Article

Not peer-reviewed version

Spiking Neural Networks: Mathematical Foundations

Ismail Can Dikmen *

Posted Date: 18 May 2026

doi: 10.20944/preprints202605.1130.v1

Keywords: spiking neural networks; leaky integrate-and-fire; point processes; time-rescaling theorem; surrogate gradient; hazard-based neuron models; neuromorphic computing

 doi:10.20944/preprints202605.1130.v1

Article

Spiking Neural Networks: Mathematical Foundations

Ismail Can Dikmen

Department of Electrical Electronics Engineering İstinye University, İstanbul, Türkiye; can.dikmen@istinye.edu.tr

Abstract

This article presents the mathematical foundations of spiking neural networks (SNNs) in a unified formalism, with a deliberate emphasis on *derivational provenance*. The same neuron model is written one way in computational neuroscience textbooks, another way in machine learning papers, and a third way in the stochastic process literature. Even within a single line of work, papers absorb constants into other constants until two equations from two sources cannot be compared by inspection. We collect the core mathematics in one place, and we attach a status label to every major equation so that the reader sees at a glance whether a given step is a mathematical identity, a parameter limit, a formal approximation under stated conditions, or a useful but unproven heuristic. The labels are *exact*, *reduction*, *approximation*, and *heuristic*. The substantive content is the following. The reduction chain from Hodgkin-Huxley dynamics through the adaptive exponential integrate-and-fire model down to leaky integrate-and-fire (LIF) is given with status labels at every step, including the spike response model as an exact reformulation under linear subthreshold dynamics. Reset semantics are analyzed in three forms (hard, soft, no reset), with implications for both spike statistics and gradient flow. Network dynamics are written down in a coupled form, and the analytical theory of recurrent SNNs (liquid state machines, the echo state property, balanced excitatory-inhibitory networks) is reviewed with explicit conditions on time constants and weight matrices. The full point process formulation is developed: counting processes, conditional intensities, the time-rescaling theorem, the likelihood for general history-dependent point processes, and the canonical model classes (homogeneous Poisson, inhomogeneous Poisson, Hawkes, point-process generalized linear models). The bridge between state-space SNNs and intensity-based formulations is made explicit, including conditions under which a generalized linear model can be embedded in a finite-dimensional spiking state space. Information-theoretic aspects of spike coding are presented through Fisher information, with a quantitative comparison of rate and time-to-first-spike codes. Computational capacity is treated through three lines of results: the Maass third-generation argument and its noisy temporal-coding strengthening, the Stanojevic exact mapping from feedforward ReLU networks to time-to-first-spike SNNs, and the Date-Schuman Turing-completeness construction. The article closes with a status-labeled taxonomy of the hazard-based H-LIF family and its Liquid extension, drawn from a public, patent-scoped reference implementation with custom CUDA kernel and FPGA validation; the other LIF variant families (multi-spectral, wavelet, fractional, control-theoretic, information-theoretic, and domain-specific gating) are deferred to a companion v2. This article is the second installment in a series on spiking neural networks. The first installment, *Spiking Neural Networks: A Tutorial on Models, Coding, and Training* [1], introduces the practical side at a tutorial level; the present article develops the underlying mathematics in depth. The two share notation, and a reader who has followed the first installment can read this one essentially in any order; the cross-references between them are explicit. The intended audience is the graduate student or researcher who needs the mathematical underpinnings of SNNs in a single document, rather than reconstructed from a dozen textbooks and review papers.

Keywords: spiking neural networks; leaky integrate-and-fire; point processes; time-rescaling theorem; surrogate gradient; hazard-based neuron models; neuromorphic computing

 doi:10.20944/preprints202605.1130.v1

1. Introduction

The mathematical literature on spiking neural networks is rich, fragmented, and old enough to have accumulated several independent notational traditions. Lapicque's integrate-and-fire model dates from 1907 [2]; Hodgkin and Huxley wrote down their voltage-clamp equations in 1952 [3]; the point process treatment of spike trains was systematized by the 1980s [4–6]; the third-generation expressivity argument is from the late 1990s [7,8]; the surrogate gradient framework that drives modern deep SNNs is from the late 2010s [9,10]. Each of these threads developed its own conventions for time, currents, voltages, and spike trains, and the resulting picture is harder to read than it should be.

This article is an attempt to put the mathematical foundations of SNNs in one place, in one notation, with the derivational status of each step made explicit. We do not introduce fundamentally new theoretical frameworks; the model variants discussed in Section 8 are organized from a patent-scoped public reference implementation [11], and several of the constructions presented here have been rediscovered or simplified in modern papers. We cite the originals where we know them.

1.1. Why Status Labels

The same equation can describe very different things depending on what the surrounding text claims about it. Consider a typical step in a derivation,

$$\tau_m \frac{dV}{dt} = -(V - E_L) + R_m I(t).$$

This may be the *exact* subthreshold dynamics of a single-compartment LIF neuron after the current-based approximation; it may be a *reduction* from a higher-dimensional system under an explicit limit; it may be a *heuristic* stand-in for biophysical dynamics whose accuracy has not been quantified. The reader cannot tell from the equation itself. We have observed that this is a real source of confusion when students first try to read across textbooks and modern papers.

Throughout the article, every major equation carries one of four labels.

- *exact*: the equation is a mathematical identity, a definition, or an exact solution under stated assumptions.
- *reduction*: the equation follows from a higher-dimensional model under a parameter limit that is stated explicitly. The limit itself is exact, but it removes structure that may matter outside the limit.
- *approximation*: the equation involves a formal approximation. The error is not necessarily bounded, but the conditions under which the approximation is expected to be small are stated.
- *heuristic*: the equation is a computationally useful construction that lacks a rigorous justification. We use it because empirical evidence supports it, not because of a derivation.

The labels are written next to equations in a uniform format and explained in Section 2. They are intended as a reading aid, not as a formal type system. A label tells the reader what to expect from a derivation that follows; it does not replace the derivation.

1.2. Scope and Structure

We focus on the mathematical content rather than implementation, training pipelines, or hardware. A practical introduction with code-level guidance, a treatment of neuromorphic datasets, and a tour of training frameworks is in the first installment of the series [1]. That paper covers the path from a first reading to a working SNN; this paper covers the underlying mathematics in greater depth.

Section 2 fixes notation and explains the status-label system. Section 3 develops the reduction chain from Hodgkin-Huxley through AdEx to LIF and SRM, with each step carrying its label. Section 4 writes down the network equations and develops the theory of recurrent SNNs (liquid state machines, the echo state property, balanced networks). Section 5 develops the point process formulation, with the time-rescaling theorem, the general history-dependent likelihood, and the bridge to state-space SNNs. Section 6 treats the information-theoretic aspects of spike coding, in particular the Fisher information

 doi:10.20944/preprints202605.1130.v1

for rate and time-to-first-spike codes. Section 7 presents the three main capacity results (Maass, Stanojevic, Date) and the open questions in SNN complexity. Section 8 gives a status-labeled taxonomy of the hazard-based LIF family and its Liquid extension, drawn from a public, patent-scoped reference implementation; a broader taxonomy covering the other variant axes is deferred to a companion v2. Section 9 closes with a list of open problems.

The reader is assumed to be comfortable with ordinary differential equations, basic measure theory and probability, the standard formalism of artificial neural networks, and elementary information theory. Background in computational neuroscience is helpful but not required; we develop the relevant biophysics from the equations.

1.3. What This Article Does Not Do

A few topics that a reader might expect to see are deliberately left out.

Neuromorphic hardware:

Analog and digital neuromorphic implementations (Loihi [11], SpiNNaker, BrainScaleS, TrueNorth, Akida) are out of scope. The mathematics is independent of the hardware substrate. A separate body of literature treats the engineering trade-offs and we do not duplicate it here.

Implementation and software:

Code, frameworks (snnTorch, SpikingJelly, Norse), and dataset loaders are covered in the first installment of the series [1] and are absent here.

Empirical benchmark results:

Accuracy numbers, ablation studies, and architecture comparisons are not the subject of a foundations paper. We mention published numbers only where they bear directly on a theoretical claim.

Open problems beyond a final summary:

A research community's open problems list is large and moves quickly. We give a focused summary in Section 9 and stop there.

2. Notation and Conventions

This section collects the notation used throughout. Table 1 lists the most frequently used symbols. The status-label system is described in detail at the end of the section. A reader familiar with computational neuroscience can skim this section; the status labels are the part most likely to differ from other treatments.

Table 1. Summary of key symbols used throughout the article.

Symbol	Description	Units	Section
$V(t)$, $V_i(t)$	membrane potential	mV	3
$V_i[t]$	discrete-time membrane potential	mV	3
E_L, E_{rest}	resting / leak reversal potential	mV	3
V_{th}	firing threshold	mV	3
V_{reset}	reset potential	mV	3
C_m	membrane capacitance	F	3
g_L	leak conductance	S	3
$\tau_m = C_m/g_L$	membrane time constant	s	3
$R_m = 1/g_L$	membrane resistance	Ω	3
$I_{\text{tot}}(t)$	total input current (physical)	A	3
$S_i(t)$	spike train of neuron i	s^{-1}	3
$s_i[t]$	discrete-time spike output	$\{0,1\}$	3
$\kappa(t)$	postsynaptic current kernel	s^{-1}	3
$\eta(t)$	refractory kernel (SRM)	mV	3.5
$\epsilon(t)$	PSP kernel (SRM)	mV	3.5
w_{ij}	synaptic weight from j to i	varies	3
d_{ij}	transmission delay from j to i	s	4

 doi:10.20944/preprints202605.1130.v1

Table 1. *Cont.*

Symbol	Description	Units	Section
$\mathbf{W}$, $\mathbf{W}_{\text{rec}}$	weight matrices (input, recurrent)	varies	4
$N_i(t)$	counting process for neuron i	–	5
$\lambda_i(t \mid \mathcal{H}_t)$	conditional intensity	s^{-1}	5
$\mathcal{H}_t$	filtration (history up to t)	–	5
$\Lambda_i(t)$	integrated intensity	–	5
$\mathcal{I}(\theta)$	Fisher information matrix	varies	6
$I(X;Y)$	mutual information	nats or bits	6
ρ	spectral radius	–	4
β	escape-rate sharpness or surrogate sharpness	–	5
$\alpha = e^{-\Delta t/\tau_m}$	discrete leak factor	–	3

2.1. Continuous Time and Discrete Time

Most neuron models are stated first in continuous time, with $t \in \mathbb{R}_{\geq 0}$ measured in seconds or milliseconds. Voltages, currents, and conductances are physical quantities with the dimensions in Table 1. When we move to discrete time, we write $V_i[t]$ with square brackets to distinguish it from the continuous-time variable $V_i(t)$. The discrete time index t stands for the t-th step of size Δt, so the physical time at index t is $t \cdot \Delta t$. In the discrete setting, spike outputs are binary, $s_i[t] \in \{0, 1\}$.

