THRESHOLDS, NOT NORMALIZATION: A HOMEOSTATIC SPIKING LAYER AT PARITY WITHOUT THE OVERHEAD

Anonymous authors
Paper under double-blind review

ABSTRACT

Spiking neural networks processing long event streams frequently rely on normalization to match input currents to a fixed firing threshold. However, this reliance introduces batch reductions, floating-point operations, and per-timestep statistics that neuromorphic hardware struggles to accommodate efficiently. We propose adapting the threshold itself instead of the incoming current. We introduce HomeoLayer, a leaky integrate-and-fire layer featuring ℓ_2-normalized weights, a per-feature affine transform, closed-form spike-probability gradients, and an end-to-end learned threshold controller. Pooling this controller over the batch or layer recovers the classic BatchNorm/LayerNorm dichotomy. At inference, HomeoLayer utilizes only additions and multiplications, stores no running statistics, and remains completely batch-independent. Across two architectures and four datasets (SHD, SSC, GSC, and N-MNIST), HomeoLayer achieves parity with per-timestep batch normalization while maintaining a fraction of the per-neuron state. Furthermore, learning the firing rate target eliminates the need for manual threshold tuning. Transplanting the controller into existing adaptive models (C-SiLIF and RadLIF) successfully replaces native normalization with a marginal accuracy cost. This work is open-sourced at `https://anonymous.4open.science/r/snn_homeolayer-E003`.

1 INTRODUCTION

Spiking neural networks (SNNs) trained on long event streams routinely incorporate normalization layers. Spiking-specific formulations such as BNTT (Kim & Panda, 2021) and tdBN (Zheng et al., 2021) operate on pre-synaptic currents, MPBN (Guo et al., 2023) normalizes the membrane potential directly, and state-of-the-art adaptive models on long-stream audio benchmarks—including RadLIF (Bittar & Garner, 2022), SiLIF, and C-SiLIF (Fabre et al., 2026)—all retain batch normalization in their canonical implementations.

Despite its ubiquity, normalization incurs severe computational and architectural penalties. Standard batch normalization and its temporal variants compute the mean and variance across the batch for every feature at every timestep, introducing three distinct bottlenecks:

- **Batch coupling:** Dynamic batch statistics make inference on an individual sample contingent on the rest of the batch.
- **Floating-point arithmetic:** Calculating standard deviations and reciprocal scalings requires square root and division operations that intrinsic neuron dynamics never naturally require.
- **State overhead:** Maintaining temporal statistics creates substantial memory pressure; BNTT, for example, stores separate running statistics at each timestep, amounting to numerous values per neuron.

Furthermore, static batch statistics can only be folded into synaptic weights at deployment if the parameters remain strictly time-invariant. Time-dependent variants require maintaining discrete copies of the affine parameters, and per-sample mechanisms like LayerNorm cannot be folded at all. These operational constraints clash directly with the architectural design principles of neuromorphic

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.

Mothod	Dynamic state	Update origin	Firing rate	No BPTT	Pool	Norm
HomeoLayer (ours)	Threshold θ	$\nabla(p_s - r^*)^2$	Controlled (r^*)	✓	B/L	✓
adLIF (Baronig et al., 2025)	Current w	Task loss ($\nabla_w \mathcal{L}$)	Emergent	×	×	×
ALIF (Yin et al., 2021)	Threshold θ	Spike decay	Emergent	×	×	×
LSNN (Bellec et al., 2018)	Threshold θ	Spike decay	Emergent	×	×	×
SpiKL-IP (Zhang & Li, 2019)	Gain/bias $f(I)$	$\nabla \mathrm{KL}(p \,\|\, p_{\mathrm{tgt}})$	Matched†	✓	×	×
Lateral Inh. (Liu et al., 2026)	E/I connectivity	Architecture	Emergent	×	×	✓
tdBN / BNTT / MPBN	Input drive	Moments	Uncontrolled	–	Impl.	Base

Table 1: Comparison across rate-regulation and adaptive mechanisms. †SpiKL-IP adapts input transfer functions rather than thresholds.

substrates like Loihi (Davies et al., 2018; Orchard et al., 2021), TrueNorth (Merolla et al., 2014), and SpiNNaker (Furber et al., 2014; Gonzalez et al., 2024), which prioritize integer pipelines and lack dedicated floating-point hardware.

Scope. This work specifically addresses *long event streams* and the SNN architectures dedicated to their processing. We evaluate models unrolled for up to hundreds of timesteps (e.g., on SHD, SSC, GSC, and N-MNIST), where temporal dispersion encodes the primary signal and networks must integrate evidence over long periods before reaching a decision. This continuous regime is fundamentally distinct from static-image conversions (e.g., ImageNet or CIFAR), which unroll across just a few steps as a quantization proxy. Because homeostatic threshold adaptation relies on empirical firing statistics that require adequate time to converge, short-window tasks are inherently unsuitable. Consequently, our baselines consist of models built specifically for extended temporal integration.

In conventional pipelines, the firing threshold is treated as a static hyperparameter tuned per task. We argue that the need for normalization and the sensitivity of threshold selection reflect the same underlying limitation: normalization artificially stabilizes the pre-synaptic drive so that a rigid threshold remains viable. By closing the control loop directly around the threshold, this compensatory overhead becomes redundant.

To this end, we introduce HomeoLayer, a spiking formulation that governs network dynamics via adaptive threshold homeostasis rather than input standardization. Our core contributions are as follows:

- **Mechanism:** We propose a leaky integrate-and-fire (LIF)-based spiking layer governed by a homeostatic threshold controller, derived directly from closed-form spiking probabilities. Its pooling axis perfectly recovers the BatchNorm/LayerNorm dichotomy.
- **Cost efficiency:** At inference, HomeoLayer uses only addition and multiplication, stores no running statistics, and maintains a fixed, minimal footprint per neuron regardless of sequence length.
- **Performance parity:** Trained under standard field recipes, HomeoLayer matches per-timestep BatchNorm on SHD and exceeds it on SSC, all while holding a fraction of the per-neuron state.
- **Learned targets:** The firing-rate target is learned via exact gradients, successfully eliminating time-consuming manual threshold tuning.
- **Transferability:** The threshold controller reads spikes alone and transplants directly into state-of-the-art models like C-SiLIF and RadLIF, replacing their native normalization at a cost of roughly one accuracy point.

2 Related work

Normalization in SNNs. As SNNs deepened, standard normalization techniques (Ioffe & Szegedy, 2015; Ba et al., 2016) were repurposed into temporal variants. tdBN ties pre-synaptic variance directly to the firing threshold (Zheng et al., 2021), BNTT assigns independent statistics to each timestep to track non-stationary dynamics (Kim & Panda, 2021), and MPBN standardizes membrane potential trajectories directly (Guo et al., 2023). All share our core premise: normalization exists primarily so a static threshold remains functional (Figure 1A). Instead, we adapt the threshold directly (Figure 1B).

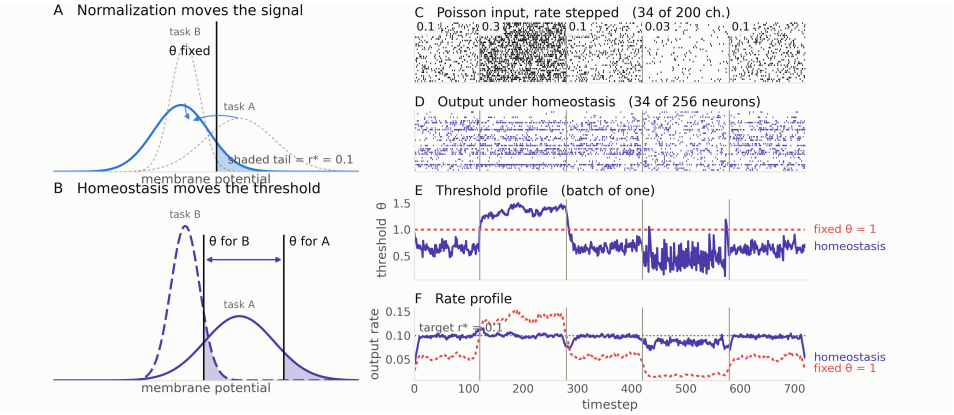


Figure 1: **Firing rate as the mass above threshold. A** Normalization standardizes each distribution so the fixed θ cuts off firing rate r^*. **B** Homeostasis moves θ to maintain r^*. **C–F** Measured on one weight-normalized LIF layer of 256 neurons, batch of one, input rate stepped $0.1 \to 0.3 \to 0.1 \to 0.03 \to 0.1$.

