forward pass:

$$K'_\tau = 2\,\mathrm{Re}\left[\sum_{r=0}^{q-1} c_r b_r \alpha_r^\tau\right], \qquad u = K' * I + dI. \tag{21}$$

Thus $N$ real internal coordinates are stored as $q$ complex coordinates. Each conjugate pair contributes one real temporal curve to the empirical mode analysis.

This parameterization has an equivalent real form in Eq. (4). Write $\alpha_r = \alpha_{r,\mathrm{R}} + \mathrm{i}\alpha_{r,\mathrm{I}}$ and use the two-coordinate state $[\mathrm{Re}\, h_r, \mathrm{Im}\, h_r]^\top$. Its transition block, input vector, and readout are

$$T_r = \begin{bmatrix} \alpha_{r,\mathrm{R}} & -\alpha_{r,\mathrm{I}} \\ \alpha_{r,\mathrm{I}} & \alpha_{r,\mathrm{R}} \end{bmatrix}, \qquad \zeta_r = \begin{bmatrix} \mathrm{Re}\, b_r \\ \mathrm{Im}\, b_r \end{bmatrix}, \qquad \xi_r = 2[\mathrm{Re}\, c_r, -\mathrm{Im}\, c_r]. \tag{22}$$

Stacking the blocks gives $\boldsymbol{T} \in \mathbb{R}^{N\times N}$ and a real membrane potential. The direct term changes only the zero-delay coefficient of the total input-to-membrane kernel. The temporal-mode figures analyze $K'$ separately from this direct contribution.

### A.4 Parallel kernel evaluation

For a length-$L$ sequence, the model constructs $K' = [K'_0, \ldots, K'_{L-1}]$ from the learned neuronal parameters. Linear convolution is evaluated by zero-padding the input and kernel to length $2L$:

$$u = \mathrm{crop}_L[\mathrm{IRFFT}(\mathrm{RFFT}(I, 2L) \odot \mathrm{RFFT}(K', 2L), 2L)] + dI, \tag{23}$$

where $\odot$ denotes pointwise multiplication and $\mathrm{crop}_L$ retains the first $L$ causal outputs. Zero-padding prevents circular wraparound. Kernel construction costs $O(mL)$ operations per neuron, followed by $O(L \log L)$ FFT convolution per input sequence. This evaluation removes the sequential dependency between time steps in the expanded temporal dynamics. Hard event generation is applied to the resulting membrane sequence.

**Initialization.** We use a structured spectral initialization motivated by HiPPO-based temporal representations (Gu et al., 2020; 2022). Specifically, the kernel initializes $a_r = -\frac{1}{2} + \mathrm{i}\pi r$, draws $\log \Delta$ uniformly between $\log 10^{-3}$ and $\log 10^{-1}$, and initializes $c_r$ with complex Gaussian values. The decay parameters, frequencies, discretization steps, and readout weights are learned.

## B Temporal Response Theory

This section gives the complete arguments behind the response-space statements in Section 3.3. We use the delay convention

$$k_r(\tau) = \alpha_r^\tau, \qquad \tau = 0, \ldots, L-1, \tag{24}$$

and initially consider real, pairwise-distinct coefficients. The rank result is algebraic and does not require stability. The correlation limit additionally assumes $|\alpha_r| < 1$. The extension to the implemented oscillatory modes is given below.

### B.1 Temporal-basis rank

For $m$ expanded modes, collect their length-$L$ responses as

$$\boldsymbol{K} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{L-1} & \alpha_2^{L-1} & \cdots & \alpha_m^{L-1} \end{bmatrix}. \tag{25}$$

*Proof of Proposition 1.* Because $L \geq m$, the first $m$ rows form the square Vandermonde matrix

$$\boldsymbol{K}_{1:m} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{m-1} & \alpha_2^{m-1} & \cdots & \alpha_m^{m-1} \end{bmatrix}. \tag{26}$$

Its determinant is

$$\det(\boldsymbol{K}_{1:m}) = \prod_{1 \leq r < q \leq m} (\alpha_q - \alpha_r). \tag{27}$$

Pairwise-distinct coefficients make every factor nonzero, so $\boldsymbol{K}_{1:m}$ is nonsingular. Hence $\operatorname{rank}(\boldsymbol{K}) \geq m$. Since $\boldsymbol{K}$ has exactly $m$ columns, $\operatorname{rank}(\boldsymbol{K}) = m$. □

The scalar primitive LIF response is the special case $m = 1$. Dimension expansion therefore changes the number of available independent response directions from one to $m$. Learned compression selects one combination for the membrane trajectory. Algebraic rank characterizes the available response space; the singular spectrum measures how evenly those directions are represented numerically.

**Oscillatory extension.** The same determinant argument holds over $\mathbb{C}$ for pairwise-distinct discrete coefficients. A conjugate pair $\alpha, \overline{\alpha}$ can be represented by the two real sequences $\operatorname{Re}(\alpha^\tau)$ and $\operatorname{Im}(\alpha^\tau)$. The transformation between these sequences and the conjugate columns is invertible when the conjugate columns are distinct. Thus, with $L \geq m$ and $m$ distinct coefficients closed under conjugation, the equivalent real temporal basis has rank $m$. A real coefficient contributes one direction; coincident or aliased discrete coefficients are counted once. The proposition concerns the basis before readout. Appendix F measures the $q = N/2$ real, readout-weighted pair contributions in Eq. (21); its effective-rank denominator is $q$.

### B.2 Departure from a single geometric response

The primitive kernel satisfies

$$\frac{K_{\tau+1}}{K_\tau} = \alpha \tag{28}$$

whenever $K_\tau \neq 0$. Its delay profile is therefore restricted to one geometric sequence. For the expanded kernel, write $w_r = \xi_r \zeta_r$ and let $\mathcal{A} = \{r : w_r \neq 0\}$ denote the active modes:

$$K'_\tau = \sum_{r \in \mathcal{A}} w_r \alpha_r^\tau, \qquad \frac{K'_{\tau+1}}{K'_\tau} = \frac{\sum_{r \in \mathcal{A}} w_r \alpha_r^{\tau+1}}{\sum_{r \in \mathcal{A}} w_r \alpha_r^\tau}. \tag{29}$$

The ratio depends on $\tau$ through the relative contribution of each mode.

More precisely, suppose $n_a = |\mathcal{A}| \geq 2$, the active $\alpha_r$ are distinct, and $L \geq n_a + 1$. If $K'_\tau = c\beta^\tau$ for all $\tau = 0, \ldots, L-1$, then

$$\sum_{r \in \mathcal{A}} w_r \alpha_r^\tau - c\beta^\tau = 0 \quad \text{for} \quad \tau = 0, \ldots, n_a. \tag{30}$$

When $\beta$ differs from every active $\alpha_r$, these $n_a + 1$ equations form a nonsingular Vandermonde system and force all $w_r$ and $c$ to zero. When $\beta$ equals one active coefficient, merging that term forces every other active weight to zero. Both cases contradict the assumption of at least two active modes. Under these assumptions, the expanded response is not a single geometric sequence. The argument also holds over $\mathbb{C}$; applying it to the distinct active conjugate coefficients covers the real oscillatory kernel. The direct input term affects only delay zero, so the argument can be applied to the remaining delays when sufficiently many samples remain.