2.2. Spike Trains

A spike train of neuron i is written as a sum of Dirac delta functions, $S_i(t) = \sum_f \delta(t - t_f^{(i)})$, with $t_f^{(i)}$ the firing times. The same spike train is described by the counting process $N_i(t) = \int_0^t S_i(s)\, ds$. In the point process sections (Section 5), we use $N_i(t)$ and $\lambda_i(t \mid \mathcal{H}_t)$ for the counting process and conditional intensity, respectively, and $\mathcal{H}_t$ for the filtration of all observed spike trains up to time t. The Dirac form is convenient inside convolutions; the counting form is convenient inside integrals.

2.3. Currents and Normalized Inputs

In the conductance-based derivation (Section 3.4), $I_{\text{tot}}(t)$ has units of amperes and appears in the LIF equation as $R_m I_{\text{tot}}(t)$, which has units of volts. Many papers absorb R_m into the input and write the LIF equation with a dimensionless or voltage-scaled input $I(t)$. We keep R_m explicit throughout the derivation in Section 3.4 and note the convention change whenever it occurs.

2.4. Equation Status Labels

A status label is a tag attached to an equation that tells the reader the derivational status of the equation. Four labels are used.

exact

The equation is a mathematical identity, a definition, or an exact solution to a stated problem under stated assumptions. No approximation is involved. Examples: the closed-form solution of a linear ODE between events; the definition of the conditional intensity; a parameter limit that yields a simpler model exactly.

reduction

The equation follows from a higher-dimensional model under a parameter limit. The limit itself is exact, but the original model contained structure that the reduced model does not. The reduction is justified under the stated limit but may break outside it. Example: setting $a = b = 0$ and $\Delta_T \to 0$ in AdEx to recover LIF.

approximation

The equation involves a formal approximation. The conditions under which the approximation is expected to be small are stated, but a tight error bound is not always available. Example: the current-based approximation in deriving LIF from a conductance-based model, valid when V stays close to E_L between spikes.

 doi:10.20944/preprints202605.1130.v1

heuristic

The equation is a computationally useful construction that lacks a rigorous justification. We use it because empirical evidence supports it. Example: the surrogate gradient replacement of the Heaviside derivative.

The labels are placed at the end of the equation in the form [*label: short justification*]. They are not exhaustive: a single equation can be exact under one reading and approximate under another, and we use the label that fits the context of the surrounding derivation. Where the choice matters, we explain it. Two examples illustrate the system.

Example 1:

The closed-form between-event solution of the LIF subthreshold ODE,

$$V(t) = E_L + \big(V(t_0) - E_L\big)e^{-(t-t_0)/\tau_m} + \frac{1}{C_m}\int_{t_0}^{t} e^{-(t-s)/\tau_m} I_{\text{tot}}(s)\, ds,$$

is *exact*. The integration is exact between any two consecutive spike events, and the formula is a closed-form solution of a linear first-order ODE.

Example 2:

The Euler discretization of the same ODE,

$$V[t+1] = V[t] + \frac{\Delta t}{\tau_m}\big(-(V[t] - E_L) + R_m I_{\text{tot}}[t]\big),$$

is *approximation* with error $\mathcal{O}(\Delta t^2)$ per step. The conditions for the approximation to be useful (small Δt relative to τ_m) are stated.

These two examples differ at the level of an equation that looks structurally similar. The first is a closed-form identity. The second is a numerical approximation. The label captures the difference in two letters.

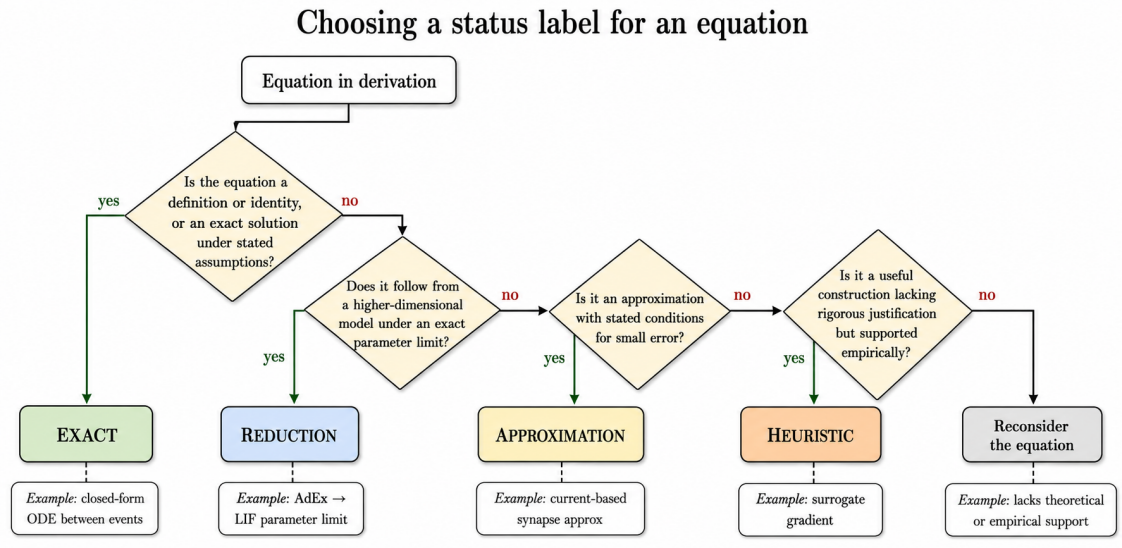

Figure 1. Decision tree for assigning a status label to an equation in a derivation. The four labels are reached by sequential yes/no questions, and a fifth "reconsider" branch covers equations that fall through every test. The labels are not meant to be a formal type system; they are a reading aid. A label tells the reader what to expect from the derivation that follows; it does not replace the derivation itself. Equations that look structurally similar can carry different labels depending on the surrounding context, and the same equation can be *exact* in one role and *approximation* in another.

 doi:10.20944/preprints202605.1130.v1

3. Reduction Chain: From Biophysics to LIF

A spiking neuron model specifies how an internal state evolves and when a spike is emitted. The literature contains a hierarchy of such models, ranging from the four-state Hodgkin-Huxley equations to the one-state LIF equation. This section presents the hierarchy as a reduction chain, with the status of each step labeled. The chain is, in order of increasing simplification:

$$\text{HH} \xrightarrow{\text{adiabatic elimination}} \text{AdEx} \xrightarrow{\text{parameter limit}} \text{LIF} \xleftrightarrow{\text{linearity}} \text{SRM}.$$

We then discuss reset semantics, which sit outside the chain but matter for both spike statistics and the gradient flow used in modern training. Figure 3 summarizes the chain visually.

3.1. Hodgkin-Huxley as the Biophysical Starting Point

The Hodgkin-Huxley model [3] describes the membrane voltage of a single-compartment neuron together with three voltage-gated channel variables. The voltage equation is

$$C_m \frac{dV}{dt} = -\bar{g}_{\text{Na}} m^3 h (V - E_{\text{Na}}) - \bar{g}_K n^4 (V - E_K) - g_L (V - E_L) + I_{\text{ext}}(t), \tag{1}$$

[*exact: voltage equation of a single-compartment HH model.*] together with kinetic equations for the three gating variables,

$$\frac{dx}{dt} = \alpha_x(V)(1 - x) - \beta_x(V) x, \qquad x \in \{m, h, n\}. \tag{2}$$

[*exact: definitions of the gating dynamics. The functions $\alpha_x(V)$, $\beta_x(V)$ are fit from voltage-clamp data and are taken as given.*]

The HH model reproduces the action potential waveform itself. A spike is not an instantaneous event but a stereotyped excursion of V on the millisecond scale. The four-dimensional state (V, m, h, n) is necessary because the activation and inactivation of sodium and potassium channels operate on different time scales and they cannot be summarized by V alone.

For SNN engineering work, the four-state system carries more detail than the task usually requires. The HH model is a useful baseline because every reduction can be referenced back to it, and because its accuracy in matching intracellular recordings is well-documented. For everything that follows, the reader can take HH as the ground-truth biophysical reference.

3.2. From HH to AdEx by Adiabatic Elimination

The adaptive exponential integrate-and-fire model (AdEx) [12,13] keeps the membrane voltage V and one slow adaptation variable w, dropping the fast gating variables. In one common parametrization,

$$C_m \frac{dV}{dt} = -g_L (V - E_L) + g_L \Delta_T \exp\left(\frac{V - V_T}{\Delta_T}\right) - w + I_{\text{ext}}(t), \tag{3}$$

$$\tau_w \frac{dw}{dt} = a(V - E_L) - w, \tag{4}$$

with the spike-and-reset rule: if $V(t^-) \geq V_{\text{cut}}$, then $V(t^+) = V_{\text{reset}}$ and $w(t^+) = w(t^-) + b$.

The exponential term $g_L \Delta_T \exp((V - V_T)/\Delta_T)$ replaces the fast sodium kinetics. The parameter Δ_T controls the sharpness of spike initiation: as $\Delta_T \to 0$, the exponential becomes a step at $V = V_T$, recovering a hard-threshold behavior. The parameter V_T is a soft threshold, and the spike-and-reset rule replaces the explicit modeling of the spike waveform.