Adaptation and related claims. Dynamic thresholds entered deep SNNs through LSNN's spike-triggered exponential decay (Bellec et al., 2018), a decay rate later made learnable by ALIF (Yin et al., 2021). Alternatively, Baronig et al. (2025) introduce learned subthreshold adaptation currents, asserting that explicit normalization is redundant in adaptive recurrent networks. Rather than testing this claim in isolation, Section 5.5 benchmarks our transplanted threshold controller directly against the native normalization layers of two such models.

Homeostasis and alternatives. Homeostatic intrinsic plasticity typically tunes neuronal excitability toward target profiles via gradient descent (Triesch, 2005) or KL-divergence minimization (Zhang & Li, 2019), while synaptic scaling adjusts weights directly (Turrigiano & Nelson, 2004). Architecturally, Liu et al. (2026) remove normalization using balanced excitatory-inhibitory circuits, introducing structural overhead where our method requires only three scalars per layer. Conceptually, our controller extends this homeostatic tradition but differs fundamentally in its closed-form derivation and direct replacement of normalization.

Threshold controller distinctions. Table 1 highlights three core differences. First, our rule uses an explicit setpoint: rather than letting rates emerge from the input drive, we step the threshold proportionally to rate errors to actively maintain a target. Second, it is *analytically derived* via exact gradient descent on the squared rate error. Third, rate errors are pooled across a population axis: batch pooling yields one threshold per neuron, while feature pooling yields batch-independent, sample-specific thresholds. Consequently, HomeoLayer avoids backpropagating adaptation states through time (unlike LSNN, ALIF, and adLIF) and operates bidirectionally (Figure 1E–F)—lowering thresholds to rescue quiescent neurons and scaling updates by gradient density to act only where the threshold possesses active leverage.

3 Theory

We adopt a probabilistic view of SNNs. In the forward pass, spikes sampled from probability density functions (PDFs) propagate to the output layer. In the backward pass, backpropagation adjusts those PDFs so that the spikes sampled from them yield high task accuracy. Applying the backward step to the weights recovers ordinary training without surrogate gradients; applying it to the threshold yields homeostasis. The homeostatic rule is therefore derived rather than designed. Two identities are used hereafter. The spiking probability p_s equals the expected number of spikes since

$$\mathbb{E}[n_\mathrm{s}] = 1 \cdot p_\mathrm{s} + 0 \cdot (1 - p_\mathrm{s}) = p_\mathrm{s},$$

and the spike variance is

$$\mathrm{Var}(s) = p_\mathrm{s}(1 - \mathbb{E}[n_\mathrm{s}])^2 + (1 - p_\mathrm{s})\mathbb{E}[n_\mathrm{s}]^2 = p_\mathrm{s}(1 - p_\mathrm{s}). \tag{1}$$

3.1 PROBABILISTIC LIF NEURON

Forward pass. We adopt an LIF model for the probabilistic neuron. The forward pass is carried by actual spikes, together with the synaptic current (I_{syn}) and membrane potential (u_{ac}) as follows:

$$I_{\text{syn}}[t] = \beta I_{\text{syn}}[t-1] + \frac{w}{\tau_{\text{syn}}} \cdot s_{\text{in}}[t], \qquad \beta = e^{-\Delta t/\tau_{\text{syn}}}, \qquad s_{\text{in}}[t] = \delta(t - \hat{t}_i),$$

where $\hat{t}_i$ denotes the timing of the i-th input spike. The actual potential u_{ac} is given by

$$u_{\text{ac}}[t] = \alpha u_{\text{ac}}[t-1] + I_{\text{syn}}[t], \qquad \alpha = e^{-\Delta t/\tau_u},$$

which is reset to u_{rst} in the same timestep if $u_{\text{ac}} \geq \theta$.

Backward pass. The backward pass is carried by the PDFs of the membrane potential before reset (*pre-reset* potential) and after it (*post-reset* potential). Both are distinct from the actual potential u_{ac} used in the forward pass. The pre-reset potential follows a closed-form distribution,

$$p_u = \frac{1}{\sqrt{2\pi}\sigma_u} \exp\left(-\frac{(u-\mu_u)^2}{2\sigma_u^2}\right) = \frac{1}{\sigma_u}\phi\left(\frac{u-\mu_u}{\sigma_u}\right),$$

where ϕ is the standard normal density, and μ_u and σ_u are respectively the mean and standard deviation of the pre-reset potential. Each neuron therefore defines, at every timestep, a spiking probability in closed form:

$$p_{\text{s}} = P(u \geq \theta) = \frac{1}{2}\left[1 - \text{erf}\left(\frac{\theta - \mu_u}{\sigma_u\sqrt{2}}\right)\right]. \tag{2}$$

Under the soft expected-reset approximation, the post-reset mean m_u and variance q_u at timestep t are approximated as

$$m_u[t] = (1 - p_{\text{s}}[t])\mu_u[t] + p_{\text{s}}[t]u_{\text{rst}}, \qquad q_u[t] = (1 - p_{\text{s}}[t])\sigma_u^2[t] + p_{\text{s}}[t](1 - p_{\text{s}}[t])(\mu_u[t] - u_{\text{rst}})^2.$$

The pre-reset values at the following timestep are obtained from $m_u[t]$ and $q_u[t]$:

$$\mu_u[t+1] = \alpha m_u[t] + I_{\text{syn}}[t+1], \qquad \sigma_u^2[t+1] = \alpha^2 q_u[t] + \sigma_\xi^2[t],$$

where $\sigma_\xi^2[t]$ receives two contributions: noise propagated from the previous layer, and noise injected within the current one. To bound the first, **all feedforward weight matrices are ℓ_2-normalized**. The variance σ_u^2 could be set directly by injecting white noise of variance σ_ξ^2, but doing so is costly. We therefore use a surrogate variance $\sigma_{\xi,\text{surr}}^2$, treated as a hyperparameter. Unless otherwise stated, every result in this paper uses $\sigma_{\xi,\text{surr}}^2 = 0.1$ with no injected noise, so σ_u^2 is built from the surrogate variance and the propagated q_u alone.

Gradient estimation. Because the spiking probability p_{s} in Eq. 2 is differentiable, this formulation dispenses with surrogate gradients altogether.

$$\frac{\partial p_{\text{s}}}{\partial \mu_u}[t] = \frac{1}{\sigma_u[t]}\phi\left(\frac{\mu_u[t] - \theta}{\sigma_u[t]}\right). \tag{3}$$

This expression resembles the Gaussian surrogate gradient, but differs from it in being centered on the mean of the pre-reset potential rather than on the actual potential u_{ac}.