### B.3 Finite-length mode redundancy

Let the input satisfy

$$\mathbb{E}[I(i)] = 0, \qquad \operatorname{Cov}(I(i), I(j)) = 0 \ (i \neq j), \qquad \operatorname{Var}(I(i)) = \sigma_I^2 > 0. \tag{31}$$

Before compression, the $r$-th expanded coordinate at step $L$ is

$$z_r(L) = \zeta_r \sum_{i=1}^{L} \alpha_r^{L-i} I(i). \tag{32}$$

For nonzero $\zeta_r$ and $\zeta_q$,

$$\mathrm{Cov}[z_r(L), z_q(L)] = \zeta_r \zeta_q \sigma_I^2 \sum_{k=0}^{L-1} (\alpha_r \alpha_q)^k, \tag{33}$$

$$\mathrm{Var}[z_r(L)] = \zeta_r^2 \sigma_I^2 \sum_{k=0}^{L-1} \alpha_r^{2k}. \tag{34}$$

Consequently, the correlation magnitude is

$$|\rho_{rq}(L)| = \frac{\left|\sum_{k=0}^{L-1} (\alpha_r \alpha_q)^k\right|}{\sqrt{\sum_{k=0}^{L-1} \alpha_r^{2k}} \sqrt{\sum_{k=0}^{L-1} \alpha_q^{2k}}}. \tag{35}$$

This is the absolute normalized inner product between the two response vectors $[1, \alpha_r, \ldots, \alpha_r^{L-1}]$ and $[1, \alpha_q, \ldots, \alpha_q^{L-1}]$. Cauchy–Schwarz gives $|\rho_{rq}(L)| \le 1$. Equality requires proportional vectors. Since both first entries equal one, proportionality requires equality of the vectors; for $L \ge 2$, their second entries then require $\alpha_r = \alpha_q$. Hence distinct modes are not perfectly correlated under the stated input model.

If $|\alpha_r| < 1$ and $|\alpha_q| < 1$, taking $L \to \infty$ gives

$$|\rho_{rq}| = \frac{\sqrt{(1-\alpha_r^2)(1-\alpha_q^2)}}{|1 - \alpha_r \alpha_q|}. \tag{36}$$

The identity

$$(1 - \alpha_r \alpha_q)^2 - (1 - \alpha_r^2)(1 - \alpha_q^2) = (\alpha_r - \alpha_q)^2 \tag{37}$$

shows that distinct stable coefficients make the denominator strictly larger than the numerator. Therefore $|\rho_{rq}| < 1$. This is a statement of non-perfect correlation under the white-input model. The correlation can approach one as coefficients approach each other.

**Connection to the empirical measurements.** For arbitrary real impulse responses $f_r$ and $f_q$, including the real pair contributions in Eq. (21), the same white-input calculation gives $\mathrm{Corr}(z_r, z_q) = \langle f_r, f_q \rangle / (\|f_r\|_2 \|f_q\|_2)$. Appendix F reports centered Pearson correlation along the delay axis, obtained by replacing each $f$ by $f - \bar{f}\mathbf{1}$. This measures similarity of response shape after removing its mean. The empirical effective ranks additionally summarize the full collection of response directions. These measurements assess the degree of learned redundancy beyond the algebraic rank and non-perfect-correlation results.

# C Exponential Event Model

## C.1 Membrane-to-probability mapping

For the scalar membrane readout $u$, define the nonnegative auxiliary quantity

$$e = \mathrm{ReLU}(\lambda u) \ge 0. \tag{38}$$

One equivalent construction introduces a unit-rate exponential latent threshold $X \sim \mathrm{Exp}(1)$ and emits an event when the auxiliary quantity reaches that threshold:

$$s = \mathbb{I}[X \le e]. \tag{39}$$

Because $P(X > e) = \exp(-e)$,

$$P(s = 0 \mid e) = e^{-e}, \qquad P(s = 1 \mid e) = 1 - e^{-e}. \tag{40}$$

The mapping is monotone in $e$, satisfies $p(0) = 0$, and approaches one as $e$ grows. Its derivative $dp/de = e^{-e}$ is largest near $e = 0$ and decreases smoothly as the firing probability saturates. The learned scale $\lambda$ is unconstrained in the implementation; rectification ensures a nonnegative $e$ for either sign of the membrane value.

### C.2 Nonnegative increments and multiplicative survival

For any nonnegative increments $e_1, \ldots, e_R$ whose sum is $e = \sum_r e_r$, the exponential link factorizes:

$$P(s = 0 \mid e) = \exp\left(-\sum_{r=1}^{R} e_r\right) = \prod_{r=1}^{R} \exp(-e_r). \tag{41}$$

This identity describes how the silence probability changes with nonnegative increments of the auxiliary quantity $e$. In the neuron, signed temporal responses are first combined into $u$ and then rectified to obtain $e$. The mode contributions can reinforce or cancel before rectification. Consequently, the factors above refer to increments of $e$, not separate probabilities assigned to the signed temporal modes.

### C.3 Conditional event sampling

Along the nonnegative threshold axis, the exponential latent threshold satisfies, for $a, b \geq 0$,

$$P(X > a + b \mid X > a) = \frac{P(X > a + b)}{P(X > a)} = \frac{e^{-(a+b)}}{e^{-a}} = e^{-b} = P(X > b). \tag{42}$$

For temporal evaluation, PND draws fresh sampling noise at each neuron and step, conditional on the computed membrane sequence. The input history enters through the expanded dynamics. Equation (42) is a property of the latent threshold distribution; temporal dependence of the membrane readout is determined by the neuronal recurrence.

## D Binary Sampling and Gradient Estimation

### D.1 Binary Gumbel-Max sampling

For a fixed neuron $j$ and time $t$, we suppress these indices throughout the scalar derivation, writing $p = p_j(t)$, $g_c = g_{j,c}(t)$, $k = k_j(t)$, and $s = s_j(t)$. Let $p_0 = 1 - p$ and $p_1 = p$, with $0 < p < 1$. Draw independent standard Gumbel variables $g_0, g_1$ and define

$$k = \arg\max_{c \in \{0,1\}} [\log p_c + g_c], \qquad s = \mathbb{I}[k = 1]. \tag{43}$$

The Gumbel-Max identity gives

$$P(k = c) = \frac{e^{\log p_c}}{e^{\log p_0} + e^{\log p_1}} = p_c. \tag{44}$$

For completeness, using the standard Gumbel CDF $F(g) = \exp[-e^{-g}]$ and density $f(g) = e^{-g}F(g)$,

$$\begin{aligned} P(k = 1) &= \int_{-\infty}^{\infty} f(g) F\left(g + \log \frac{p_1}{p_0}\right) dg \\ &= \int_{0}^{\infty} \exp\left[-v\left(1 + \frac{p_0}{p_1}\right)\right] dv \\ &= \frac{p_1}{p_0 + p_1} = p_1, \end{aligned} \tag{45}$$

where the substitution is $v = e^{-g}$. The implementation constructs logits from $p_{\text{safe}} = \text{clip}(p, 10^{-7}, 1 - 10^{-7})$. The forward sample therefore follows $\text{Bernoulli}(p_{\text{safe}})$, while the gradient path below uses $p$. Clipping prevents infinite logarithms and changes the event probability by at most $10^{-7}$.