The reduction from HH to AdEx is justified by adiabatic elimination of the fast gating variables [13]. Near a stable subthreshold fixed point, m, h, and n relax on time scales much shorter than the membrane time constant τ_m, so they can be treated as functions of V alone. Substituting the steady-state values $m_\infty(V)$, $h_\infty(V)$, $n_\infty(V)$ into (1) produces a one-dimensional equation in V that, after fitting the voltage dependence of the resulting effective conductance, takes the AdEx form with

 doi:10.20944/preprints202605.1130.v1

an exponential nonlinearity. [*approximation: adiabatic elimination is exact in the limit of separated time scales but breaks during spike generation, where V moves on time scales comparable to those of m and h. AdEx therefore reproduces the subthreshold and adaptation behavior of HH but treats the spike as an instantaneous reset event rather than a dynamical excursion.*]

The slow adaptation variable w in (4) captures spike-triggered slowing: each spike adds b to w, and w decays back toward $a(V - E_L)$ on the time scale τ_w. With appropriate parameter choices, AdEx reproduces a wide range of firing patterns (regular firing, bursting, spike-frequency adaptation, initial bursting) using only two state variables [12]. AdEx is the standard compromise between biophysical realism and computational tractability.

3.3. From AdEx to LIF by Parameter Limit

Setting $a = 0$, $b = 0$, and taking $\Delta_T \to 0$ in AdEx removes the adaptation current entirely and replaces the exponential nonlinearity by a hard threshold at V_T. With these substitutions, the voltage equation becomes

$$C_m \frac{dV}{dt} = -g_L(V - E_L) + I_{\text{ext}}(t), \tag{5}$$

[*exact: parameter limit of AdEx with $a = b = 0$, $\Delta_T \to 0$. The LIF model is a special case of AdEx, not an approximation of it.*] together with the threshold-and-reset rule: if $V(t^-) \geq V_{\text{th}}$, then $V(t^+) = V_{\text{reset}}$.

This is the leaky integrate-and-fire model in its biophysical parametrization. We give the more familiar form below.

3.4. The LIF Equation

The LIF model, introduced over a century ago [2], has a single state variable, the membrane potential $V(t)$, and a threshold-and-reset spike mechanism. Starting from (5) and including a generic input current $I_{\text{tot}}(t)$ (which combines synaptic and external currents), and dividing by g_L on both sides, one obtains

$$\tau_m \frac{dV(t)}{dt} = -\big(V(t) - E_L\big) + R_m I_{\text{tot}}(t), \tag{6}$$

[*exact: rewriting of (5) after defining $\tau_m = C_m/g_L$, $R_m = 1/g_L$, and $I_{\text{tot}} = I_{\text{ext}} + I_{\text{syn}}$.*] with the threshold-and-reset rule applied at $V = V_{\text{th}}$ and a spike train written as

$$S(t) = \sum_f \delta(t - t_f). \tag{7}$$

[*exact: definition.*]

Subthreshold solution:

Between spikes, the dynamics in (6) is linear, and the closed-form solution for $t > t_0$ is

$$V(t) = E_L + \big(V(t_0) - E_L\big)e^{-(t-t_0)/\tau_m} + \frac{1}{C_m}\int_{t_0}^{t} e^{-(t-s)/\tau_m} I_{\text{tot}}(s)\, ds. \tag{8}$$

[*exact: closed-form integration of the linear ODE between events.*]

The integral expresses the exponential memory of the LIF neuron: past inputs are discounted on the time scale τ_m, and once a spike resets V, the integration starts over. The exponential memory kernel is one of the defining features of LIF and the structural reason why LIF dynamics are Markov.

Synaptic input:

Synaptic currents are themselves driven by incoming spikes. Let $\kappa(t)$ be a postsynaptic current (PSC) kernel with support on $t \geq 0$. The synaptic current into neuron i from presynaptic neurons $\{j\}$ is

$$I_{\text{syn},i}(t) = \sum_j w_{ij}(\kappa * S_j)(t), \tag{9}$$

 doi:10.20944/preprints202605.1130.v1

[*exact: definition of synaptic input as a linear convolution of presynaptic spike trains with a kernel.*]

The simplest kernel is the single exponential, $\kappa(t) = \tau_s^{-1} e^{-t/\tau_s} H(t)$, with H the Heaviside step. Alpha and double-exponential kernels are alternatives; the choice does not affect the conclusions of the present article.

Discrete-time form:

For numerical work it is convenient to discretize (6). Two forms appear in the literature.

The exact integrating-factor update over a step Δt,

$$V_i[t+1] = E_L + \big(V_i[t] - E_L\big) e^{-\Delta t/\tau_m} + \frac{1}{C_m} \int_t^{t+\Delta t} e^{-(t+\Delta t - s)/\tau_m} I_{\mathrm{tot},i}(s)\, ds, \tag{10}$$

[*exact: integrating-factor solution over* $[t, t+\Delta t]$.] treats the leak exactly and any approximation enters only through the treatment of $I_{\mathrm{tot},i}$ on the interval.

The Euler form,

$$V_i[t+1] = V_i[t] + \frac{\Delta t}{\tau_m}\big(-(V_i[t] - E_L) + R_m I_{\mathrm{tot},i}[t]\big), \tag{11}$$

[*approximation: first-order Euler with error* $\mathcal{O}(\Delta t^2)$ *per step.*] is simpler and is what most modern SNN training pipelines use. The trade-off between (10) and (11) is accuracy versus implementation simplicity. For $\Delta t \leq \tau_m/10$, the Euler form is accurate enough for most purposes, and the exact form is preferred only when long unrolled simulations require the additional accuracy.

3.5. The Spike Response Model as an Exact Reformulation

The spike response model (SRM) [13,14] writes the membrane potential directly as a sum of stereotyped responses to past events,

$$V(t) = V_{\mathrm{rest}} + \sum_{f:\, t_f < t} \eta(t - t_f) + \sum_j w_j \sum_k \epsilon(t - t_{j,k}), \tag{12}$$

[*exact: definition. The kernels* η *and* ϵ *are given (or fit from data), not derived.*] where η is a refractory kernel describing the contribution to V due to the neuron's own past spike at time t_f, and ϵ is a postsynaptic potential (PSP) kernel describing the response to an incoming spike at time $t_{j,k}$. The spike condition is $V(t) \geq \Theta(t)$ for a possibly time-varying threshold $\Theta(t)$.

The SRM is conceptually distinct from LIF in how it presents the dynamics, but for linear subthreshold dynamics the two are equivalent. To see this, take the LIF subthreshold solution (8) between spikes and observe that each spike resets V to V_{reset}. The cumulative effect of the reset can be encoded as a refractory kernel $\eta(\cdot)$ that subtracts the right amount of voltage to bring V to V_{reset} at each spike time, plus an exponential decay back to E_L. With the right choice of η, the integral form (8) becomes the kernel sum (12).

Proposition 1 (LIF-SRM equivalence under linear dynamics)**.** *Let a LIF neuron obey* (6) *with hard reset to* V_{reset} *at threshold crossings. Then there exist kernels* $\eta(\cdot)$ *and* $\epsilon(\cdot)$*, depending only on* τ_m*,* τ_s*,* V_{reset}*, and the choice of synaptic kernel* κ*, such that the membrane potential* $V(t)$ *produced by the LIF dynamics equals the SRM expression* (12) *at every time* t *that is not a spike time.*

[*exact: this is a standard result in computational neuroscience [13]; the proof is a direct calculation using* (8).]

The proposition is exact under linear subthreshold dynamics. If the subthreshold dynamics are nonlinear (as in AdEx with $\Delta_T > 0$), the kernel decomposition breaks because the response to one spike depends on the recent history, not just on time elapsed since the spike. So the LIF-SRM equivalence does not extend to AdEx.

 doi:10.20944/preprints202605.1130.v1

When to use which:

LIF is convenient when one wants a differential-equation simulation, when the synaptic kernel can be folded into a state variable, or when the dynamics need to be analyzed in terms of fixed points and linearizations. SRM is convenient when the kernels can be measured directly from data, when one wants a closed-form expression for $V(t)$ in terms of input spike times, or when developing point process approximations (Section 5) where the kernel form is the natural starting point. The two are mathematically equivalent under linear subthreshold dynamics; the choice between them is one of convenience.

3.6. Reset Semantics

The threshold-and-reset rule was introduced as a single phrase in (6): "if $V(t^-) \geq V_{\text{th}}$ then $V(t^+) = V_{\text{reset}}$." Three variants of this rule appear in the literature, and the differences matter for both spike statistics and gradient flow during training.

Hard reset:

After a threshold crossing, the membrane potential is set to a fixed value:

$$V(t^+) = V_{\text{reset}}. \tag{13}$$

[*exact: definition.*] Any information about how far $V(t^-)$ exceeded V_{th} is discarded. This is the rule most often associated with the textbook LIF model and the rule that is implicit in the SRM derivation in Section 3.5.

Soft (subtractive) reset:

The threshold value is subtracted from the membrane potential:

$$V(t^+) = V(t^-) - (V_{\text{th}} - V_{\text{reset}}). \tag{14}$$

[*exact: definition.*] The amount by which $V(t^-)$ overshot the threshold is preserved. Soft reset is the default in many modern SNN frameworks because it allows information to flow through the spike event rather than being discarded. It is also closer to the behavior of a perfect integrator that simply subtracts a unit charge at each spike.

No reset:

The membrane potential is left unchanged at the spike, and a refractory mechanism (or the leak alone) brings it back down. This option is rare in current SNN practice but appears in some integrate-and-fire models for analytical convenience.

Effect on dynamics:

For a single neuron with constant input I, hard reset and soft reset produce different firing rates. Under hard reset, the membrane resets to V_{reset} after each spike, and the next interspike interval is the time to integrate from V_{reset} to V_{th}. Under soft reset, the membrane resets to $V_{\text{th}} - (V_{\text{th}} - V_{\text{reset}}) + (V(t^-) - V_{\text{th}})$, which is $V_{\text{reset}} + (V(t^-) - V_{\text{th}}) \geq V_{\text{reset}}$. The next interspike interval is shorter under soft reset, by an amount that depends on the overshoot. For high-rate inputs, the difference accumulates and the firing rates can differ noticeably.