3.2 HOMEOSTASIS BY THRESHOLD ADAPTATION

Consider a layer of N neurons evaluated over a batch of size B, each with spiking probability $p_{\text{s}}[t, b, n]$. The firing rate $r[t, b, n]$ within a sliding window T_{w} is

$$r[t, b, n] = \frac{1}{T_{\text{w}}} \sum_{\tau=0}^{T_{\text{w}}} \mathbb{E}[n_{\text{s}}[t - \tau, b, n]] = \frac{1}{T_{\text{w}}} \sum_{\tau=0}^{T_{\text{w}}} p_{\text{s}}[t - \tau, b, n]. \tag{4}$$

Homeostasis requires a choice of pooling axis $A \in \{\text{BATCH}, \text{LAYER}\}$. All units along that axis share a single threshold, so pooling over the layer yields one threshold per sample and pooling over the

batch one per neuron, giving $\theta_{\mathrm{L}} \in \mathbb{R}^B$ and $\theta_{\mathrm{B}} \in \mathbb{R}^N$, respectively. The mean firing rate over the pooling axis is

$$\mathrm{mean}_{\mathrm{L}}(r[t,b,n]) \triangleq \bar{r}_{\mathrm{L}}[t] = \frac{1}{N}\sum_{n=1}^{N} r[t,b,n], \qquad \mathrm{mean}_{\mathrm{B}}(r[t,b,n]) \triangleq \bar{r}_{\mathrm{B}}[t] = \frac{1}{B}\sum_{b=1}^{B} r[t,b,n].$$

We update the threshold θ_A online to drive $\bar{r}_A$ toward a target rate $\bar{r}^*_A$, by gradient descent on the objective $\mathcal{L}_{\text{rate}} = 0.5(\bar{r}_A - \bar{r}^*_A)^2$. The update requires the gradient $\partial\bar{r}/\partial\theta$, which follows from Eq. 4 as

$$\frac{\partial \bar{r}_A}{\partial \theta_A}[t] = \frac{1}{T_{\mathrm{w}}}\sum_{\tau=0}^{T_{\mathrm{w}}} \mathrm{mean}_A\left(\frac{\partial p_{\mathrm{s}}}{\partial \theta_A}[t-\tau,b,n]\right).$$

The threshold update, $\Delta\theta_A[t]$, therefore becomes

$$\Delta\theta_A[t] = \eta_\theta(\bar{r}_A[t] - \bar{r}^*_A)\frac{1}{T_{\mathrm{w}}}\sum_{\tau=0}^{T_{\mathrm{w}}} \mathrm{mean}_A\left(\frac{\partial p_{\mathrm{s}}}{\partial \theta_A}[t-\tau,b,n]\right), \quad \frac{\partial p_{\mathrm{s}}}{\partial \theta_A} = -\frac{1}{\sigma_u}\phi\left(\frac{\mu_u - \theta}{\sigma_u}\right). \quad (5)$$

Unlike previous homeostatic schemes, this rule controls the firing rate directly against an explicit target $\bar{r}^*$. The threshold is uniquely re-initialized at the start of every run for both training and inference. Unlike BatchNorm, which carries static running statistics into inference, our threshold controller re-derives its operating point directly from the input sequence, rendering it fundamentally batch-independent.

4 HomeoLayer

While the theory in Section 3 is exact, evaluating transcendental functions (e.g., error functions) is prohibitively expensive on neuromorphic hardware. This section describes HomeoLayer: a practical module that implements the theory by estimating analytic terms directly from binary spikes for inference.

4.1 Layer Structure

HomeoLayer processes input spikes through four sequential stages (Algorithm A1):

1. **Low-pass synaptic filter.**
2. **$\boldsymbol{\ell_2}$ weight normalization:** Every input weight vector is constrained to unit ℓ_2 norm. This rigorously bounds the variance propagated from preceding layers, safely allowing σ_u to be treated as a constant surrogate variance.
3. **Per-feature affine:** We apply a learned scale and shift ($x \leftarrow \gamma x + \beta$) immediately prior to LIF integration. This offsets the rigidity of the ℓ_2 norm and restores the expressive scaling standard normalization provides, adding only $2N$ parameters per layer without running statistics.
4. **Threshold controller:** Governed by Eq. 5, it dynamically adjusts the LIF threshold using strictly spike-based estimators.

4.2 Spike-Only Estimators and Target Learning

To bypass expensive evaluations of p_{s} at inference, we approximate the spiking probability as a sigmoid: $p_{\mathrm{s}} \approx \sigma(k(\mu_u - \theta))$ with $k = 1.702/\sigma_u$. Consequently, the threshold gradient reduces to $\partial p_{\mathrm{s}}/\partial\theta \approx -k\,\mathrm{Var}(s)$, where spike variance $\mathrm{Var}(s)$ is given by Eq. 1.

An unbiased estimator for a neuron's spike variance $\mathrm{Var}(s)$ is simply the expected one-bit flip between consecutive timesteps (i.e., $s[t] \neq s[t+1]$). By replacing the analytic terms in Eq. 5 with empirical spike counts (for the rate error) and one-bit flip statistics (for the gradient density $\bar{\varphi}$), the threshold update executes using only addition and multiplication:

$$\Delta\theta_A[t] = \eta_\theta \underbrace{(\bar{r}_A[t] - \bar{r}^*_A)}_{\text{Empirical Spike Count Error}} \times \underbrace{\tfrac{1}{T_{\mathrm{w}}}\textstyle\sum_{\tau=0}^{T_{\mathrm{w}}} \mathrm{mean}_A\left(\mathrm{Var}(s[t-\tau])\right.}_{\text{1-Bit Flip Statistic } (\bar{\varphi})}.$$

A gain floor $\varphi_{\min} = 10^{-2}$ may be applied to prevent permanently silent layers from entering an absorbing state where updates stall if needed. The controller operation is detailed in Algorithm A2.

Ablation note: While the spike-flip estimator efficiently guides the threshold controller, substituting it for the exact analytic weight gradient degrades accuracy by several points on feedforward architectures. Therefore, HomeoLayer strictly uses the exact closed-form gradient (Eq. 3) for weight updates during training, isolating the spike-only estimator to the threshold controller at inference.

Learning the target. The target rate is parameterized as $r^* = r_{\min} + (1 - r_{\min})\sigma(\rho)$ and learned end-to-end via exact gradients, costing only one scalar per layer. Under both layer and batch pooling, each layer holds a single rate target r^* that are shared by all neurons and samples, so that all thresholds are driven to the single rate target.

4.3 Layer Homeostasis as a Normalization

Under layer pooling, a single threshold $\theta_{\mathrm{L}}[b]$ is shared by all N neurons for a sample, and the threshold controller drives it until $\bar{r}_{\mathrm{L}} = r^*$. The spiking probability approximation to a sigmoid writes every neuron's spiking probability as the same link (sigmoid) applied to its own drive, offset by a shared scalar (threshold): $p_{\mathrm{s},n} = \sigma(k(\mu_{u,n} - \theta_{\mathrm{L}}))$ with $k = 1.702/\sigma_u > 0$. We write the pooled rate F and the pooled gain $\bar{\varphi}$ of Eq. 5 under this approximation as functions of the shared threshold:

$$F(\theta) = \tfrac{1}{N}\textstyle\sum_{n=1}^{N} \sigma\big(k(\mu_{u,n} - \theta)\big), \qquad \bar{\varphi}(\theta) = \tfrac{1}{N}\textstyle\sum_{n=1}^{N} k\,\sigma'\big(k(\mu_{u,n} - \theta)\big),$$

so that, within a window over which the drive is stationary, the layer-pooled update reads $\theta \leftarrow \theta + \eta_\theta\,(F(\theta) - r^*)\,\bar{\varphi}(\theta)$.

Proposition 1 (The layer-pooled fixed point is a normalizer)**.** *Let $N \geq 1$, $k > 0$, $\boldsymbol{\mu}_u \in \mathbb{R}^N$ and $r^* \in (0, 1)$. Then: (i) there exists a unique $\theta^* \in \mathbb{R}$ with $F(\theta^*) = r^*$, equivalently $\sum_n p_{s,n}(\theta^*) = Nr^*$; (ii) θ^* is the unique fixed point; and (iii) θ^* is locally asymptotically stable if $0 < \eta_\theta\,\bar{\varphi}(\theta^*)^2 < 2$.*

The fixed-point equation of (i) is fundamentally a normalization: scores $\mu_{u,n}$ are passed through a fixed link and one shared offset is chosen so that the total mass is a prescribed constant. The threshold θ_{L} is therefore not a nuisance parameter but the normalizer itself, and the threshold controller acts as a convergent fixed-point solver.