### D.2 Probability-path straight-through estimator

Sampling is discrete, so its pathwise derivative is unavailable. PND uses

$$\widetilde{s} = s - \text{sg}(p) + p, \tag{46}$$

where sg stops the gradient. Numerically, the forward value is exactly $s \in \{0, 1\}$ because the last two terms cancel. In the backward pass,

$$\frac{\partial \widetilde{s}}{\partial p} = 1. \tag{47}$$

For the auxiliary quantity $e = \mathrm{ReLU}(\lambda u)$ and $p = 1 - e^{-e}$,

$$\frac{\partial p}{\partial e} = e^{-e}. \tag{48}$$

With $e = \mathrm{ReLU}(\lambda u)$, away from the ReLU kink,

$$\frac{\partial \widetilde{s}}{\partial u} = \mathbb{I}[\lambda u > 0]\, \lambda e^{-e}. \tag{49}$$

The estimator sends gradients through the analytical membrane-to-probability path. It is a biased straight-through estimator for the discrete sampling node.

### D.3 Implementation and temperature

The implementation uses tensors in $(B, H, L)$ order inside the neuronal layer. It computes

$$e_{bht} = \mathrm{ReLU}(\lambda_h u_{bht}), \qquad p_{bht} = -\operatorname{expm1}(-e_{bht}), \tag{50}$$

where the second form is numerically stable for small $e$. Training and evaluation both sample hard events and then apply Eq. (46). The sampled tensor is detached before the straight-through expression, so gradients cannot enter through sampling noise.

The implementation passes $\tau = 1$ to the categorical sampling routine. For any common $\tau > 0$,

$$\arg\max_c \frac{\log p_c + g_c}{\tau} = \arg\max_c [\log p_c + g_c], \tag{51}$$

so the hard category is invariant to this common scaling. The backward pass follows $p$ directly and is also independent of this temperature.

After sampling, each layer applies sequence-shared dropout, a pointwise $H \to 2H$ channel projection, and a gated linear unit that returns $H$ channels. The resulting continuous-valued branch is added to the residual stream. Appendices E and I specify these operations and their computational accounting.

## E Complete Experimental Settings

This appendix records the implementation and evaluation protocol used for the five Long Range Arena (LRA) tasks. The PND implementation follows the dynamics and sampling equations in Appendices A and D. Each network contains six unidirectional residual blocks and preserves the input sequence length throughout the backbone.

### E.1 Model architecture

The external tensor layout is $(B, L, H)$, where $B$, $L$, and $H$ denote batch size, sequence length, and hidden width. Each PND layer temporarily transposes the tensor to $(B, H, L)$. Its expanded linear dynamics produces the scalar membrane readout described in Section 3.2. The implementation then computes

$$e_{bht} = \mathrm{ReLU}(\lambda_h u_{bht}), \qquad p_{bht} = -\operatorname{expm1}(-e_{bht}), \tag{52}$$

and clips $p$ to $[10^{-7}, 1 - 10^{-7}]$ only when constructing log probabilities. Training and inference both use hard binary Gumbel-Max samples. The straight-through tensor is

$$\widetilde{s} = s - \mathrm{sg}(p) + p, \tag{53}$$

so its forward values are exactly binary. Appendix D gives the distribution and gradient derivations.

The output path of every PND layer is

$$\text{hard spike} \to \text{DropoutNd} \to \mathrm{Conv1d}(H, 2H, 1) \to \mathrm{GLU} \to \text{residual branch}. \tag{54}$$

The channel projection and gated linear unit return an $H$-channel sequence. Sequence-shared dropout masks are used both inside the layer and on the residual branch. All experiments use batch normalization. AAN, IMDB, and Pathfinder use pre-normalization; ListOps and sCIFAR use post-normalization. Only sCIFAR uses nonzero dropout, with rate $0.1$.

### E.2 Datasets and preprocessing

**ListOps.** We use the official `basic_train`, `basic_val`, and `basic_test` splits, containing 96,000, 2,000, and 2,000 examples. The tokenizer replaces the closing bracket by a dedicated token, removes parentheses, and splits on whitespace. The vocabulary is built only from the training split and includes padding, unknown, and end-of-sequence tokens. An end-of-sequence token is appended and the total length is capped at 2,048. Batches use dynamic right padding, and sequence pooling excludes padded positions through the recorded lengths.

**IMDB.** We use character-level inputs with a maximum length of 4,096 and append an end-of-sequence token. Characters occurring fewer than 15 times in the training set are excluded from the vocabulary. Dynamic padding and length-aware mean pooling are used. The official training and test splits each contain 25,000 examples.

**AAN.** We use the official train, validation, and test pair files with 147,086, 18,090, and 17,437 document pairs. Each document is character-tokenized, terminated by an end-of-sequence token, and capped at 4,000 positions. The shared vocabulary has 98 symbols. The two documents are padded to a common batch length and concatenated along the batch dimension before the shared backbone. The retrieval decoder forms $[h_1, h_2, h_1 - h_2, h_1 \odot h_2]$ from the two pooled representations and applies a GELU classification head.

**Sequential CIFAR-10.** Each image is converted to grayscale, transformed to a tensor, and normalized using mean $122.6/255$ and standard deviation $61.0/255$. Row-major flattening yields a sequence of length 1,024 with one scalar per position. We use no permutation or image augmentation. The official training set is divided with seed 42 into 45,000 training and 5,000 validation examples; the official test set contains 10,000 examples.

**Pathfinder-32.** We use the `curv_contour_length_14` subset. Grayscale $32 \times 32$ images are flattened in row-major order to length 1,024 without centering or spatial pooling. The 200,000 examples are split with seed 42 into 160,000 training, 20,000 validation, and 20,000 test examples.

### E.3 Training configuration

Table 4 gives all task-specific settings. The five tasks share six residual blocks, $\Delta t \in [10^{-3}, 10^{-1}]$, hard sampling with the compatibility setting $\tau = 1$, and a separate learning rate of $10^{-3}$ for temporal-dynamics parameters. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$ and cosine decay after linear warmup. Biases, normalization parameters, and temporal-dynamics parameters receive no weight decay. Gradient accumulation is one, gradient clipping is disabled, and training uses 32-bit precision on one GPU. The loss is cross entropy. Runs save a best-evaluation checkpoint and the last checkpoint. Checkpoint selection uses validation accuracy for ListOps, AAN, sCIFAR, and Pathfinder; IMDB has no separate validation split and reports the best test accuracy recorded during training, with hyperparameters adjusted using the same evaluation.

ListOps, IMDB, and sCIFAR use mean sequence pooling followed by a linear output layer; padding is excluded for ListOps and IMDB. Pathfinder uses fixed-length mean pooling. AAN uses the paired retrieval decoder described above. Data loaders use four workers, pinned memory, and `drop_last=true`.