Effect on gradients:

The gradient of the discrete-time membrane recurrence with respect to the past is

$$\frac{\partial V[t+1]}{\partial V[t]} = \alpha - (V_{\text{th}} - V_{\text{reset}}) \frac{\partial s[t]}{\partial V[t]}, \tag{15}$$

 doi:10.20944/preprints202605.1130.v1

where $\alpha = e^{-\Delta t/\tau_m}$ and $s[t] = H(V[t] - V_{\text{th}})$. Under hard reset implemented as a clamp ($V[t+1] \leftarrow V_{\text{reset}}$ when $s[t] = 1$), the recurrence is non-differentiable at the spike, and the gradient is set to zero whenever $s[t] = 1$. Under soft reset implemented as a subtraction ($V[t+1] = V[t+1] - (V_{\text{th}} - V_{\text{reset}})s[t]$), the recurrence is smooth in $V[t]$ everywhere except where the surrogate gradient is needed (at the spike threshold itself), and the gradient is propagated cleanly through the recurrence. [*exact: direct differentiation of the discrete update.*] For backpropagation through time over many steps, soft reset preserves long-range gradient information and hard reset truncates it at every spike event. This is one of the main reasons why soft reset has become the default in modern SNN training pipelines.

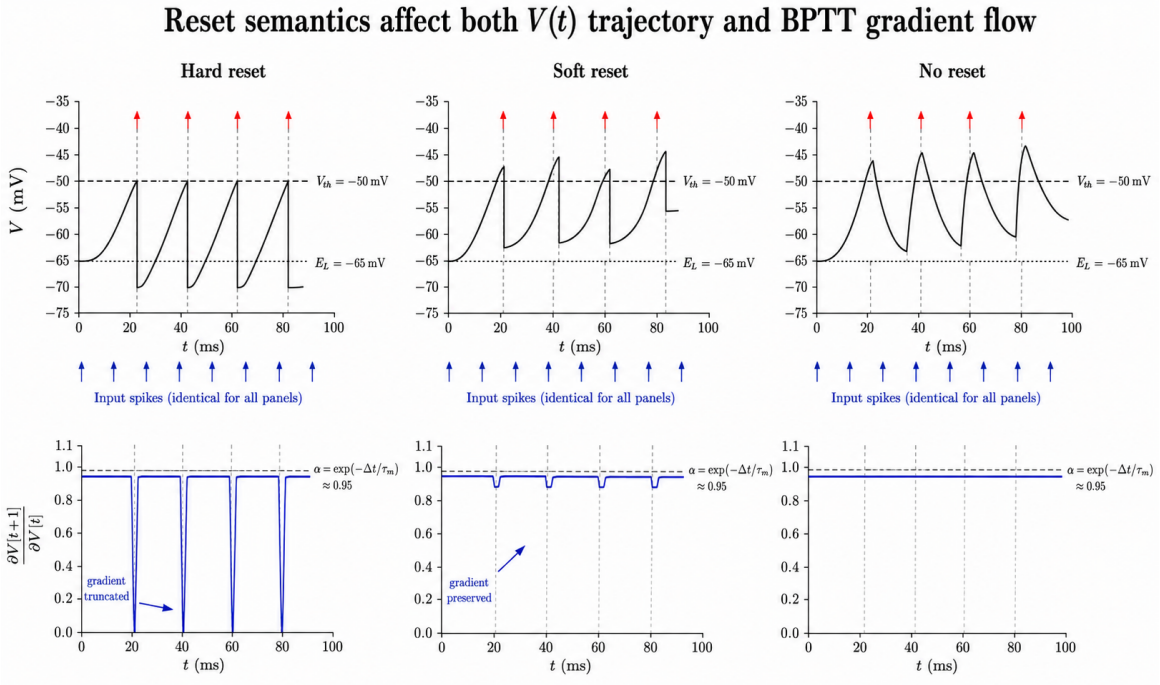

Figure 2. Reset semantics affect both the membrane trajectory $V(t)$ and the per-step BPTT factor $\partial V[t+1]/\partial V[t]$. Top row: $V(t)$ trajectories under three reset rules driven by an identical input spike train. Bottom row: the corresponding per-step gradient factor as a function of time. Under hard reset (left), the gradient drops to zero at every spike, so BPTT through a long sequence accumulates a product that is zero at every reset. Under soft reset (middle), the gradient stays close to the leak factor $\alpha = e^{-\Delta t/\tau_m}$ across spikes; long-range gradient information is preserved. Under no reset (right), the gradient is at α throughout. The hard-reset truncation is the main reason soft reset has become the default in deep SNN training.

3.7. The Reduction Chain Visualized

The reduction chain in Figure 3 is not the only way to derive LIF, but it is the cleanest one in the sense that each step has a stated status. The reader who comes across an alternative derivation in another paper can ask: which step is being made? Is the corresponding approximation justified in this paper's setting? Is it an exact limit or a working heuristic? The answer is often visible from a careful reading.

 doi:10.20944/preprints202605.1130.v1

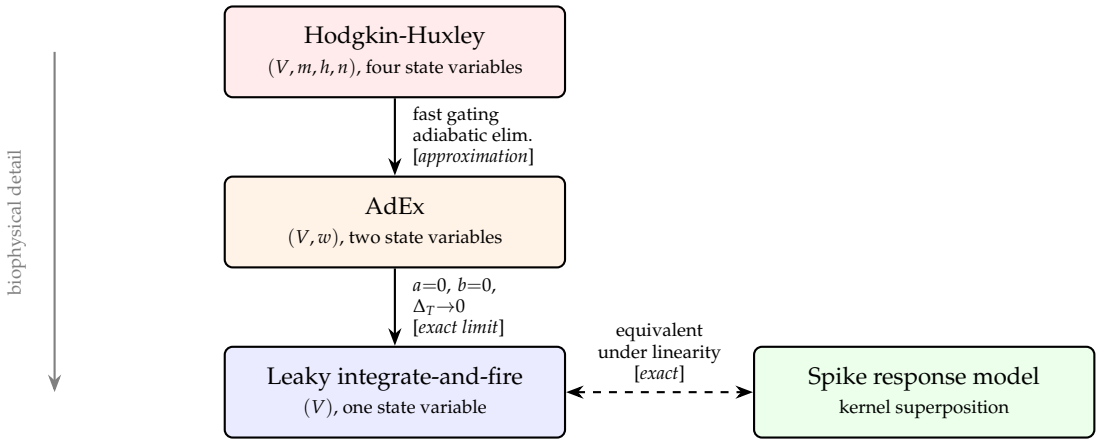

Figure 3. The reduction chain from Hodgkin-Huxley to LIF, with status labels at each step. The HH-to-AdEx reduction is an approximation; it is exact only in the limit of separated time scales between the membrane and the gating variables. The AdEx-to-LIF reduction is an exact parameter limit. The LIF-SRM correspondence is exact under linear subthreshold dynamics. The chain breaks at the HH-to-AdEx step during spike generation, where the approximation is no longer valid; this is why the spike is modeled as an instantaneous reset rather than a dynamical excursion in AdEx and LIF.

4. Network Dynamics

A spiking neural network is a system of coupled differential equations, one for each neuron, with a threshold-and-reset rule applied independently to each. This section develops the matrix form of the network dynamics, the analytical theory of recurrent SNNs (liquid state machines, the echo state property), and the balanced-network regime that underlies a large fraction of computational neuroscience.

4.1. Coupled Spiking Dynamics in Matrix Form

Consider a network of n LIF neurons. The subthreshold dynamics of neuron i is, from (6) and (9),

$$\tau_m \frac{dV_i(t)}{dt} = -\big(V_i(t) - E_L\big) + R_m \sum_{j=1}^{n} w_{ij}(\kappa * S_j)(t) + R_m I_{\text{ext},i}(t), \tag{16}$$

[*exact: direct extension of* (6) *with synaptic input from* (9).] with the threshold-and-reset rule applied to each neuron independently. With transmission delays, the contribution of neuron j becomes $w_{ij}(\kappa * S_j)(t - d_{ij})$.

When the synaptic kernel is the single exponential, $\kappa(t) = \tau_s^{-1} e^{-t/\tau_s} H(t)$, the synaptic input itself satisfies a first-order ODE. Defining $u_i(t) = \sum_j w_{ij}(\kappa * S_j)(t)$, one has

$$\tau_s \frac{du_i(t)}{dt} = -u_i(t) + \sum_j w_{ij} S_j(t). \tag{17}$$

[*exact: differentiating the convolution with an exponential kernel.*]

Writing $\mathbf{V}(t) = (V_1(t), \ldots, V_n(t))^\top$ and $\mathbf{u}(t) = (u_1(t), \ldots, u_n(t))^\top$, and taking $E_L = 0$ and $R_m = 1$ for notational convenience, the network dynamics becomes

$$\tau_m \frac{d\mathbf{V}(t)}{dt} = -\mathbf{V}(t) + \mathbf{u}(t) + \mathbf{I}_{\text{ext}}(t), \tag{18}$$

$$\tau_s \frac{d\mathbf{u}(t)}{dt} = -\mathbf{u}(t) + \mathbf{W}\mathbf{S}(t), \tag{19}$$

[*exact: matrix form of the network dynamics with exponential synapses.*] where $\mathbf{W} \in \mathbb{R}^{n \times n}$ is the weight matrix with $(\mathbf{W})_{ij} = w_{ij}$, and $\mathbf{S}(t) = (S_1(t), \ldots, S_n(t))^\top$ is the vector of spike trains. The threshold-and-reset rule is applied componentwise.

 doi:10.20944/preprints202605.1130.v1

This is a state-space representation of the network with state $(\mathbf{V}, \mathbf{u}) \in \mathbb{R}^{2n}$. The dynamics is linear between spikes, and nonlinear at spike events through the threshold and reset. The rest of this section analyzes the implications. Figure 4 gives a block-diagram view of the same dynamics.