Across the N neurons in a layer, their mean pre-reset potentials $\mu_{u,n}$ differ. We write their standard deviation over the neurons as $\mathrm{sd}(\mu_u)$. The spiking probability of neuron n is a sigmoid function of $k(\mu_{u,n} - \theta_{\mathrm{L}})$, and $k^{-1}(= \sigma_u/1.702)$ represents the width of that sigmoid curve. The layer's behavior is determined by the ratio of the potential spread to this curve width:

$$D \;\triangleq\; k\,\mathrm{sd}(\mu_u) \;=\; 1.702\,\mathrm{sd}(\mu_u)/\sigma_u.$$

Small D cases. Here, most neurons stay in the graded region of the curve. At a low firing rate target $\bar{r}^*$, every neuron's spiking probability is small, and thus close to an exponential, $\sigma(x) \approx e^x$. The mass constraint of (i) in Proposition 1 leads to $e^{-k\theta_{\mathrm{L}}} \sum_n e^{k\mu_{u,n}} = Nr^*$, and solving for the threshold yields:

$$p_{\mathrm{s},n} \;\approx\; Nr^*\,\mathrm{softmax}(k\boldsymbol{\mu}_u)_n, \qquad \theta_{\mathrm{L}} \;\approx\; \frac{1}{k}\left[\log \textstyle\sum_{n=1}^{N} e^{k\mu_{u,n}} - \log Nr^*\right].$$

Each neuron's spiking probability effectively becomes a softmax over the mean pre-reset potentials, scaled to the prescribed mass.

Large D cases. The neurons' mean pre-reset potentials are spread far wider than the sigmoid curve, causing a number of neurons to stay either well above or well below the threshold. The sigmoid therefore acts as a step function: $p_{\mathrm{s},n} \to \mathbf{1}[\mu_{u,n} > \theta_{\mathrm{L}}]$. The same mass constraint then keeps the Nr^* most strongly driven neurons active. This is equivalent to a winner-take-all (WTA) mechanism in which the number of winners is set by the rate target $\bar{r}^*$. These dynamics are analyzed empirically in Appendix D.

Softmax and k-winner-take-all are thus the two ends of a single continuous mechanism, dynamically selected by the pre-reset potential distribution. This layer pooling highlights the normalizer within each sample quantity, demonstrating why a batch of one is not a special case. Batch pooling lacks a shared offset to normalize against; thus, the pooling axis inherently dictates whether the layer computes competitive normalization across neurons or merely regulates them individually.

Table 2: HomeoLayer against fixed $\theta \in \{0.25, 0.5, 1\}$, BNTT, and LayerNorm.

Dataset	Model	Fixed θ (best of 3)	BNTT	LayerNorm	HomeoLayer	Δ
SHD	AttnSNN	93.71±0.88	93.61±0.43	93.08±0.67	93.95±0.38	+0.24
SSC	AttnSNN	71.05±0.37	71.60±0.23	69.07±0.69	74.52±0.42	+2.92
GSC	AttnSNN	82.87±0.35	86.67±0.23	84.74±0.33	86.88±0.09	+0.21
SHD	FFSNN	90.18±0.87	92.52±0.42	84.11±0.51	90.39±0.34	-2.13
NMNIST	FFSNN	98.63±0.03	98.66±0.02	98.66±0.01	98.77±0.09	+0.11

5 Experiments

5.1 Protocol

Because our core claim is one of substitution, controlled relative differences matter more than absolute benchmark scores. Every comparison isolates the normalization mechanism by holding all other conditions (architecture, parameter budget, optimizer) strictly constant. **Unless otherwise stated, "HomeoLayer" denotes the layer-pooled variant.** While a more advanced architecture would likely elevate absolute accuracy across all methods, our evaluation focuses strictly on the normalization differential. We evaluate on long event streams (SHD, SSC, GSC, N-MNIST) using two SNN architectures (detailed in Appendix C). All models are trained under single-step truncation rather than BPTT, using the hyperparameters in Table A2. Accuracy is reported across three seeds ($n = 3$).

Feedforward SNN (FFSNN): A baseline architecture comprising an input linear layer, one hidden linear layer, and a readout layer.

Attention SNN (AttnSNN): An architecture featuring an input layer, two linear attention layers—whose fast weights are updated via the delta rule (Widrow & Hoff, 1960; Schlag et al., 2021)—and a readout layer.

Baselines. We benchmark HomeoLayer against fixed threshold cases, standard BNTT (per-timestep BatchNorm), and LayerNorm. These baselines are trained using identical hyperparameters but rely on conventional boxcar surrogate gradients (Table A2). Note that tdBN (which shares a single set of batch statistics across all timesteps) failed to yield comparable accuracy in our setup and is therefore excluded from the main comparisons.

5.2 HomeoLayer replaces normalization

Table 2 presents the central result of this paper. Evaluated against the best baseline (each trained under its own distinct recipe), HomeoLayer achieves parity on three of the five benchmarks and outperforms on one. Specifically, it performs within 0.24, 0.21, and 0.11 points on SHD AttnSNN, N-MNIST FFSNN, and GSC AttnSNN. On SSC, HomeoLayer significantly outperforms BNTT by 2.92 points (seven standard deviations). The sole exception is SHD FFSNN, where BNTT retains a 2.13-point lead.

The fixed-threshold column isolates the hyperparameter that the threshold controller actively eliminates. Its optimal value is architecture-dependent (Appendix F) and can only be discovered via an exhaustive sweep. On highly sensitive tasks (like SSC and GSC), accuracy degrades monotonically with sub-optimal thresholds. Even at its best swept value, the fixed threshold significantly trails HomeoLayer by 3.5 points on SSC AttnSNN and 4.0 points on GSC. Ultimately, a fixed threshold only competes with the threshold controller when perfectly manually tuned. The threshold controller discovers this optimal operating point autonomously via learned feedback.

We delimit our claims explicitly. On SHD AttnSNN, GSC, and N-MNIST, the performance gaps fall within a single seed standard deviation or a fraction of a point, demonstrating exact comparability rather than absolute superiority. The central claim we advance is that HomeoLayer strictly matches the performance of heavy temporal normalization across distinct architectures and modalities, achieving this parity at a fraction of the computational footprint.

Replacement of iterative threshold searches. HomeoLayer with learning the firing rate target allows us to avoid time-consuming iterative searches for optimal threshold. In Table 2, the hand-set threshold results in good accuracy because preliminary experiments found an appropriate threshold value.

Table 3: Number of variables per neuron for inference. The numbers of timesteps and neurons in the same layer are denoted by T and N, respectively.

Method	Affine	Statistics	Controller	Total	$\sqrt{\cdot}, \div$
BNTT	$2T$	$2T$	–	$4T$	×
LayerNorm	2	$2/N$	–	$2 + 2/N$	✓
HomeoLayer, batch	2	–	$2T_{\mathrm{w}} + 1 + 3/N$	$2T_{\mathrm{w}} + 3 + 3/N$	×
HomeoLayer, layer	2	–	$(2T_{\mathrm{w}} + 4)/N$	$2 + (2T_{\mathrm{w}} + 4)/N$	×

Learning in SNNs is significantly sensitive to the threshold as shown in Table A4 in Appendix F, where the full accuracy values for $\theta \in \{0.25, 0.5, 1\}$ are reported. Therefore, we can largely accelerate SNN learning by using HomeoLayer with learning the firing rate target. The cost of firing rate target learning is marginal since it can reuse the existing backprop pipeline.

5.3 Implementation cost

Table 3 compares the per-neuron inference memory footprint of each normalization mechanism. BNTT stores a running mean and variance for every neuron at every timestep ($2T$ values) alongside a temporal affine transform, totaling $4T$ variables per neuron. While static statistics can be folded into the weights to avoid runtime square root and division operations, dynamic evaluation prohibits this. LayerNorm requires two affine variables per neuron and computes shared layer statistics dynamically, permanently mandating square root and division operations at every timestep.

HomeoLayer similarly allocates two variables per neuron for its affine transform, but drastically reduces dynamic state and operational overhead. Under batch pooling (Algorithm A2), the threshold controller tracks two ring buffers of length T_{w} and a current threshold ($2T_{\mathrm{w}} + 1$ variables per neuron), plus three layer-shared scalars (learning rate η_θ, target firing rate $\bar{r}^*$, and initial threshold). Under layer pooling (Algorithm A2), this entire controller state is shared globally across the layer, reducing the controller's footprint to just $(2T_{\mathrm{w}}+4)/N$ variables per neuron. Because we set $T_{\mathrm{w}} = 1$ throughout this work, this memory overhead is strictly marginal.