### E.4 Component comparisons

The LRA component comparisons retain the task-specific backbone and training configurations. Threshold-SG uses the expanded membrane dynamics with a deterministic threshold and a tanh-based surrogate derivative. It has no spike-triggered reset, preserving parallel evaluation of the same linear integration. PND w/o LE replaces the expanded dynamics with a scalar leaky recurrence and retains exponential firing probabilities and probability-path straight-through training. Threshold-SG therefore changes the event rule and its training derivative together. The repeated-inference experiments use independently selected fixed checkpoints as specified in Appendix G.

Table 4: Task-specific architecture and optimization settings.

| Setting | ListOps | IMDB | AAN | sCIFAR | Pathfinder |
|---|---|---|---|---|---|
| Maximum length | 2,048 | 4,096 | 4,000 | 1,024 | 1,024 |
| Input encoder | embedding | embedding | embedding | linear | linear |
| Hidden width $H$ | 256 | 256 | 256 | 512 | 256 |
| Expanded state $d_{\text{state}}$ | 16 | 32 | 8 | 64 | 64 |
| Residual blocks | 6 | 6 | 6 | 6 | 6 |
| Pre-normalization | no | yes | yes | no | yes |
| Dropout | 0 | 0 | 0 | 0.1 | 0 |
| Batch size | 32 | 16 | 16 | 50 | 64 |
| Epochs | 80 | 32 | 30 | 200 | 200 |
| Seed | 3333 | 3333 | 3333 | 2222 | 3333 |
| Base learning rate | .01 | .01 | .01 | .01 | .004 |
| Weight decay | .05 | .05 | .05 | .05 | .05 |
| Warmup steps | 24,000 | 5,000 | 30,000 | 18,000 | 50,000 |
| Scheduled steps | 240,000 | 50,000 | 300,000 | 180,000 | 500,000 |

### E.5 Software and hardware

The recorded environment uses Python 3.10.20, PyTorch 2.7.1 with CUDA 12.8, Torchvision 0.22.1, cuDNN 9.7.1, PyTorch Lightning 2.5.6, Hydra 1.3.2, OmegaConf 2.3.1, Hugging Face Datasets 5.0.1, and NumPy 2.2.6. Training uses a single NVIDIA GeForce RTX 5090 with 32 GB memory; distributed data parallelism is disabled.

### E.6 Firing-rate measurement

A forward pre-hook is attached to each layer's internal dropout module. It therefore observes the hard spike tensor after probability-path straight-through sampling and before dropout and the $1 \times 1$ channel projection. For layer $\ell$, the measured firing rate is

$$r_\ell = \frac{\sum_{b,h,t} s^{(\ell)}_{bht}}{\text{numel}(s^{(\ell)})}, \tag{55}$$

and the global rate is element-weighted,

$$r_{\text{global}} = \frac{\sum_\ell \sum_{b,h,t} s^{(\ell)}_{bht}}{\sum_\ell \text{numel}(s^{(\ell)})}. \tag{56}$$

All recorded tensors satisfy $s \in \{0, 1\}$ exactly.

One complete evaluation pass aggregates the validation and test loaders because the accumulator is not reset between them. With `drop_last=true`, the measured examples per split are 1,984 for ListOps; 24,992 for each IMDB loader; 18,080 and 17,424 AAN pairs; 5,000 and 10,000 sCIFAR images; and 19,968 for each Pathfinder split. Text-task counts include padded positions. Each AAN pair contributes two temporal sequences. For IMDB, the same official test split is traversed twice because both loaders reference it. The reported rates describe hard-event density in the executed tensors, including padding and both evaluation traversals where applicable.

## F Additional Expanded-Dynamics Analysis

This appendix extends the mechanism analysis in Section 4.1. Figures 2(a–c) and the mode-level statistics use a dedicated sCIFAR-1024 model with four residual layers, width 256, and $d_{\text{state}} = 128$.

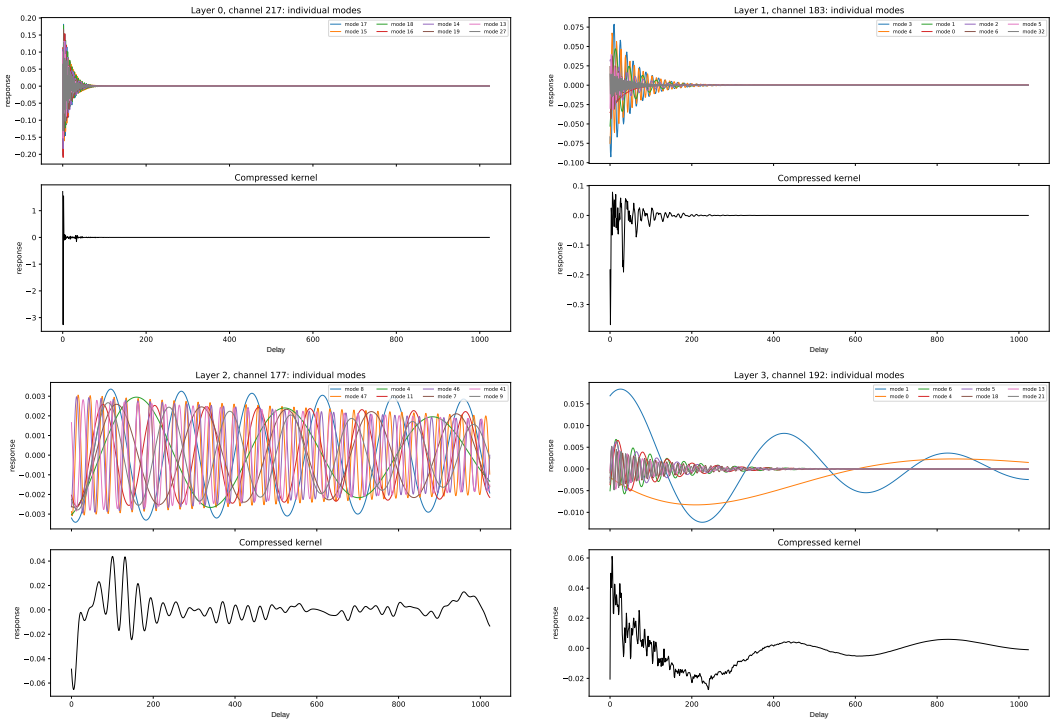



Figure 4: Representative expanded temporal responses from Layers 0–3. Each example is a fixed random channel; the eight displayed modes are selected by response energy within that channel. Aggregate analyses below include all channels and modes.

Each neuron has 64 stored conjugate pairs, producing 64 real temporal contributions as defined in Eq. (21). The model uses layer normalization, dropout $0.2$, batch size 64, and training seed 2222. It is trained for 400 epochs with learning rate $0.01$, temporal-parameter learning rate $0.001$, and weight decay $0.01$; the analyzed checkpoint is selected by validation accuracy. Statistics include all 256 channels in each of the four layers. This analysis configuration is separate from the six-layer LRA configurations in Appendix E.

### F.1 Representative responses across layers

Figure 4 shows one preselected random channel from each analyzed layer. Each panel displays the eight modes with the largest response energy within that channel together with their compressed kernel. The responses cover different decay scales, signs, and oscillation patterns. Their compression produces kernels whose shape cannot be obtained by repeating a single geometric response.