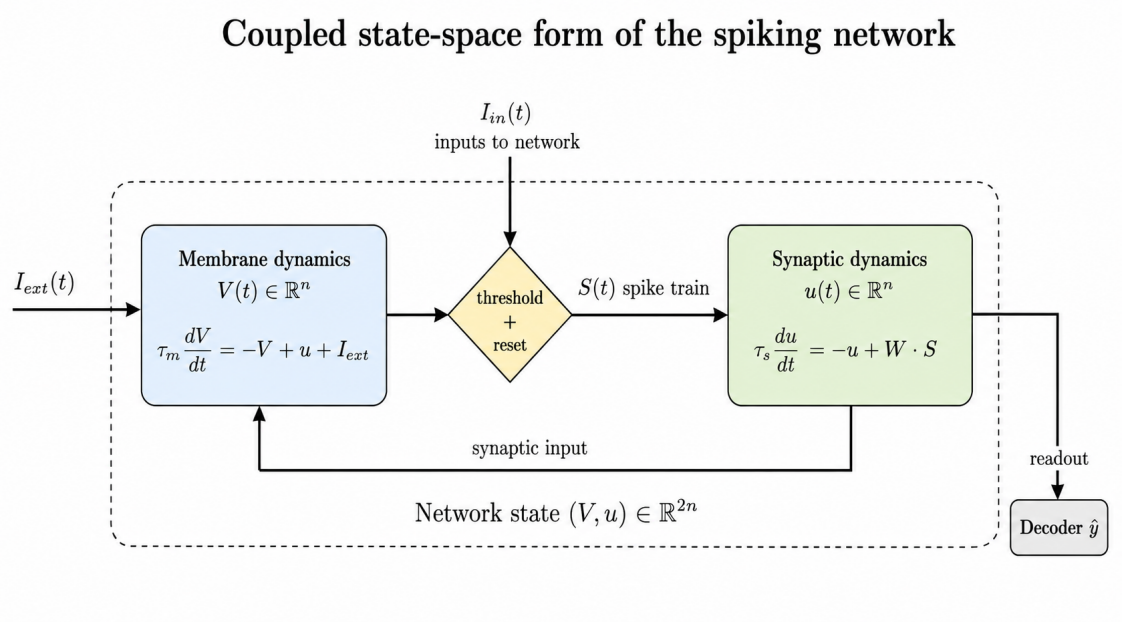

Figure 4. Coupled state-space form of the spiking network. The membrane dynamics block (left) integrates synaptic input $\mathbf{u}(t)$ and external current $\mathbf{I}_{\text{ext}}(t)$ into the membrane state $\mathbf{V}(t)$. The threshold-and-reset operator (center) produces a binary spike train $\mathbf{S}(t)$ from $\mathbf{V}(t)$. The synaptic dynamics block (right) filters the spike train through the weight matrix $\mathbf{W}$ and the synaptic time constant to produce the next-step synaptic input. The dashed enclosure marks the network state $(\mathbf{V}, \mathbf{u}) \in \mathbb{R}^{2n}$ and the recurrence loop. A readout decoder converts the spike train to a prediction $\hat{y}$. The same equations are written in (18)–(19).

4.2. Feedforward Networks and Layer-By-Layer Integration

In a feedforward network with L layers, the weight matrix has a block lower-triangular structure: $w_{ij} \neq 0$ only if neuron i is in a layer downstream of neuron j. For such a network, (18)–(19) can be solved layer by layer because the input to layer ℓ depends only on the spike trains of layers $1, \ldots, \ell - 1$. The dynamics is well-posed for any input, and the spike trains can be computed in a single forward pass.

In particular, the spike times of layer ℓ are determined by the spike times of layer $\ell - 1$, so a deep feedforward SNN with T time steps and L layers can be unrolled into a directed acyclic graph of size $\mathcal{O}(LT)$ for backpropagation. This is the structure that surrogate gradient training exploits.

4.3. Recurrent SNNs and the Echo State Property

When the weight matrix is not block lower-triangular, the network has feedback. Recurrent SNNs can store information over extended time windows and approximate non-Markov input-output mappings. They are also harder to analyze and harder to train. The two main analytical frameworks for recurrent SNNs are the liquid state machine [15] and the echo state property [16].

4.3.1. Liquid State Machines

A liquid state machine (LSM) consists of a recurrent network of spiking neurons (the liquid) driven by time-varying input. The liquid's state $\mathbf{x}(t) \in \mathbb{R}^{2n}$ at time t is the vector of all membrane potentials and synaptic states. A simple readout, typically a linear map of $\mathbf{x}(t)$ followed by a threshold or a softmax, is trained on the liquid state to perform a task. The internal weights of the liquid are not learned; they are drawn from a fixed distribution.

The key analytical property of an LSM is the *separation property*: the liquid should map different input histories to different states, so that a readout can distinguish them. Formally, given two input

 doi:10.20944/preprints202605.1130.v1

histories $\mathbf{u}_1(\cdot)$ and $\mathbf{u}_2(\cdot)$ that differ over a finite time window, the resulting liquid states $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ should be different. The closer the differing inputs are mapped to nearby liquid states, the harder the readout's task.

Maass and colleagues showed [15]:

Theorem 1 (Universal approximation by an LSM, informal statement)**.** *Under mild conditions on the connectivity and the neuron model, an LSM with sufficiently many neurons can approximate any time-invariant filter with fading memory to arbitrary accuracy.*

[*exact: stated and proved in [15] under the conditions: (i) the neuron model has the point-wise separation property; (ii) the readout class is universal for static functions on the liquid state space.*]

The fading memory condition requires that the influence of past inputs decay over time, which is related to the spectral properties of the network's weight matrix and the leak time constants. If the network operates near the boundary between stable and chaotic regimes (the "edge of chaos"), it can maintain long transients while still being robust. The edge-of-chaos regime has been studied extensively [17–19]; the terminology is suggestive but the rigorous dynamical-systems content is more limited than the name implies.

4.3.2. The Echo State Property

The echo state property (ESP) [16] originated in the continuous-valued reservoir computing literature. For a discrete-time recurrent network with state update $\mathbf{x}[t+1] = f(\mathbf{x}[t], \mathbf{u}[t])$, the ESP states that the network's state asymptotically depends only on the input history and not on the initial conditions.

Definition 1 (Echo state property)**.** *A recurrent network has the echo state property if, for any two initial states* $\mathbf{x}_1[0], \mathbf{x}_2[0]$ *and any bounded input sequence* $\{\mathbf{u}[t]\}_{t\geq 0}$*, the resulting state trajectories satisfy*

$$\|\mathbf{x}_1[t] - \mathbf{x}_2[t]\| \to 0 \quad \text{as } t \to \infty.$$

For a linear recurrent network $\mathbf{x}[t+1] = \mathbf{W}_{\text{rec}}\mathbf{x}[t] + \mathbf{W}_{\text{in}}\mathbf{u}[t]$ with $\mathbf{W}_{\text{rec}} \in \mathbb{R}^{n\times n}$, the ESP holds if and only if the spectral radius $\rho(\mathbf{W}_{\text{rec}}) < 1$. [*exact: standard result for linear time-invariant systems.*] For nonlinear systems, the spectral radius criterion is necessary but not sufficient; the standard sufficient condition is that the largest singular value of $\mathbf{W}_{\text{rec}}$ be less than one, which gives a contraction in the Euclidean metric. [*exact: sufficient condition for contraction maps.*]

For a spiking reservoir, an analogous condition can be formulated. Let $\mathbf{V}_1$ and $\mathbf{V}_2$ be two trajectories of (18)–(19) starting from different initial states with the same input. The spiking ESP requires that $\|\mathbf{V}_1(t) - \mathbf{V}_2(t)\| \to 0$ as $t \to \infty$. The threshold-and-reset rule complicates the analysis because the spike events of $\mathbf{V}_1$ and $\mathbf{V}_2$ may differ, and the trajectories of the spike trains are themselves what feed back through $\mathbf{W}$.

A sufficient condition for the spiking ESP is that the recurrent dynamics is contractive in expectation between spike events, and the perturbation introduced by differing spike times decays on a time scale shorter than the input correlations. A clean, fully rigorous condition does not yet appear in the literature for the general spiking case. For the linearized subthreshold dynamics, the spectral-radius condition $\rho(\mathbf{W}_{\text{rec}})e^{-\bar{T}/\tau_m} < 1$ is necessary, where $\bar{T}$ denotes the mean interspike interval of the recurrent population (not the simulation step Δt). This is a useful heuristic for practical reservoir design, even if a rigorous theorem is lacking. [*heuristic: linearized spectral-radius criterion; rigorous extension to spiking dynamics is open.*]

4.4. Balanced Excitatory-Inhibitory Networks

Cortical circuits operate in a regime of approximately balanced excitation and inhibition: each neuron receives strong excitatory input that is approximately canceled by strong inhibitory input, and

 doi:10.20944/preprints202605.1130.v1

the residual fluctuations drive the spike firing. The balanced-network model of van Vreeswijk and Sompolinsky [20] is the standard analytical framework for this regime, refined by Brunel [21].

Consider a network of N_E excitatory and N_I inhibitory LIF neurons, with sparse random connectivity. Each neuron receives input from $K \ll N$ presynaptic neurons (a fixed in-degree), and the synaptic strength scales as $1/\sqrt{K}$. In the limit $N, K \to \infty$ with $K/N \to 0$, the membrane potential of each neuron undergoes Gaussian fluctuations whose mean is approximately zero and whose variance is $\mathcal{O}(1)$.

Theorem 2 (Balanced regime, informal statement)**.** *In a sparsely connected network of LIF neurons with synaptic strength scaling as $1/\sqrt{K}$, where K is the in-degree, the network operates in a balanced regime where each neuron's membrane potential is driven by approximately Gaussian fluctuations with $\mathcal{O}(1)$ variance and a mean that is canceled to leading order. The firing rate of the population is determined self-consistently by the balance between excitatory and inhibitory drive.*

[*exact in the limit: stated and proved in [20,21] for the random sparse network with $1/\sqrt{K}$ scaling.*]

The balanced regime has a number of consequences. The interspike intervals are approximately Poisson, so the spike trains have a coefficient of variation close to one (matching cortical recordings). The network is in an asynchronous irregular state for a wide range of parameters, and is robust to perturbations. The state can be characterized by mean-field equations that relate the population firing rate to the input statistics.