Crucially, both HomeoLayer variants rely exclusively on simple addition and multiplication at test time. By entirely eliminating complex floating-point arithmetic and minimizing persistent state, HomeoLayer provides a highly hardware-compatible regulatory mechanism for neuromorphic substrates.

5.4 Ablation Study

Spike-flip statistics as the weight gradient. The threshold controller seamlessly reads the spike-flip statistics (unbiased estimator) as its gain $\hat{\varphi}$ (Section 4.2). A natural ablation asks whether these same binary statistics can entirely replace the closed-form weight gradient, allowing the entire layer to train using only simple spike statistics.

Table 4: Spike-flip statistic vs. closed-form gradient.

Dataset	Model	Approx grad	Closed-form grad
SHD	AttnSNN	93.98±0.76	93.95±0.38
SSC	AttnSNN	68.55±0.64	74.52±0.42
GSC	AttnSNN	85.79±0.26	86.88±0.09
SHD	FFSNN	86.74±0.33	90.39±0.34
NMNIST	FFSNN	98.23±0.08	98.77±0.09

Table 5: Batch pooling vs. layer pooling.

Dataset	Model	Accuracy (batch)	Accuracy (layer)
SHD	AttnSNN	93.27±0.24	**93.95±0.38**
SSC	AttnSNN	70.80±0.08	**74.52±0.42**
GSC	AttnSNN	84.35±0.19	**86.88±0.09**
SHD	FFSNN	**91.58±0.39**	90.39±0.34
NMNIST	FFSNN	98.51±0.03	**98.77±0.09**

Table 4 reports the outcome for all architectures when trained exclusively with the unbiased estimator. The results are notably split: AttnSNN on SHD and GSC successfully reaches the closed-form counterpart's accuracy, while the remaining configurations fall several percentage points short. Given this inconsistency across models and datasets, the exact closed-form gradient retains its role as HomeoLayer's default for weight updates, confining the lightweight unbiased estimator strictly to the threshold controller at inference.

Pooling axis. Layer pooling outperforms its BATCH counterpart across the three attention benchmarks and N-MNIST, with the SHD feedforward architecture being the sole exception (Table 5). We adopt layer pooling as our canonical default for the following reasons:
Batch independence: A layer-pooled threshold is derived strictly within-sample, ensuring that predictions remain entirely independent of batch composition. Consequently, inference at a batch size of one (online processing) is naturally supported, whereas batch pooling intrinsically couples samples and degenerates in the single-sample limit.
No accuracy penalty: As detailed in Table 5, layer pooling typically improves performance rather than degrading it. On SHD attention.
Low-cost controller: Because layer pooling requires only a single threshold per sample per layer, the controller's memory footprint per neuron asymptotically vanishes as $\mathcal{O}(1/N)$ (Table 3). Conversely, batch pooling mandates maintaining a distinct threshold state for every neuron.

5.5 THRESHOLD CONTROLLER TRANSPLANTS INTO DIFFERENT MODELS

HomeoLayer is built on LIF, but its threshold controller reads spikes only, so it can be transplanted into different neuron models. We port it, together with the ℓ_2 weight normalization, into C-SiLIF (Fabre et al., 2026) and RadLIF (Bittar & Garner, 2022), which are adaptive neuron models that achieve the state-of-the art on SHD and SSC. We replaced only normalization with our threshold controller and trained them under the reported configurations and backprop pipeline. The results are listed in Table A3 in Appendix E. In C-SiLIF and RadLIF, our threshold controller replaces the native normalization at the cost of $1.1\% - 1.4\%$ accuracy loss. These models regulate their own potential distribution relative to threshold through learned per-neuron dynamics, which the shared firing rate target competes with. Appendix E gives the evaluation protocol.

6 LIMITATIONS

While HomeoLayer achieves competitive parity with conventional normalization, its absolute accuracies fall slightly below the published state-of-the-art on SHD, SSC, and GSC given the use of plain architectures. The primary contribution of this work lies in the mechanism and its hardware efficiency rather than securing top benchmark positions. Furthermore, our evidence is currently weighted toward audio datasets, with validation on visual event tasks limited to a single dataset (N-MNIST).

Transplanting the threshold controller into adaptive models incurs a minor accuracy penalty of 1.1 to 1.4 points (Bittar & Garner, 2022; Fabre et al., 2026). Because these adaptive neurons self-regulate through learned per-neuron dynamics, enforcing a shared rate target competes directly with their learned heterogeneity. Consequently, HomeoLayer may be best deployed as a normalization replacement in architectures fundamentally constructed around a fixed threshold, rather than as a universal upgrade for all SNNs.

7 CONCLUSION

Standard normalization artificially standardizes incoming signals to ensure a fixed threshold remains usable. HomeoLayer flips this paradigm: it dynamically adjusts the threshold to accommodate whatever signal arrives, accomplishing this using only basic addition and multiplication operations. Across two distinct architectures and multiple long-event modalities, this homeostatic approach proves competitive with per-timestep batch normalization. It delivers substantial computational benefits by eliminating stored statistics, test-time batch dependencies, and costly division operations. Although tailored for LIF neurons, the threshold controller is inherently portable, successfully replacing normalization in published adaptive-neuron models with minimal performance trade-offs. Ultimately, this work demonstrates that closed-loop threshold adaptation is a highly effective, hardware-native alternative to normalization in SNNs.

AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on research experiments. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for creating

images, editing software code, editing a research paper to improve readability, and title proposal. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement

To ensure full reproducibility, HomeoLayer is completely specified in Section 4.1 and provided as pseudocode in Appendix A (Algorithms A1 and A2, which include the target-learning step).

The two evaluated architectures, their dataset-specific dimensions, and the attention mechanism equations are detailed in Appendix C (Table A1). Every hyperparameter for both our model and all baselines is explicitly listed in Table A2 (Appendix C). Notably, no hyperparameters were tuned on a per-dataset basis. The experimental protocol—including the single-step truncation scheme, data processing logic (time/channel binning and GSC log-mel standardization), and the three-seed evaluation convention—is fully described in Section 5.1 and Appendix C.

Comprehensive supporting data, including the per-threshold accuracies underlying the fixed-threshold baseline, are tabulated in Appendices F. The fixed-point derivation presented in Section 4.3 is formally proved in Appendix B. Finally, the adaptive-model transplant experiments are detailed in Appendix E.

Complete source code for HomeoLayer, both architectures, all baselines, the trained model analysis, and the transplant patches for published models—alongside deterministic execution commands for every reported run—is available anonymously at `https://anonymous.4open.science/r/snn_homeolayer-E003`.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Maximilian Baronig, Romain Ferrand, Silvester Sabathiel, and Robert Legenstein. Advancing spatio-temporal processing in spiking neural networks through adaptation. *Nature Communications*, 16: 5776, 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 (NeurIPS)*, 2018.

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

Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

Maxime Fabre, Lyubov Dudchenko, Younes Bouhadjar, and Emre Neftci. SiLIF: Structured state space model dynamics and parametrization for spiking neural networks. *arXiv:2506.06374v4*, 2026.

Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The SpiNNaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

Hector A. Gonzalez, Jiaxin Huang, Florian Kelber, Khaleelulla Khan Nazeer, Tim Langer, Chen Liu, Matthias Lohrmann, Amirhossein Rostami, Mark Schöne, Bernhard Vogginger, Timo C. Wunderlich, Yexin Yan, and Christian Mayr. SpiNNaker2: A large-scale neuromorphic system for event-based and asynchronous machine learning. *arXiv:2401.04491*, 2024.

Yufei Guo, Yuhan Zhang, Yuanpei Chen, Weihang Peng, Xiaode Liu, Liwen Zhang, Xuhui Huang, and Zhe Ma. Membrane potential batch normalization for spiking neural networks. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Youngeun Kim and Priyadarshini Panda. Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. *Frontiers in Neuroscience*, 15:773954, 2021.