### F.2 Mode correlation and redundancy

For modes $\boldsymbol{k}_r$ and $\boldsymbol{k}_q$, we compute their centered Pearson correlation $\rho_{rq}$ and summarize $|\rho_{rq}|$ over all off-diagonal pairs. Figure 5(a) averages the absolute correlation matrix over 256 channels per layer. The diagonal remains one by definition, whereas most off-diagonal entries occupy the low-correlation range. Figure 5(b) shows the complete off-diagonal distributions on a logarithmic density scale. Their medians are $0.0180$, $0.0213$, $0.0249$, and $0.0270$ from Layers 0 to 3. The means remain between $0.0528$ and $0.0656$ despite a small tail of highly correlated pairs.

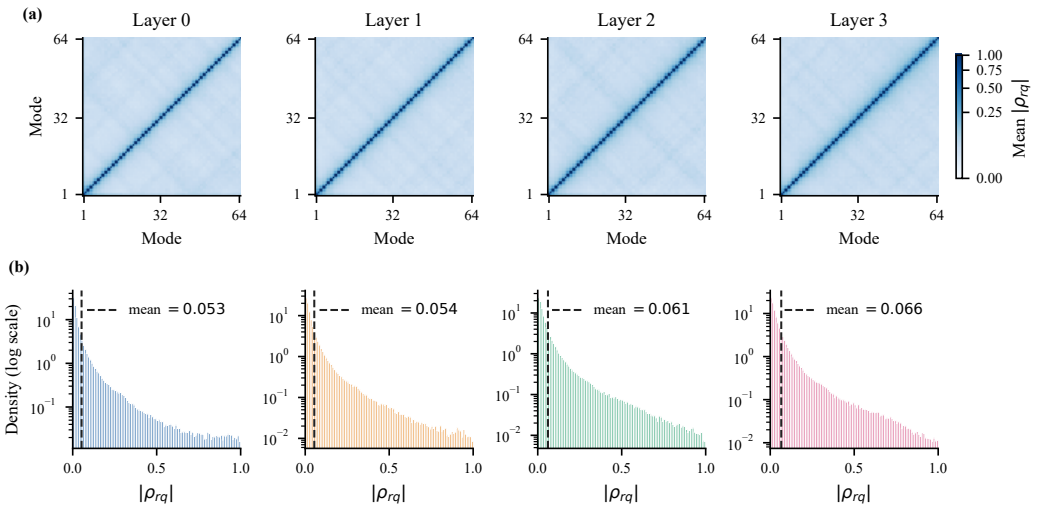



Figure 5: Complete temporal-mode correlation analysis. (a) Absolute correlation matrices averaged over all channels. A power-scaled color map preserves the structure of low off-diagonal values. (b) Distributions of all off-diagonal absolute correlations. Dashed lines denote layer means.

Table 5: Layer-wise mode redundancy and effective rank. Values are mean ± standard deviation over neurons, except correlation, which aggregates all off-diagonal mode pairs.

| **Metric** | **Layer 0** | **Layer 1** | **Layer 2** | **Layer 3** |
|---|---|---|---|---|
| Mean $\lvert\rho_{rq}\rvert$, $r \neq q$ | .0528 | .0540 | .0609 | .0656 |
| Raw effective rank | $45.36 \pm 11.21$ | $41.53 \pm 12.43$ | $37.38 \pm 13.31$ | $33.21 \pm 13.84$ |
| Unit-column effective rank | $57.11 \pm 8.31$ | $56.84 \pm 7.18$ | $55.74 \pm 7.46$ | $54.43 \pm 7.76$ |
| Unit-column rank / 64 | .892 | .888 | .871 | .851 |

### F.3 Effective rank and singular spectra

For each neuron, let $\boldsymbol{K} \in \mathbb{R}^{L\times q}$ collect its real pair contributions $K_{\tau r} = 2\,\mathrm{Re}(c_r b_r \alpha_r^\tau)$, with $q = 64$ for this analysis. Given singular values $\{\sigma_j\}$, we define

$$q_j = \frac{\sigma_j}{\sum_k \sigma_k}, \qquad r_{\text{eff}} = \exp\left(-\sum_j q_j \log(q_j + \epsilon)\right). \tag{57}$$

We report both the original matrix and a matrix whose columns are normalized to unit Euclidean norm. The first retains learned amplitude allocation; the second measures diversity of temporal shape independently of column amplitude. Effective rank approaches $q$ when the singular values are evenly distributed and approaches one when a single direction dominates.

Figure 6(a,b) gives the median singular spectra and interquartile ranges over all neurons. The original spectra become progressively more concentrated with depth. After column normalization, their tails remain broad across all four layers. This distinction appears directly in Figure 6(c): mean raw rank decreases from $45.36$ to $33.21$, while mean unit-column rank remains between $54.43$ and $57.11$ out of 64. Deeper layers therefore allocate response amplitude more selectively while retaining a large set of distinct response shapes.

### F.4 Departure from a single geometric response

A primitive response satisfies $K_{t+1} = \alpha K_t$ for one constant $\alpha$. For each learned compressed kernel, the recorded analysis fits a nonnegative decay coefficient with numerical regularization

$$\widehat{\alpha} = \mathrm{clip}_{[0,1]}\left(\frac{\langle \boldsymbol{K}_{1:L-1}, \boldsymbol{K}_{2:L}\rangle}{\|\boldsymbol{K}_{1:L-1}\|_2^2 + \epsilon}\right), \qquad \epsilon = 10^{-7}. \tag{58}$$

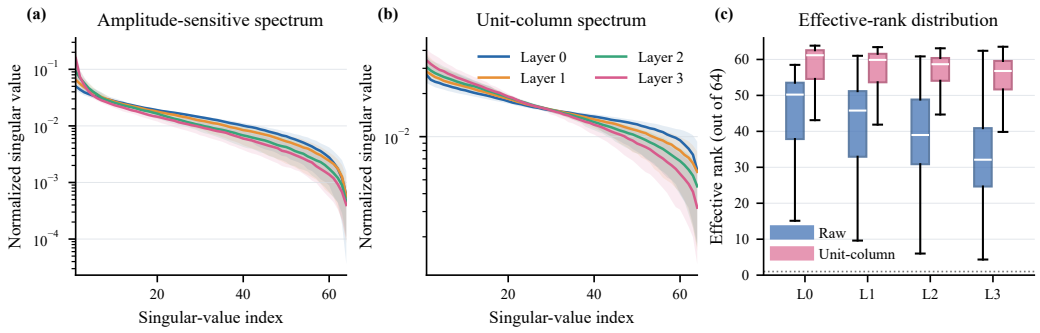



Figure 6: Singular spectra and effective rank over 256 neurons per layer. (a) Spectra of the original basis matrices. (b) Spectra after normalizing every temporal-mode column to unit norm. Lines show medians and bands show interquartile ranges. (c) Effective-rank distributions; the dotted line marks the rank-one duplicated-mode reference.