For SNN engineering work, the balanced regime is the natural operating point for randomly connected reservoirs and for biologically motivated recurrent networks. For supervised SNN training with surrogate gradients, the balanced regime is rarely entered explicitly, while structured weight initialization moves the network into a more controlled regime where individual neurons fire at moderate rates without strong inhibition. The mean-field analysis remains a useful diagnostic, however, when a trained network is examined for its emergent properties.

4.5. Stability and Gradient Flow in Deep SNNs

For deep SNN training with surrogate gradient methods, the question is not just whether the dynamics is stable but whether gradients can propagate through many layers and time steps. The relevant quantity is the product of derivative factors (15) along the unrolled computation graph. For a feedforward SNN unrolled over T time steps and L layers, the gradient with respect to a weight in the first layer involves a product of LT such factors, and the magnitude of this product determines whether the gradient vanishes or explodes.

A stability analysis of surrogate gradient flow in deep SNNs is incomplete. Empirically, networks trained with arctan or fast-sigmoid surrogates and soft reset can be trained stably to depths of 8–16 layers and $T = 100$ time steps without specialized initialization. Beyond this, batch normalization, residual connections, and careful weight initialization become necessary. The analytical question of why arctan and fast-sigmoid surrogates produce more stable gradient flow than sigmoid surrogates remains open. [*heuristic: empirical observation [22,23]; no rigorous stability theorem for surrogate gradient flow in deep SNNs is currently known.*]

5. Spike Trains as Point Processes

The point process formulation treats a spike train as a stochastic counting measure on the time axis. This formulation is the natural setting for spike train analysis (estimation, inference, model criticism), for connecting SNNs to the broader stochastic process literature, and for proving certain theoretical results that the differential-equation formulation does not naturally support. We develop the formulation here in some depth, including the time-rescaling theorem, the general history-dependent likelihood, and the canonical model classes.

 doi:10.20944/preprints202605.1130.v1

5.1. Spike Trains as Counting Measures

Let Ω be a probability space and let $T_1 < T_2 < \cdots$ be a sequence of random spike times. The associated counting process is

$$N(t) = \sum_{k \geq 1} \mathbf{1}\{T_k \leq t\}. \tag{20}$$

[*exact: definition.*] $N(t)$ is right-continuous, takes nonnegative integer values, and increases by one at each T_k.

A point process is fully characterized by the joint distribution of $(T_1, T_2, \dots)$ or, equivalently, by the law of $N(\cdot)$. For most processes of interest, the law is more conveniently specified by the conditional intensity, which we develop next.

5.2. Conditional Intensity and History Filtration

Let $\mathcal{H}_t$ denote the history of the process up to time t^-: the σ-algebra generated by $\{N(s) : s < t\}$ and any other observed covariates. The conditional intensity of the process is

$$\lambda(t \mid \mathcal{H}_t) = \lim_{\Delta \to 0^+} \frac{\Pr\{N(t+\Delta) - N(t^-) = 1 \mid \mathcal{H}_t\}}{\Delta}, \tag{21}$$

[*exact: definition; the limit exists for orderly point processes [5].*] when the limit exists. Intuitively, $\lambda(t \mid \mathcal{H}_t)\, dt$ is the probability of observing a spike in $[t, t+dt)$ given the past. For a population of neurons indexed by $i = 1, \dots, n$, we write $\lambda_i(t \mid \mathcal{H}_t)$, and the joint process is characterized by the vector of intensities.

The conditional intensity is a complete characterization in the following sense.

Theorem 3 (Characterization by conditional intensity [5])**.** *Two simple point processes (i.e., processes with at most one event at any time) with the same conditional intensity have the same law.*

[*exact: standard result in point process theory.*]

Special cases. If $\lambda(t \mid \mathcal{H}_t) = \lambda$ (a constant), the process is homogeneous Poisson. If $\lambda(t \mid \mathcal{H}_t) = \lambda(t)$ (depends only on time), the process is inhomogeneous Poisson. If $\lambda(t \mid \mathcal{H}_t)$ depends on the past spike times, the process is history-dependent (self-exciting if the dependence is increasing, self-inhibiting if decreasing).

5.3. The Time-Rescaling Theorem

Define the integrated conditional intensity, also called the compensator,

$$\Lambda(t) = \int_0^t \lambda(s \mid \mathcal{H}_s)\, ds. \tag{22}$$

[*exact: definition.*]

The time-rescaling theorem says that, after rescaling time by the compensator, the process becomes a homogeneous Poisson process with rate one.

Theorem 4 (Time-rescaling theorem [24,25])**.** *Let $\{T_k\}$ be a simple point process with continuous conditional intensity $\lambda(t \mid \mathcal{H}_t) > 0$ on $[0, T^*]$. Define*

$$z_k = \Lambda(T_k) - \Lambda(T_{k-1}) = \int_{T_{k-1}}^{T_k} \lambda(s \mid \mathcal{H}_s)\, ds, \tag{23}$$

with $T_0 = 0$. Then $\{z_k\}$ is a sequence of independent and identically distributed exponential random variables with mean one. Equivalently, $u_k = 1 - \exp(-z_k)$ is i.i.d. uniform on $(0, 1)$.

[*exact: standard result; the proof uses a change of time variable and the hazard transform [24].*]

 doi:10.20944/preprints202605.1130.v1

The time-rescaling theorem is the foundation for goodness-of-fit testing of point process models. Given an estimated $\hat{\lambda}(t \mid \mathcal{H}_t)$, one computes $\hat{z}_k$ and tests whether the resulting sequence is consistent with i.i.d. exponentials of mean one (equivalently, whether $\hat{u}_k$ is uniform). Kolmogorov-Smirnov tests on $\hat{u}_k$ and quantile-quantile (Q-Q) plots are the standard diagnostics. The rescaling construction is illustrated in Figure 5.

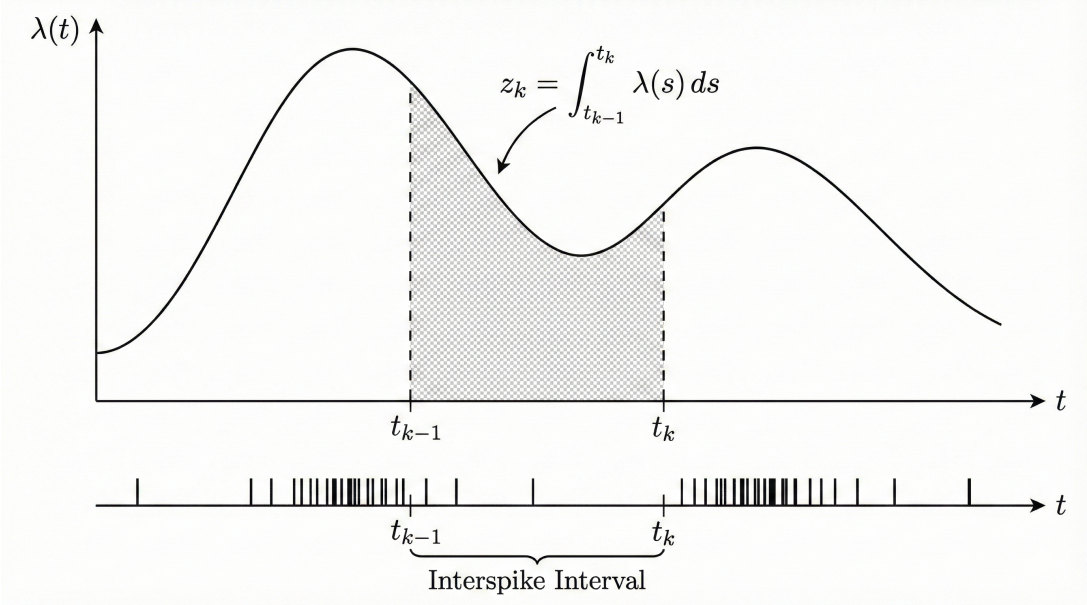

Figure 5. Time-rescaling illustration. Top: an inhomogeneous conditional intensity $\lambda(t)$ with three observed spike times T_1, T_2, T_3. Middle: the corresponding integrated intensity $\Lambda(t)$, which is monotone increasing. Bottom: the rescaled spike times $\Lambda(T_k)$, on which the process becomes a homogeneous Poisson process of rate one. The rescaled intervals $z_k = \Lambda(T_k) - \Lambda(T_{k-1})$ are i.i.d. exponential with mean one (Theorem 4).

5.4. Likelihood for General History-Dependent Point Processes

Given the conditional intensity, the joint likelihood of a sequence of spike times has a clean form.

Theorem 5 (Likelihood of a history-dependent point process)**.** *Let $\{T_1, \ldots, T_n\}$ be the observed spike times of a simple point process on $[0, T^*]$, with conditional intensity $\lambda(t \mid \mathcal{H}_t)$ depending on parameters θ. The likelihood is*

$$L(\theta) = \left[\prod_{k=1}^{n} \lambda(T_k \mid \mathcal{H}_{T_k}; \theta)\right] \exp\left(-\int_0^{T^*} \lambda(s \mid \mathcal{H}_s; \theta)\, ds\right). \tag{24}$$

The log-likelihood is

$$\ell(\theta) = \sum_{k=1}^{n} \log \lambda(T_k \mid \mathcal{H}_{T_k}; \theta) - \int_0^{T^*} \lambda(s \mid \mathcal{H}_s; \theta)\, ds. \tag{25}$$

[*exact: standard result obtained by multiplying conditional hazards between events with survival factors on the interspike intervals [6,26].*]

For a multivariate process with neurons $i = 1, \ldots, n$, the joint log-likelihood sums (25) across neurons, with the conditional intensity $\lambda_i(t \mid \mathcal{H}_t; \theta)$ for each neuron now possibly depending on the spike trains of all neurons.

Maximum likelihood estimation:

Under standard regularity conditions, the maximum likelihood estimator (MLE) $\hat{\theta} = \arg\max \ell(\theta)$ is consistent and asymptotically normal with covariance matrix $\mathcal{I}(\theta^*)^{-1}$, where $\mathcal{I}$ is the Fisher information matrix [26]. The MLE is the standard tool for fitting point process models to spike train data.

 doi:10.20944/preprints202605.1130.v1

When several competing models are fit to the same spike data, approximate model selection can be done with information criteria,

$$\mathrm{AIC} = -2\ell(\hat{\theta}) + 2p, \qquad \mathrm{BIC} = -2\ell(\hat{\theta}) + p \log n, \tag{26}$$

[*exact: standard definitions; AIC is consistent for predictive accuracy under specific assumptions, BIC for model identification.*] where p is the number of parameters and n is the number of observed spikes.