Peiyu Liu, Jianhao Ding, and Zhaofei Yu. Training deep normalization-free spiking neural networks with lateral inhibition. *arXiv:2509.23253*, 2026.

Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

Garrick Orchard, E. Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T. Sommer, and Mike Davies. Efficient neuromorphic signal processing with Loihi 2. In *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 254–259, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning (ICML)*, pp. 9355–9366, 2021.

Jochen Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. In *International Conference on Artificial Neural Networks (ICANN)*, 2005.

Gina G. Turrigiano and Sacha B. Nelson. Homeostatic plasticity in the developing nervous system. *Nature Reviews Neuroscience*, 5(2):97–107, 2004.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record, Part 4*, pp. 96–104, 1960.

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

Wenrui Zhang and Peng Li. Information-theoretic intrinsic plasticity for online unsupervised learning in spiking neural networks. *Frontiers in Neuroscience*, 13:31, 2019.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *AAAI Conference on Artificial Intelligence*, 2021.

A PSEUDOCODE

Algorithm A1: HomeoLayer variable update. Black lines execute at inference using only addition and multiplication; blue lines evaluate during training to supply closed-form gradients.

Input: spikes from the previous block $\mathbf{s}_t^{[l-1]} \in \{0,1\}^m$ for $t = 1, \dots, T$
Given: weights $\mathbf{W} \in \mathbb{R}^{n \times m}$ with unit-norm rows, affine $\boldsymbol{\gamma}, \boldsymbol{\beta} \in \mathbb{R}^n$, decays $\beta = e^{-1/\tau_{\text{syn}}}$, $\alpha = e^{-1/\tau_u}$, surrogate variance $\sigma^2_{\xi,\text{surr}}$, rate target logit ρ
Initialize: $\mathbf{p}_{\text{op}} \in \mathbb{R}^m \leftarrow \mathbf{0}$, $\mathbf{u} \in \mathbb{R}^n \leftarrow \mathbf{0}$, $\theta \leftarrow \theta_0$, $r^* \leftarrow \sigma(\rho)$;
Initialize: $\mathbf{m}_u, \boldsymbol{\sigma}^2_{\text{post}} \in \mathbb{R}^n \leftarrow \mathbf{0}$;
for $t = 1$ **to** T **do**
  $\mathbf{p}_{\text{op}} \leftarrow \beta\, \mathbf{p}_{\text{op}} + p_0\, \mathbf{s}^{[l-1]}[t]$; // synaptic filter
  $\mathbf{x} \leftarrow \boldsymbol{\gamma} \odot (p_0 \tau_{\text{syn}})^{-1}\, \mathbf{W}\mathbf{p}_{\text{op}} + \boldsymbol{\beta}$; // affine transform
  $\mathbf{u}_{\text{ac}} \leftarrow \alpha \mathbf{u}_{\text{ac}} + \mathbf{x}$; // membrane potential
  $\boldsymbol{\mu}_u \leftarrow \alpha\, \mathbf{m}_u + \mathbf{x}$; $\boldsymbol{\sigma}^2_u \leftarrow \alpha^2 \boldsymbol{q}^2_u + \sigma^2_{\xi,\text{surr}}$; // pre-reset moments
  $p_s \leftarrow$ Eq. (2); $\partial p_s / \partial \mu_u \leftarrow$ Eq. (3);
  $\mathbf{m}_u \leftarrow (1 - p_s)\boldsymbol{\mu}_u$; $\boldsymbol{\sigma}^2_{\text{post}} \leftarrow (1-p_s)\boldsymbol{\sigma}^2_u + p_s(1-p_s)\boldsymbol{\mu}^2_u$; // soft expected reset
  $\mathbf{s}_t^{[l]} \leftarrow H(\mathbf{u}_{\text{ac}} - \theta)$; $\mathbf{u}_{\text{ac}} \leftarrow \mathbf{u}_{\text{ac}} \odot \left(\mathbf{1} - \mathbf{s}_t^{[l]}\right)$; // spike and hard reset
  $\varphi \leftarrow \max([\mathbf{s}_{t-1}^{[l]} \odot (\mathbf{1} - \mathbf{s}_t^{[l]}) + (\mathbf{1} - \mathbf{s}_{t-1}^{[l]}) \odot \mathbf{s}_t^{[l]}]/2, \varphi_{\min})$; // Gain estimate
  Update θ using **Algorithm** A2 with $\mathbf{s}_t^{[l]}$ and φ
end

Algorithm A1 details a single timestep of the HomeoLayer layer, encompassing the synaptic filter, the ℓ_2-normalized weights, the affine transform, membrane integration, hard reset, the one-bit gain estimate, and the threshold controller call. Black lines execute at inference using only addition and multiplication; blue lines evaluate strictly during training to supply closed-form moments and gradients (Section 3). The rate target enters via its logit ρ, which is the sole parameter that the task loss updates through the threshold.

Algorithm A2: One timestep of the threshold controller.

Given : spikes $s_t \in \{0,1\}^{B \times N}$, previous spikes s_{t-1}, threshold θ
rate target $\bar{r}^*$, learning rate η_θ, pooling axis $A \in \{\text{BATCH}, \text{LAYER}\}$
State : ring buffers $\mathcal{R}$ (length T_{w}), $\mathcal{G}$ (length T_{w})

/* $\Pi_A(x) = \text{mean}_b\, x[b,n] \in \mathbb{R}^N$ under BATCH, one value per neuron;
   $\text{mean}_n\, x[b,n] \in \mathbb{R}^B$ under layer, one per sample */
$\varphi \leftarrow \max([s_{t-1}(1-s_t) + (1-s_{t-1})s_t]/2, \varphi_{\min})$
if $A =$ BATCH **then**
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{\Pi_{\text{BATCH}}(\varphi)\}$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{\Pi_{\text{BATCH}}(s_t)\}$ // oldest evicted at T_{w}
  $\bar{\varphi} \leftarrow \text{mean}_t(\mathcal{G})$ $\bar{r} \leftarrow \text{mean}_t(\mathcal{R})$
  $\theta \leftarrow \theta + \eta_\theta\, (\bar{r} - \bar{r}^*)\bar{\varphi}$ // $\theta \in \mathbb{R}^N$, per neuron
else // $A =$ LAYER
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{\Pi_{\text{LAYER}}(\varphi_t)\}$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{\Pi_{\text{LAYER}}(s_t)\}$ // oldest evicted at T_{w}
  $\bar{\varphi} \leftarrow \text{mean}_t(\mathcal{G})$ $\bar{r} \leftarrow \text{mean}_t(\mathcal{R})$
  $\theta \leftarrow \theta + \eta_\theta\, (\bar{r} - \bar{r}^*)\bar{\varphi}$ // $\theta \in \mathbb{R}^B$, per sample

Algorithm A2 details the threshold controller's internal execution: managing the flip statistic (with its floor), updating the windowed rate and gain ring buffers, and executing the threshold step. The difference between batch and layer pooling lies entirely in the axis over which means are computed, dictating the shape of θ.

B Proof of Proposition 1

The functions F and $\bar{\varphi}$ are defined as follows in Section 4.3.

$$F(\theta) = \tfrac{1}{N}\sum_{n=1}^{N} \sigma\big(k(\mu_{u,n} - \theta)\big), \qquad \bar{\varphi}(\theta) = \tfrac{1}{N}\sum_{n=1}^{N} k\,\sigma'\big(k(\mu_{u,n} - \theta)\big).$$

Proposition 1 (The layer-pooled fixed point is a normalizer)**.** *Let $N \geq 1$, $k > 0$, $\boldsymbol{\mu}_u \in \mathbb{R}^N$ and $r^* \in (0, 1)$. Then: (i) there exists a unique $\theta^* \in \mathbb{R}$ with $F(\theta^*) = r^*$, equivalently $\sum_n p_{s,n}(\theta^*) = Nr^*$; (ii) θ^* is the unique fixed point; and (iii) θ^* is locally asymptotically stable if $0 < \eta_\theta\, \bar{\varphi}(\theta^*)^2 < 2$.*

Proof. (i) Because $\sigma'(x) = \sigma(x)(1 - \sigma(x)) > 0$ for all x, each term of F is continuous and strictly decreasing with θ, and thus so is their mean. As $\theta \to -\infty$, every term tends to 1, and as $\theta \to +\infty$, every term tends to 0. Therefore, F maps $\mathbb{R}$ onto $(0, 1)$. Since $r^* \in (0, 1)$, the intermediate value theorem guarantees the existence of a θ^* satisfying $F(\theta^*) = r^*$, and strict monotonicity ensures its uniqueness. Multiplying by N recovers the mass constraint formulation.