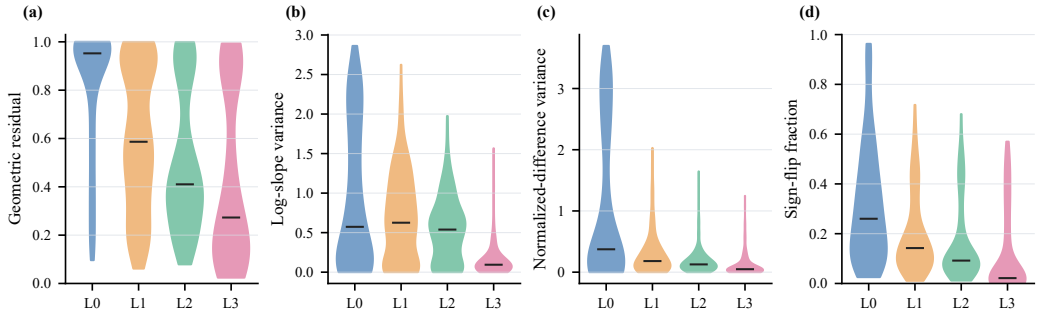



Figure 7: Distributions of compressed-response statistics over all analyzed neurons. Black bars denote medians. (a) Residual from the regularized one-step decay fit. (b) Variance of local log-magnitude slopes. (c) Variance of normalized first differences. (d) Fraction of adjacent sign changes.

We measure the normalized residual

$$E_{\text{geo}} = \frac{\|\boldsymbol{K}_{2:L} - \widehat{\alpha}\boldsymbol{K}_{1:L-1}\|_2}{\|\boldsymbol{K}_{2:L}\|_2 + \epsilon}. \tag{59}$$

Its means are $0.816$, $0.605$, $0.505$, and $0.425$ across Layers 0–3. The distributions in Figure 7(a) summarize mismatch to this fitted decay across the model. The regularizer also affects near-zero kernels, so this statistic measures the implemented fit alongside the response-shape and rank analyses.

We also measure the variance of local log-magnitude slopes, the variance of normalized first differences, and the fraction of adjacent sign changes. Figure 7(b–d) shows a shift from more oscillatory shallow responses toward more concentrated deep responses. The mean sign-flip fraction decreases from $0.316$ in Layer 0 to $0.120$ in Layer 3, while the geometric residual remains nonzero at every depth. These measurements describe how response shape changes across the network and complement the basis-rank analysis.

### F.5 Expansion-capacity sweep

Table 6 gives the complete sCIFAR-1024 expansion sweep shown compactly in Figure 2(d). The model uses four layers with hidden width 256 and is trained for 400 epochs. Increasing $d_{\text{state}}$ raises the number of complex-paired empirical modes and the parameter count. Accuracy improves throughout the sweep and approaches a plateau between 64 and 128 state dimensions. The result aligns with the learned-rank measurements: useful response capacity grows with the expansion, while gains diminish once the model already retains many temporal directions.

Table 6: Expansion-capacity sweep on sCIFAR-1024.

| $d_{\text{state}}$ | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|
| Complex-paired modes | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| Parameters | 538,634 | 542,730 | 550,922 | 567,306 | 600,074 | 665,610 | 796,682 |
| Best validation (%) | 69.26 | 73.88 | 79.20 | 82.74 | 84.80 | 85.90 | 86.08 |
| Test accuracy (%) | 67.20 | 72.35 | 78.26 | 81.68 | 83.76 | 85.33 | 85.42 |

# G Additional Probabilistic-Firing Analysis

This appendix separates two properties of probabilistic firing: variation across repeated event samples and the density of the hard events passed to the next layer. The first is measured by repeatedly evaluating one checkpoint while changing only the sampling randomness. The second is measured directly at the hard-spike tensor described in Appendix E.6.

## G.1 Repeated stochastic inference

The five-task stability study uses an earlier fixed checkpoint for each task, evaluated ten times on its test split. It is a separate experiment from the benchmark runs in Table 1; the means below characterize those earlier checkpoints. For $R = 10$ predictions $\{\widehat{y}_i^{(r)}\}_{r=1}^{R}$ of sample $i$, we define majority agreement as

$$A_i = \frac{1}{R} \max_c \sum_{r=1}^{R} \mathbb{I}[\widehat{y}_i^{(r)} = c]. \tag{60}$$

The reported agreement is the mean of $A_i$ over samples. This statistic distinguishes aggregate accuracy stability from sample-level decision stability.

Table 7 gives the available task-level statistics. Accuracy varies little across complete inference passes. Agreement resolves a second pattern: sCIFAR, AAN, and Pathfinder retain the same majority label for most samples, while IMDB and ListOps contain larger sets of sampling-sensitive examples. Chance-corrected Fleiss $\kappa$ also separates the higher-agreement sCIFAR, AAN, and Pathfinder results from IMDB and ListOps.

Across the five tasks, the accuracy standard deviation ranges from $0.062$ to $0.503$ percentage points. Pathfinder has the smallest aggregate variation, followed by AAN and sCIFAR. ListOps has the lowest mean agreement and the largest accuracy variation. Probabilistic firing therefore preserves stable aggregate behavior while leaving more sample-level variation on the tasks with lower agreement.

For $n$ test samples and vote counts $n_{ic}$ for class $c$, Fleiss agreement is defined by

$$\bar{P} = \frac{1}{nR(R-1)} \sum_{i,c} n_{ic}(n_{ic}-1), \quad \pi_c = \frac{1}{nR} \sum_i n_{ic}, \quad \kappa = \frac{\bar{P} - \sum_c \pi_c^2}{1 - \sum_c \pi_c^2}. \tag{61}$$

It measures excess pairwise agreement relative to the class frequencies across all passes, whereas majority agreement measures the largest vote share for each sample.

Figure 8 is a separate diagnostic on the four-layer sCIFAR analysis model in Appendix F. It uses ten test passes of its validation-selected checkpoint and has mean accuracy $85.324\%$. For the confidence-variance panel, the class is fixed to each sample's majority prediction across the ten passes. The plotted quantity is the population variance of that class's softmax probability over passes. Most samples retain the same predicted label, while a smaller group varies with the sampled events.

## G.2 Layer-wise hard-event density

Table 8 gives firing rates measured immediately before dropout and the pointwise output projection. The distribution over depth depends on the input structure. ListOps concentrates events in its first two layers, which account for approximately $88.1\%$ of its hard events. AAN concentrates approximately $86.8\%$ in Layer 0. IMDB remains below $3\%$ in every layer. The visual tasks distribute their events more broadly: sCIFAR spans $6.89$–$12.32\%$, and Pathfinder reaches its maximum in Layer 1.