5.5. Canonical Model Classes

We list the four most common point process models for neural data.

Homogeneous Poisson:

The conditional intensity is constant, $\lambda(t \mid \mathcal{H}_t) = \lambda$. Interspike intervals are i.i.d. exponential with mean $1/\lambda$. This is the simplest baseline and corresponds to a memoryless spike-generating process.

Inhomogeneous Poisson:

The intensity depends only on time, $\lambda(t \mid \mathcal{H}_t) = \lambda(t)$. Interspike intervals are no longer i.i.d., but the process is still memoryless given the time. This is the appropriate model when an external stimulus modulates the firing rate but the spike history does not influence future spikes.

Hawkes self-exciting process:

The intensity depends on the past spike times of the same neuron and possibly other neurons,

$$\lambda_i(t \mid \mathcal{H}_t) = \mu_i + \sum_j \int_0^{t^-} \phi_{ij}(t-s)\, dN_j(s), \tag{27}$$

[*exact: definition of a linear Hawkes process [27].*] with nonnegative kernels ϕ_{ij} and base rates μ_i. Stationarity holds when the spectral radius of the matrix with entries $\int_0^\infty \phi_{ij}(\tau)\, d\tau$ is less than one. The mean rate vector $\bar{\boldsymbol{\lambda}}$ then solves $(\mathbf{I} - \boldsymbol{\Phi}_1)\bar{\boldsymbol{\lambda}} = \boldsymbol{\mu}$ with $(\boldsymbol{\Phi}_1)_{ij} = \int_0^\infty \phi_{ij}(\tau)\, d\tau$ [5]. The classical Hawkes model requires nonnegative kernels, which means it cannot directly represent inhibitory synaptic connections; nonlinear Hawkes extensions [28] exist but lose the convenient linear structure.

Point-process generalized linear model (PP-GLM):

The intensity is the output of a linear-nonlinear cascade,

$$\lambda_i(t \mid \mathcal{H}_t) = g\left(c_i + (k_i * \mathbf{x})(t) + \sum_j (h_{ij} * dN_j)(t) \right), \tag{28}$$

[*exact: definition of a PP-GLM [26,29,30].*] where $\mathbf{x}(t)$ is a vector of exogenous covariates, k_i is a stimulus filter, h_{ij} is a spike-history kernel, c_i is a bias, $*$ denotes convolution, and g is a positive link function (commonly $g = \exp$). With $g = \exp$, the log-likelihood is concave in the parameters under standard bases for k_i and h_{ij}, which allows globally optimal MLE via convex programming. The PP-GLM has become the standard model class for neural encoding analysis [30,31].

Escape-rate spiking neuron:

A bridge to the dynamical-systems formulation: take a LIF (or SRM) neuron with subthreshold potential $V(t)$, and replace the hard threshold by a soft hazard,

$$\lambda(t \mid \mathcal{H}_t) = \rho \exp(\beta(V(t) - \theta)), \tag{29}$$

[*exact: definition of an escape-rate model [14].*] with scale $\rho > 0$, slope $\beta > 0$, and soft threshold θ. As $\beta \to \infty$, the model approaches a hard threshold. The escape-rate model embeds biophysically meaningful

 doi:10.20944/preprints202605.1130.v1

state variables inside a probabilistic spike generator, which is a clean way to add stochasticity to a LIF neuron without leaving the point process framework.

5.6. Bridging State-Space SNNs and Intensity-Based Formulations

The point process view and the dynamical systems view are compatible rather than competing. We make the bridge explicit in two directions.

From state-space to intensity:

A state-space SNN defines the subthreshold dynamics for $V_i(t)$ and prescribes spikes by a threshold rule. Introducing noise in input currents or thresholds, then marginalizing the unobserved noise, induces an effective conditional intensity of the form (29) for each neuron [14]. The escape-rate parameterization is the cleanest example: a noisy LIF neuron with threshold noise of standard deviation σ and a fast-decaying hazard becomes, in the limit of small σ and a sharp threshold, a deterministic LIF; in the limit of moderate σ and a soft threshold, a PP-GLM with $V(t)$ as the linear predictor.

From intensity to state-space:

Conversely, when the spike-history kernels h_{ij} in a PP-GLM (28) can be expressed as outputs of a finite-dimensional linear time-invariant filter, the convolutional spike history admits a state augmentation that rewrites it as a set of first-order ODEs driven by spikes. This embeds the GLM inside a linear SNN with stochastic spike generation. The state augmentation does not work for arbitrary kernel shapes; it requires that the kernel family be realizable by a finite-dimensional state space. The exponential, alpha function, and raised-cosine basis families that are common in practice all satisfy this requirement [30]. [*exact: standard state-space realization theory; the kernel-to-state-space embedding is constructive.*]

The bridge means that one can choose between writing an SNN as a coupled ODE system or as a vector of conditional intensities, depending on which side of the analysis is more convenient. The two formulations are different presentations of the same mathematical object, under the assumption that the kernels satisfy the realizability condition.

5.7. Diagnostics: Model Criticism for Point Process Spike-Train Models

The exponential structure induced by the true intensity (Theorem 4) is the basis for goodness-of-fit testing. Two diagnostics are standard.

Time-rescaling diagnostic:

After fitting an estimate $\hat{\lambda}$ to the data, compute $\hat{z}_k = \int_{T_{k-1}}^{T_k} \hat{\lambda}(s)\, ds$. Here, if the model is correctly specified, the transformed variables $\hat{u}_k = 1 - e^{-\hat{z}_k}$ are i.i.d. uniform on $(0, 1)$. The Kolmogorov-Smirnov statistic on $\{\hat{u}_k\}$ tests this hypothesis. A Q-Q plot of $\hat{u}_k$ against the uniform quantiles is a visual version of the same test [24].

Thinning diagnostic:

Each spike is deleted independently with a computed probability proportional to $\hat{\lambda}$, and the residual process is tested for being a homogeneous Poisson process with constant rate [25]. The thinning diagnostic is more sensitive to local structure (clustering, refractoriness) than the time-rescaling diagnostic, which integrates over longer windows and can miss local violations.

Both diagnostics detect overfitting to rates while sometimes missing dependence in history or coupling. A third class of diagnostics, based on residual analysis of the decoded spike-history kernel, is more sensitive to misspecification of h_{ij} but requires more careful interpretation; we do not develop it here.

 doi:10.20944/preprints202605.1130.v1

6. Information-Theoretic Aspects of Spike Coding

The point process formulation provides a clean entry point to information theory of spike trains. Mutual information, Fisher information, and channel capacity have been used to compare neural codes, to bound the precision of neural decoding, and to analyze the energetic cost of information transmission [32,33]. We treat the part most relevant to SNN engineering: a quantitative comparison of rate and time-to-first-spike codes.

6.1. Fisher Information for Spike Coding

Let a stimulus $x \in \mathbb{R}$ be encoded by a population of n neurons, each producing a spike train. The Fisher information of the stimulus, given the observed spike trains $\mathbf{S}_{[0,T]}$, is

$$\mathcal{I}(x) = -\mathbb{E}\left[\frac{\partial^2 \log p(\mathbf{S}_{[0,T]} \mid x)}{\partial x^2} \,\middle|\, x\right]. \tag{30}$$

[*exact: definition.*]

Under standard regularity conditions, the Cramér-Rao lower bound states that any unbiased estimator $\hat{x}$ of x satisfies $\mathrm{Var}(\hat{x}) \geq 1/\mathcal{I}(x)$. The Fisher information is thus a measure of how precisely x can be decoded from the observed spike trains.

6.2. Rate Code Fisher Information

Suppose each neuron i produces an inhomogeneous Poisson process with rate $\lambda_i(t; x)$ that depends on x through some tuning curve, and the rates are constant over the observation window $[0, T]$, so $\lambda_i(t; x) = f_i(x)$. The log-likelihood of the spike count n_i in the window is

$$\log p(n_i \mid x) = n_i \log f_i(x) - f_i(x)T - \log(n_i!). \tag{31}$$

[*exact: log-likelihood of a Poisson count given the rate.*]

Differentiating twice with respect to x and taking the expectation, the Fisher information from neuron i is

$$\mathcal{I}_i^{\text{rate}}(x) = T\frac{(f_i'(x))^2}{f_i(x)}, \tag{32}$$

[*exact: standard derivation from the Poisson log-likelihood.*] where $f_i'(x) = df_i/dx$ is the slope of the tuning curve. The total population Fisher information is the sum over neurons under the independence assumption,

$$\mathcal{I}_{\text{pop}}^{\text{rate}}(x) = T\sum_{i=1}^{n}\frac{(f_i'(x))^2}{f_i(x)}. \tag{33}$$

[*exact: under conditional independence of the spike counts given x, which holds for inhomogeneous Poisson without coupling.*]

The rate-code Fisher information scales linearly with the observation window T. Doubling T doubles the Fisher information, halves the variance lower bound, and reduces the standard error by $\sqrt{2}$. This is the standard argument for why rate codes are slow.

6.3. Time-to-first-spike Fisher Information

For TTFS coding, each neuron produces a single spike whose latency τ_i encodes the stimulus. Suppose the latency is generated by an inhomogeneous Poisson with intensity $\lambda_i(t; x)$, so the latency density is

$$p(\tau_i \mid x) = \lambda_i(\tau_i; x)\exp\left(-\int_0^{\tau_i}\lambda_i(s; x)\,ds\right). \tag{34}$$

[*exact: standard hazard formulation; the survival factor is the probability of no spike in $[0, \tau_i]$.*]

For a deterministic LIF neuron under constant input, the latency in (34) concentrates on the value $\bar{\tau}(I) = -\tau_m \log(1 - (V_{\text{th}} - E_L)/(R_m I))$ obtained by setting $V(\bar{\tau}) = V_{\text{th}}$ in (8), and the encoding

 doi:10.20944/preprints202605.1130.v1

becomes deterministic. We instead consider the noisy case where small jitter is added to the firing time. If the latency variance is $\sigma_\tau^2(x)$ and the latency is $\bar{\tau}(x) = -\tau_m \log(1 - (V_{\text{th}} - E_L)/(R_m I(x)))$, the Fisher information from a single neuron is

$$\mathcal{I}_i^{\text{TTFS}}(x) = \frac{(\bar{\tau}_i'(x))^2}{\sigma_\tau^2(x)}, \tag{35}$$

[*exact: standard Fisher information for a Gaussian-noise model with mean* $\bar{\tau}(x)$ *and variance* $\sigma_\tau^2(x)$*, valid when the latency distribution is approximately Gaussian.*] where $\bar{\tau}_i'(x) = d\bar{\tau}_i/dx$.