(ii) Differentiating F term by term yields $F'(\theta) = -\bar{\varphi}(\theta) < 0$ for all θ. A controller step vanishes if and only if $F(\theta) = r^*$, meaning $\theta = \theta^*$. For $\theta < \theta^*$, monotonicity implies $F(\theta) > r^*$, making the update step $\eta_\theta(F(\theta) - r^*)\bar{\varphi}(\theta)$ strictly positive. For $\theta > \theta^*$, it is strictly negative.

(iii) Let $G(\theta) = \theta + \eta_\theta(F(\theta) - r^*)\bar{\varphi}(\theta)$. Then $G'(\theta^*) = 1 + \eta_\theta F'(\theta^*)\bar{\varphi}(\theta^*) + \eta_\theta(F(\theta^*) - r^*)\bar{\varphi}'(\theta^*)$. Because $F(\theta^*) = r^*$ and $F' = -\bar{\varphi}$, this simplifies to $G'(\theta^*) = 1 - \eta_\theta\, \bar{\varphi}(\theta^*)^2$. The fixed point is locally asymptotically stable when $|G'(\theta^*)| < 1$, which holds if and only if $0 < \eta_\theta \bar{\varphi}(\theta^*)^2 < 2$. □

C Model architectures and hyperparameters

Table A1: Input and output dimensions per dataset.

Dataset	Native input	Reduction	Input to the model	Output
SHD	700 channels, spikes	5:1 channel binning	140	20
SSC	700 channels, spikes	5:1 channel binning	140	35
GSC	40 log-mel bins, analog	None	40	35
N-MNIST	$34 \times 34 \times 2$ pixels, spikes	flattened, 5:1 binning	463	10

Both architectures are constructed using HomeoLayer as their fundamental spiking block. A block comprises the synaptic filter, ℓ_2-normalized weights, per-feature affine transform, and a probabilistic LIF neuron governed by the threshold controller. The readout is a non-spiking leaky integrator that outputs time-averaged logits without a threshold. Additionally, no affine transformation applies to the readout. Table A1 lists the input and output dimensions for all datasets used in this work.

Feedforward (FFSNN). A baseline architecture consisting of an input block, one hidden block of 700 spiking neurons, and a readout block, for all datasets.

Attention (AttnSNN). An input block is followed by two linear attention layers. Each attention layer projects queries, keys, and values via standard linear maps on the synaptic trace (with ℓ_2-normalized rows), split into 14 heads of width 50. Attention operates via a sample-specific fast-weight matrix A updated dynamically via the delta rule. The attention output is delivered to a spiking neuron of width 700 after Frobenius normalization. Because the projections lack neurons, they lack thresholds and are explicitly exempt from both normalization and homeostasis.

Attention layer in detail. Let $\mathbf{s}[t] \in \{0, 1\}^N$ be the spikes entering the layer (of $N(= 700)$ neurons) at timestep t. Let the number of heads $H = 14$, each head is of head dimension $d_{\text{H}} = 50$. The projections, $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{Hd_{\text{H}} \times N}$, are linear in the spikes, with no bias. The projection matrices are row-wise ℓ_2-normalized.

$$\mathbf{q}[t] = \mathbf{W}_q\mathbf{s}[t], \quad \mathbf{k}[t] = \mathbf{W}_k\mathbf{s}[t], \quad \mathbf{v}[t] = \mathbf{W}_v\mathbf{s}[t].$$

Each is reshaped into per-head vectors $\mathbf{q}_h, \mathbf{k}_h, \mathbf{v}_h \in \mathbb{R}^{d_{\text{H}}}$. The key and value vectors are low-pass filtered with the same time constant τ_{syn} as for HomeoLayer.

$$\tilde{\mathbf{k}}_h[t] = \beta\, \tilde{\mathbf{k}}_h[t-1] + \tfrac{1}{\tau_{\text{syn}}}\mathbf{k}_h[t], \qquad \tilde{\mathbf{v}}_h[t] = \beta\, \tilde{\mathbf{v}}_h[t-1] + \tfrac{1}{\tau_{\text{syn}}}\mathbf{v}_h[t], \qquad \beta = e^{-1/\tau_{\text{syn}}}.$$

The fast weight matrix for head h, $\mathbf{A}_h \in \mathbb{R}^{d_{\mathrm{H}} \times d_{\mathrm{H}}}$, is written by the delta rule without decay,

$$\mathbf{A}_h[t] = \mathbf{A}_h[t-1] + \eta \left(\tilde{\mathbf{v}}_h[t] - \mathbf{A}_h[t-1]\tilde{\mathbf{k}}_h[t]\right) \tilde{\mathbf{k}}_h[t]^\top, \qquad \eta = 0.01.$$

The read applies the normalized fast weight matrix to the current query $\mathbf{q}_h[t]$,

$$\mathbf{y}_h[t] = g \, \frac{\mathbf{A}_h[t]}{\|\mathbf{A}_h[t]\|_F} \mathbf{q}_h[t], \tag{A1}$$

where g is a gain that is a learned scalar. The reads from H heads are concatenated, leading to $\mathbf{y}[t] \in \mathbb{R}^N$ that works as the drive for the attended neurons.

$$\mathbf{x}[t] = \boldsymbol{\gamma} \odot \mathbf{y}[t] + \boldsymbol{\beta}, \qquad \mathbf{u}[t] = \alpha \mathbf{u}[t-1] + \mathbf{x}[t], \qquad \mathbf{a}[t] = H(\mathbf{u}[t] - \theta[t]) \in \{0,1\}^N.$$

The attended spikes $\mathbf{a}[t]$ enter a HomeoLayer block with a weight matrix $\mathbf{W}_o \in \mathbb{R}^{N \times N}$, resulting in attention output spikes $\mathbf{o}[t] \in \{0,1\}^N$. The layer's output is the residual sum

$$\mathbf{s}^{\text{out}}[t] = \mathbf{o}[t] + \mathbf{s}[t].$$

Note that this attention operation needs the per-head normalization of $\mathbf{A}_h$ as in Eq. A1, which is a property of this attention design and not of HomeoLayer.

Table A2 details the hyperparameter configurations for all experiments. No dataset-specific tuning, learning rate schedules, weight decay, or gradient clipping were employed.

Table A2: Hyperparameters used.

	Setting	Value
Shared	Optimizer	Adam, LR 10^{-3}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-16}$, no weight decay
	Batch size	128
	No. Epochs	150 (SHD), 60 (SSC, GSC, N-MNIST); best test accuracy reported
	No. Seeds	3
	Time bin	4 ms; clips 700 / 1000 / 1000 / 310 ms (SHD / SSC / GSC / N-MNIST)
	Input encoding	Low-pass synaptic filter, $\tau_{\text{syn}} = 5$ ms
	Membrane	LIF, $\tau_u = 20$ ms, hard reset to 0
	Hidden width	700 per spiking block
	Attention	2 layers, 14 heads of width 50, delta-rule fast weights with $\eta = 0.01$,
	Readout	Non-spiking leaky integrator, time-averaged logits
	Augmentation	Channel shift ± 40 (± 2 on GSC's 40 mel bins); same-class blend ($p = 0.5$)
HomeoLayer	Weights	ℓ_2-normalized rows (kernels), no bias
	Affine	Per-feature $\gamma = 1, \beta = 0$ at initialization
	Gradient	Closed-form density at threshold (Eq. 3), $\sigma^2_{\xi,\text{surr}} = 0.1$, no injected noise
	Controller	Layer pooling (one threshold per sample), window $T_{\text{w}} = 5$ ms (1 step)
	Gain	$\eta_\theta = 10$, rate from spike counts, gain from one-bit flip statistic
	Threshold	Initialized at 1, reset at the start of every sequence in training and inference
	Rate target	Learned ($r^* = \sigma(\rho)$), initialized at 0.10 (hidden) and 0.01 (input)
Standard	Surrogate	Boxcar of half-width 0.5 around the threshold
	Weights	Unnormalized, no bias
	Threshold	Fixed at 1 (BNTT, LayerNorm); best of $\{0.25, 0.5, 1\}$ for Fixed-θ arm
	BNTT	`BatchNorm1d` per timestep per layer, momentum 0.1, its own γ, β
	LayerNorm	`LayerNorm` over features per timestep, its own γ, β
	Fixed-θ arm	No normalization, no affine, no homeostasis

D MEASURING WHAT LAYER POOLING COMPUTES

Section 4.3 proves that the layer-pooled threshold controller mathematically instantiates a normalizer whose precise behavior is dictated by membrane noise.