Table 7: Ten-pass stochastic-inference statistics for a fixed checkpoint per task. Accuracy standard deviation is in percentage points. Each row summarizes the same ten test passes for the corresponding earlier checkpoint.

| Metric | sCIFAR | AAN | IMDB | ListOps | Pathfinder |
|---|---|---|---|---|---|
| Accuracy mean (%) | 87.812 | 86.947 | 79.667 | 53.558 | 92.215 |
| Accuracy std. (pp) | .148 | .107 | .162 | .503 | .062 |
| Mean agreement (%) | 96.54 | 97.12 | 92.48 | 84.27 | 96.68 |
| Fleiss $\kappa$ | .942 | .911 | .771 | .725 | .898 |

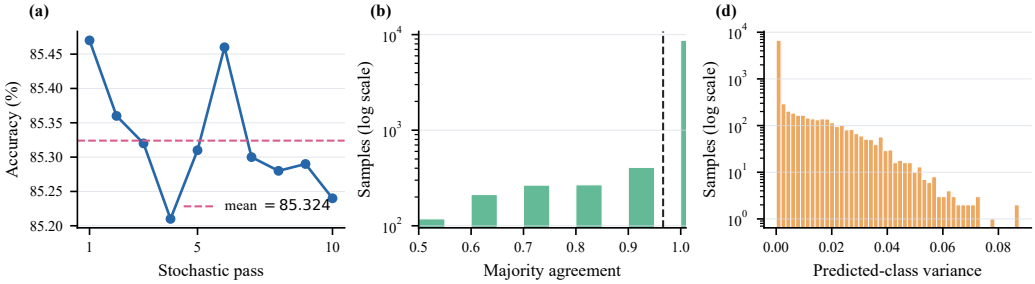



Figure 8: Separate repeated-inference diagnostic on the four-layer sCIFAR-1024 analysis model. (a) Accuracy over ten stochastic passes. (b) Sample-level majority agreement. (d) Variance of the predicted-class confidence across passes.

Table 9 reports the underlying scale of the measurement. Event counts are reconstructed from the stored floating-point rate and integer denominator. The corresponding zero-event fractions range from 90.60% on sCIFAR to 98.87% on IMDB. These values quantify the sparsity presented to the post-spike projection; Appendix I converts them into an operation-level energy estimate with the remaining model operations stated separately.

Table 8: Layer-wise hard firing rate (%) under the evaluation protocol in Appendix E.6. Global rates are weighted by the number of recorded tensor elements.

| Task | L0 | L1 | L2 | L3 | L4 | L5 | Global |
|---|---|---|---|---|---|---|---|
| ListOps | 18.834 | 23.726 | 1.138 | 1.691 | 1.655 | 1.290 | **8.056** |
| IMDB | 2.909 | .723 | 1.185 | .499 | .870 | .581 | **1.128** |
| AAN | 8.563 | .371 | .435 | .065 | .330 | .104 | **1.645** |
| sCIFAR | 7.158 | 12.319 | 7.553 | 10.416 | 6.886 | 12.088 | **9.403** |
| Pathfinder | 6.289 | 9.077 | 5.590 | 5.893 | 1.634 | 4.944 | **5.571** |

Table 9: Reconstructed hard-event counts over the complete evaluation pass. AAN counts both document sequences in each pair; text-task totals include padding.

| Task | Elements over six layers | Hard events | Zero-event fraction (%) |
|---|---|---|---|
| ListOps | 11,387,535,360 | 917,332,796 | 91.944 |
| IMDB | 249,807,421,440 | 2,817,405,580 | 98.872 |
| AAN | 436,273,152,000 | 7,174,817,675 | 98.355 |
| sCIFAR | 47,185,920,000 | 4,437,120,567 | 90.597 |
| Pathfinder | 62,813,896,704 | 3,499,452,295 | 94.429 |

# H Additional Component Experiments

This appendix extends the component analysis beyond the long-range benchmarks in Table 2. It first tests dimension expansion on sequential static, neuromorphic visual, and speech inputs. It then visualizes how layer-level hard-event responses evolve during training on sequential MNIST.

## H.1 Expansion across input modalities

Table 10 compares PND with PND w/o LE on four additional datasets. PND w/o LE keeps the exponential event model and hard probabilistic sampling but replaces dimension-expanded dynamics with a primitive single-mode LIF trajectory. Expansion improves accuracy on every dataset, including event-camera inputs and raw speech. The consistent direction across modalities supports the role of expansion in temporal integration.

These results are retained from an earlier PND implementation with dimension expansion and exponential hard spike sampling. They form a separate experiment from the current LRA comparisons and the sMNIST confusion analysis in Figure 3. The recorded configurations use four residual layers with width 256 and expanded dimension 128 for NMNIST and DVS Gesture, and six layers with width 128 and expanded dimension 64 for SpeechCommands. SpeechCommands inputs are waveform sequences sampled at 16 kHz. These results complement the LRA ablation. The improvement is visible when an image is unfolded into a long pixel sequence, when the input originates from a neuromorphic sensor, and when the input is a waveform. In each case, multiple learnable temporal responses give the event generator a richer membrane trajectory than the primitive one-mode recurrence.

## H.2 Temporal evolution on sequential MNIST

For sMNIST, each $28 \times 28$ image is flattened into 784 steps. At each step, we average the hard firing events across neurons within a layer and reshape the resulting 784-point sequence to the original image grid. The heatmaps show where in the sequential input each layer emits events.

Figure 9 compares Layers 1 and 2 at initialization and after 14 epochs for all ten labels. At initialization, the response often follows row boundaries and broad intensity transitions. Training organizes the responses around digit-specific contours. Layer 1 retains a broader outline, while Layer 2 suppresses many diffuse responses and preserves more localized transitions. These event-activity patterns give a qualitative view of learned temporal selectivity. The separate confusion analysis in Figure 3 measures the resulting classification errors.

Table 10: Effect of dimension-expanded dynamics on additional sequential and neuromorphic datasets. Accuracy is reported in percent.

| Dataset | PND w/o LE | PND |
|---|---|---|
| NMNIST | 86.408 | **98.115** |
| sMNIST | 93.60 | **99.20** |
| DVS Gesture | 71.18 | **96.53** |
| SpeechCommands | 73.05 | **95.80** |

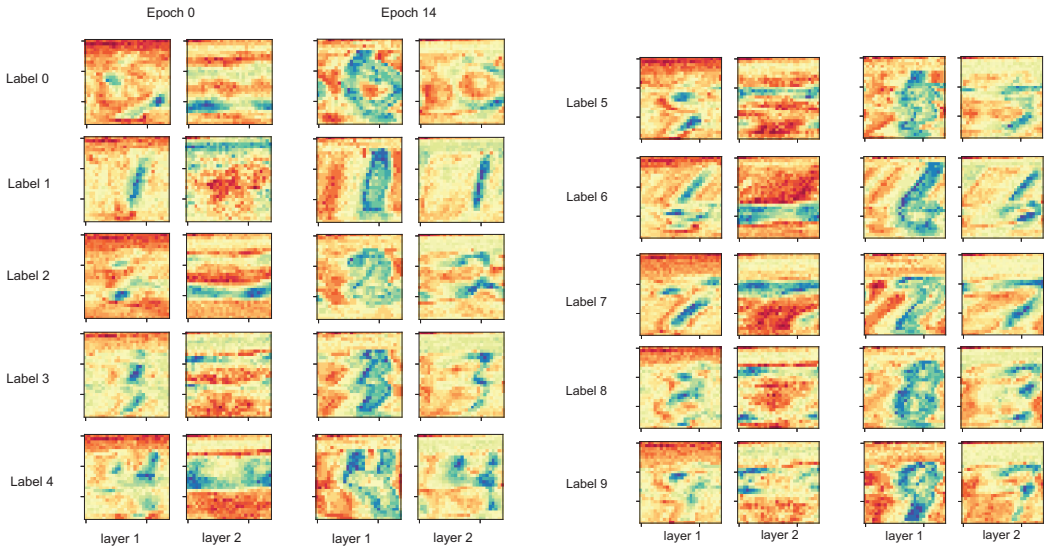



Figure 9: Temporal-response heatmaps for all sMNIST labels. The left pair of columns in each panel shows initialization, and the right pair shows epoch 14; within each pair, Layers 1 and 2 are displayed from left to right.