Trained attention layers operate at the top-*k* extreme. Figure A1 demonstrates this on the attended neurons of the first attention layer of 700 neurons (sampled from AttnSNN on an SHD test batch of 128 in size). In the trained configuration, the drive spread heavily outweighs the membrane noise such that, per timestep and sample, $\mathrm{sd}(\mu_u)$ has a median of 5.7 and a mean of 21.7 against σ_u of 0.6

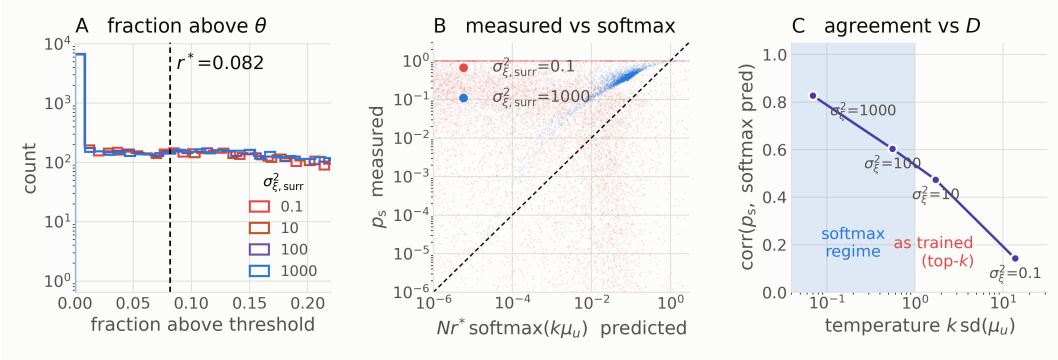


Figure A1: Empirical measurements on the attended neurons of the first attention layer (in AttnSNN trained on SHD with 0.1 $\sigma_{\xi,\text{surr}}$) on a 128-sample SHD test batch. The surrogate variance $\sigma^2_{\xi,\text{surr}}$ is swept at evaluation across $\{0.1, 10, 100, 1000\}$, which strictly widens the mean pre-reset potential μ_u distribution while holding weights, drive, spikes, and thresholds constant. The layer's learned target is $r^* = 0.082$. **A** Distribution of fraction of neurons whose mean pre-reset potential μ_u exceeds the sample-specific threshold, which was counted per timestep and sampled over the second half of the sequence. The dashed line indicates the learned fixed-point fraction r^*. The layer remains silent approximately 60% of the time; during active periods, the active fraction spans 0.01 to 0.20 (averaging 39 to 44 neurons of total 700). The surrogate variance sweep leaves this macroscopic mass unchanged. **B** Empirical spiking probability p_s plotted against the analytic softmax prediction $Nr^* \,\text{softmax}(k\mu_u)$ on logarithmic axes at the sweep extremes; the dashed line denotes perfect equality. **C** Correlation between the empirical probability and the softmax prediction as a function of the drive spread in noise units, $D = k\,\text{sd}(\mu_u)$ introduced in Section 4.3, with the pure softmax regime shaded. In normalized units $(\mu_u - \theta)/\sigma_u$, the transfer link remains an identically graded sigmoid across all temperatures; the sweep exclusively alters the width of this transition band relative to the raw drive spread μ_u. While the trained network natively operates at the hard top-k extreme, modulating this single surrogate variance parameter translates the layer seamlessly across the entire functional family.

to 1.0, and the median of D over timesteps and samples is 13.7. Consequently, the mean deviation $|p_\text{s} - \mathbf{1}[\mu_u > \theta]|$ is just 0.019, and only 9% of neurons occupy the graded probability band. The layer effectively performs a hard selection, activating roughly 39 of its 700 neurons. Sweeping the surrogate variance $\sigma^2_{\xi,\text{surr}}$ during evaluation smoothly transitions the layer toward the opposite limit (temperature $13.7 \to 0.1$, expanding the graded band from 9% to 98%). Comparing the empirical p_s against the analytic prediction $Nr^* \,\text{softmax}(k\mu_u)$ confirms this transition: their correlation rises from 0.14 at the trained temperature to 0.83 at the softmax extreme (Figure A1**C**).

The attended neurons represent the most extreme case, as the fast-weight read induces a drive spread twenty times larger than the noise. In contrast, standard linear blocks within the same network operate closer to the graded regime ($D \in [3, 7]$), placing a third of their neurons in the graded band, while the input block ($D = 4.5$) softly selects 7 out of 700 neurons. The homeostatic normalization is therefore not merely a theoretical approximation; it is an intrinsic property across the entire functional family, where the membrane noise organically selects the realized operating point.

E Transplanting the threshold controller into adaptive models

Because the threshold controller evaluates only binary spikes, it can be transplanted seamlessly into alternative, non-LIF neuron architectures. We demonstrate this by porting it into the official codebases of C-SiLIF (Fabre et al., 2026) and RadLIF (Bittar & Garner, 2022)—adaptive-neuron models defining the state-of-the-art on SHD and SSC. We excised their native batch normalization, inserted the HomeoLayer controller (with ℓ_2 weight normalization), and trained them under their original surrogate gradient pipelines.

Interpretation and state retention. Adaptive models like C-SiLIF and RadLIF regulate their operating points dynamically by construction (via complex state transitions or adaptive subthreshold

Table A3: The threshold controller replacing native normalization in published adaptive models. Configurations match the authors' released code across three standard seeds. The baselines were independently reproduced.

Model	**Dataset**	**Native norm.**	**Threshold controller**	**Δ**
C-SiLIF	SHD	94.99±0.61	93.63±0.13	−1.4
C-SiLIF	SSC	79.46±0.13	78.25±0.12	−1.2
RadLIF	SSC	77.84±0.69	76.75±0.28	−1.1

currents). Because these models learn fine-grained per-neuron heterogeneity, imposing a globally shared firing-rate target mathematically competes with their native dynamics, explaining the minor 1.1–1.4% accuracy penalty (Table A3).

F SENSITIVITY TO FIXED THRESHOLDS

Table A4 details the accuracy of models trained without normalization or homeostasis across various static thresholds. Fixed thresholds match homeostatic performance only when manually swept and precisely tuned to the dataset. HomeoLayer completely eliminates this hyperparameter sweep by learning the target end-to-end.

Table A4: Learning sensitivity to a fixed threshold without homeostasis.

Dataset	**Model**	θ=**0.25**	θ=**0.5**	θ=**1**	**HomeoLayer**
SHD	AttnSNN	93.61±0.66	93.55±0.49	**93.71**±0.88	93.95±0.38
SSC	AttnSNN	69.11±0.47	70.63±0.21	**71.05**±0.37	74.52±0.42
GSC	AttnSNN	76.87±0.80	81.07±0.30	**82.87**±0.35	86.88±0.09
SHD	FFSNN	87.60±1.62	**90.18±0.87**	89.53±0.08	90.39±0.34
NMNIST	FFSNN	98.37±0.15	98.52±0.01	**98.63±0.03**	98.77±0.09