# I Computational and Energy Analysis

This appendix separates three computational properties of PND: parallel evaluation of its linear expanded temporal dynamics, event density at the inter-layer interface, and the arithmetic associated with the pointwise output projection. These quantities measure different parts of the implemented layer and are reported separately.

## I.1 Parallel and recurrent evaluation

The expanded recurrence can be evaluated either by iterating over time or by constructing the equivalent kernel in Appendix A. For a layer with hidden width $H$, expansion size $m$, and sequence length $L$, direct recurrence performs $O(BHmL)$ arithmetic for batch size $B$ with a sequential dependency spanning $L$ steps. Kernel construction also costs $O(HmL)$, followed by $O(BHL \log L)$ FFT convolution. The latter formulation evaluates the sequence in parallel once the kernel is available.

The timing comparison in Table 11 was recorded on an NVIDIA RTX 4090 using the IMDB pipeline. The recurrent version iterates the expanded state update, while the parallel version evaluates the convolutional kernel. Data and batch settings were matched between implementations. The values are averages of iteration rates displayed in the run progress logs; an iteration refers to the loop iteration reported by those logs. The retained record does not distinguish the training and evaluation timing phases, so these values describe the recorded execution rates rather than an isolated forward-pass latency.

Table 11: Average iteration speed of recurrent and parallel expanded dynamics on IMDB sequences. Parallel seconds per iteration are the reciprocals of the recorded throughput.

| **Sequence length** | **1K** | **2K** | **3K** | **4K** |
|---|---|---|---|---|
| Recurrent (s/it) | 4.65 | 9.25 | 14.32 | 17.36 |
| Parallel (it/s) | 12.22 | 11.40 | 10.67 | 8.91 |
| Parallel (s/it) | .0818 | .0877 | .0937 | .1122 |
| Acceleration ($\times$) | 56.82 | 105.45 | 152.79 | 154.68 |

The recurrent time grows approximately with sequence length. The parallel time changes more slowly over the same range, so the measured acceleration rises from $56.82\times$ at 1K to $154.68\times$ at 4K. This result reflects the removal of the step-wise dependency in temporal integration.

### I.2 Parameter and memory scaling

For diagonal expanded dynamics, the number of temporal parameters grows linearly as $O(Hm)$ per layer. The implemented post-spike projection has $2H^2 + 2H$ weights and biases before the gated linear unit. Across $N$ blocks, these terms scale as $O(NHm)$ and $O(NH^2)$, respectively. The input encoder, normalization, and task decoder add task-specific parameters.

The parallel form stores one length-$L$ kernel per channel and the sequence activations, giving $O(HL)$ kernel storage and $O(BHL)$ activation storage before FFT work buffers. For streaming inference, direct recurrence stores an $m$-dimensional state per channel and batch element, $O(BHm)$. Training additionally stores or recomputes intermediate states for backpropagation. The implementation chooses the parallel form to reduce sequential execution time on long inputs.

### I.3 Operation-level energy model

The hard spike tensor enters a pointwise projection from $H$ input channels to $2H$ output channels. Let $Q_\ell = \sum_{b,h,t} s_{bht}^{(\ell)}$ be the number of firing events at layer $\ell$, and let $M_\ell$ be the number of elements in that layer's spike tensor. Under event-driven accumulation, each input event contributes to $2H$ output channels. The projection therefore requires

$$N_{\mathrm{AC}} = 2H \sum_{\ell=1}^{N} Q_\ell \tag{62}$$

accumulate operations over an evaluation pass. A dense implementation of the same projection requires

$$N_{\mathrm{MAC}}^{\mathrm{dense}} = 2H \sum_{\ell=1}^{N} M_\ell \tag{63}$$

multiply–accumulate operations. We use the widely adopted 45 nm FP32 arithmetic estimates $E_{\mathrm{AC}} = 0.9\,\mathrm{pJ}$ and $E_{\mathrm{MAC}} = 4.6\,\mathrm{pJ}$ (Horowitz, 2014; Zhou et al., 2023). For $n$ evaluated examples,

$$E_{\mathrm{event}} = \frac{N_{\mathrm{AC}} E_{\mathrm{AC}}}{n}, \qquad E_{\mathrm{dense}} = \frac{N_{\mathrm{MAC}}^{\mathrm{dense}} E_{\mathrm{MAC}}}{n}. \tag{64}$$

Table 12 applies these equations to the reconstructed event counts and recorded element counts in Table 9. The example denominator follows the evaluation protocol: 3,968 ListOps sequences, 49,984 IMDB sequence evaluations, 35,504 AAN document pairs, 15,000 sCIFAR images, and 39,936 Pathfinder images. An AAN example includes both documents in its pair.

The event-driven estimates range from $0.026$ to $0.273\,\mathrm{mJ}$ per evaluated example. Their task ordering depends on hidden width, executed sequence length, padding, paired inputs, and layer-wise event counts, rather than on the global firing percentage alone. For example, sCIFAR has the largest estimate because its hidden width is 512, while AAN processes two padded document sequences per example.

Table 12: Operation and energy estimates for the six post-spike $H \to 2H$ pointwise projections. GOp denotes $10^9$ operations per evaluated example. Energy uses the 45 nm FP32 arithmetic model in Eq. (64).

| **Task** | $H$ | **Firing (%)** | **Event AC (GOp)** | **Event energy (mJ)** | **Dense MAC (GOp)** | **Dense energy (mJ)** |
|---|---|---|---|---|---|---|
| ListOps | 256 | 8.056 | .118 | .107 | 1.469 | 6.759 |
| IMDB | 256 | 1.128 | .029 | .026 | 2.559 | 11.771 |
| AAN | 256 | 1.645 | .103 | .093 | 6.291 | 28.941 |
| sCIFAR | 512 | 9.403 | .303 | .273 | 3.221 | 14.818 |
| Pathfinder | 256 | 5.571 | .045 | .040 | .805 | 3.704 |

Equations (62)–(64) describe the arithmetic opportunity exposed by binary sparse events. The current PyTorch `Conv1d` executes as a dense GPU operator and does not automatically skip zero inputs. The estimate covers the weighted sums of the six post-spike pointwise projections. Kernel construction and FFT, probability computation, Gumbel sampling, GLU gating, normalization, residual addition, memory access, encoders, and decoders remain separate operations. Consequently, Table 12 is an operation-level projection estimate for an event-driven implementation, while Table 11 is the measured runtime result of the present GPU implementation.