The TTFS Fisher information does not scale with T. Once the first spike has been observed, additional time provides no further information about x, because x is already encoded in the latency. So, a TTFS code achieves its full Fisher information after a single spike.

6.4. The Speed-Precision Trade-Off

Combining (33) and (35) gives the central comparison: rate codes accumulate information linearly in time, while TTFS codes deliver their information in a single spike with timing precision σ_τ. The crossover time at which the two codes deliver equal information per neuron is

$$T_{\text{crossover}} = \frac{(\bar{\tau}'(x))^2 f(x)}{(f'(x))^2 \sigma_\tau^2(x)}, \tag{36}$$

[*exact: equating* (32) *and* (35) *per neuron and solving for* T*.*] where f, f' refer to the rate code's tuning curve and $\bar{\tau}'$ to the TTFS latency's slope.

For typical cortical parameters (firing rate around 10 Hz, tuning slope around 1 Hz/unit, TTFS latency precision around 1 ms), and assuming a latency-stimulus slope $\bar{\tau}'(x)$ on the order of 1 ms per unit stimulus, the crossover time is on the order of 10–100 ms. The exact value depends sensitively on $\bar{\tau}'(x)$ and is best computed for the specific encoder under consideration. For $T < T_{\text{crossover}}$, TTFS coding is more informative; for $T > T_{\text{crossover}}$, rate coding wins. This is the formal statement behind the intuitive claim that TTFS codes are good for fast decisions and rate codes are good for accurate ones [34].

6.5. Mutual Information and Channel Capacity

A complementary view of the information content of spike trains comes from mutual information,

$$I(X; \mathbf{S}) = H(\mathbf{S}) - H(\mathbf{S} \mid X) = H(X) - H(X \mid \mathbf{S}), \tag{37}$$

[*exact: definition.*] where $H(\cdot)$ denotes the entropy and the spike train $\mathbf{S}$ is treated as a random object whose distribution depends on the stimulus X. For a single neuron, the mutual information per unit time has been measured directly using the direct method of Strong et al. [32], yielding values on the order of bits per spike for some sensory neurons.

For an SNN, two upper bounds are useful. The first is the spike count entropy bound: a binary spike train of length $T/\Delta t$ can carry at most $T/(\Delta t \log 2)$ bits per second. The second is the achievable rate under a power constraint, given by the Shannon-Hartley-type formula adapted to point processes: a Poisson channel with mean rate $\bar{\lambda}$ and peak rate Λ has capacity at most $\bar{\lambda}$ nats per second when $\Lambda \to \infty$ [35]. [*exact: standard Poisson channel capacity result.*]

For SNN engineering, the practical implication of these bounds is that very-low-rate codes (a few spikes per neuron) are limited in the amount of information they can convey per unit time, but they are also energy-efficient: a single spike costs the same energy whether the firing rate is 1 Hz or 100 Hz. The trade-off between energy and information capacity is one of the main reasons why neuromorphic hardware can claim large efficiency advantages in some operating regimes and not in others [36,37].

 doi:10.20944/preprints202605.1130.v1

7. Computational Capacity

Three lines of work establish the computational capacity of SNNs: the Maass third-generation argument, the Stanojevic exact mapping from feedforward ReLU networks to TTFS-coded SNNs, and the Date-Schuman Turing-completeness construction. We present each in turn, then summarize the open questions in SNN complexity.

7.1. Expressivity: the Third-Generation Argument

Maass introduced the term "third generation" of neural network models to distinguish SNNs from threshold-perceptron networks (first generation) and networks with differentiable activations (second generation) [7]. The key claim is that SNNs are strictly more expressive than second-generation networks under temporal coding.

Theorem 6 (Expressivity of noisy spiking neurons [8])**.** *There exists a family of input-output mappings that can be implemented by a network of spiking neurons with temporal coding and noise but cannot be implemented by any finite network of sigmoidal (second-generation) neurons.*

[*exact: stated and proved in [8]; the proof constructs a specific separation problem that uses spike timing as a continuous parameter.*]

The result is a statement about the richness of the function class, not about practical trainability. Several caveats. First, the result assumes continuous time and noise of a specific structure; the constructive separation breaks down for synchronous discrete-time networks. Second, the result requires temporal coding; under rate coding, SNNs are equivalent in expressivity to standard recurrent neural networks. Third, the result does not imply that an SNN of given size can be efficiently trained to achieve the separation; trainability is a separate question.

For SNN engineering work, Theorem 6 is a useful background fact: it tells us that SNNs are not intrinsically less expressive than ANNs, even when the spike output looks discrete. The expressivity advantage is realized through temporal coding and is lost under crude rate coding.

7.2. Equivalence with ReLU Networks under TTFS Coding

A more recent and constructive result establishes a precise equivalence between feedforward ReLU networks and SNNs with TTFS coding.

Theorem 7 (Stanojevic et al. [34])**.** *For every feedforward ReLU network $\mathcal{N}$ with weights $W^{(\ell)}$, biases $b^{(\ell)}$, and ReLU activations, there exists a feedforward spiking neural network $\mathcal{N}'$ with TTFS coding such that, for every input $\mathbf{x}$, the spike times produced by $\mathcal{N}'$ encode the activations of $\mathcal{N}(\mathbf{x})$ exactly. The mapping is constructive: given $W^{(\ell)}$ and $b^{(\ell)}$, the weights and thresholds of $\mathcal{N}'$ can be written down explicitly.*

[*exact: stated and proved constructively in [34]; the construction uses TTFS coding with one spike per neuron and exponentially decaying synaptic kernels with carefully chosen time constants.*]

The construction works as follows. Each ReLU activation $a_i = \max(0, z_i)$ is encoded as a single spike at time $t_i = T_{\max} - \alpha a_i$ for an encoding constant α and a maximum time $T_{\max}$. A larger activation produces an earlier spike. Here, the weighted sum at the next layer is computed by the linear superposition of exponential PSPs, which become a continuous-time analog of the ReLU dot product when interpreted at the appropriate time. The threshold of the next-layer LIF neuron triggers at the exact time corresponding to the next ReLU activation. The construction is exact in the noiseless setting; in the presence of jitter, the spike times become noisy estimates of the activation values.

The equivalence has two practical implications. First, any deep learning task that can be solved by a feedforward ReLU network can in principle be solved by an SNN of the same architecture, with no loss of accuracy. Second, the resulting SNN uses one spike per neuron per inference, which is the lowest possible activity. The follow-up paper [38] demonstrates 0.3 spikes per neuron on practical tasks, a drastic reduction from the dozens of spikes per neuron in rate-coded SNNs.

 doi:10.20944/preprints202605.1130.v1

The result does not extend to networks with batch normalization, residual connections, or attention; the construction has been extended to specific architectural variants in subsequent work [39,40], but the general extension is open.

7.3. Computability: Turing Completeness

Beyond expressivity, one can ask about computability in the formal sense.

Theorem 8 (Date et al. [41]). *There exists an SNN architecture with biologically plausible neuron models (e.g., LIF) that is Turing complete: for every Turing machine M and input w, there is an SNN that simulates M on w.*

[*exact: stated and proved constructively in [41]; the construction uses recurrent connectivity and spike timing to encode the tape and head state of a universal Turing machine.*]

Turing completeness is a strong statement, but it is not without caveats. The construction in [41] requires an unbounded amount of time, an unbounded number of spikes, and arbitrarily fine timing precision. None of these are achievable in physical hardware. So, the result says that SNNs are not formally weaker than general-purpose computation; it does not say that SNNs are practically capable of solving arbitrary problems within a fixed resource budget.

The corresponding result for second-generation networks (ANN Turing completeness) was established earlier; it requires similar caveats. The contribution of [41] is to show that the SNN spike-timing mechanism does not introduce additional formal limitations.

7.4. Open Questions in SNN Complexity

While expressivity (Theorem 6), constructive equivalence with ReLU networks (Theorem 7), and Turing completeness (Theorem 8) are established, the computational complexity of SNNs in a finite-resource setting is largely open.

VC dimension:

The VC dimension of a feedforward SNN with n neurons and T time steps under TTFS coding is at most $\mathcal{O}(n^2 T \log(nT))$ [42]; tighter bounds and matching lower bounds for general spiking architectures are not known.

Sample complexity:

The number of training examples required to learn an SNN to a given accuracy is poorly understood. Generalization bounds based on Rademacher complexity or PAC-Bayes have been derived for specific SNN classes but the general picture is incomplete.

Optimization landscape:

The non-differentiability of spike generation makes the loss landscape of SNNs structurally different from that of ANNs. Surrogate gradient training appears to work surprisingly well in practice [22], but a theoretical explanation of why is missing.

Energy complexity:

A formal theory of the energy cost of SNN computation, in terms of spike counts and synaptic operations, would clarify when SNNs offer practical advantages over ANNs. The trade-off (36) is one ingredient; a complete theory is open.

Sparsity and lottery tickets:

Whether the lottery ticket hypothesis extends to SNNs, and whether sparse subnetworks can be identified in trained SNNs without retraining from scratch, is being actively investigated.

These open questions are the subject of current research. We list them not as a complete agenda but as pointers for the reader who wants to know what is known and what is not.

 doi:10.20944/preprints202605.1130.v1

8. Hazard-Based LIF Variants: A Status-Labeled Taxonomy

The LIF model is the simplest spiking neuron in common use, and it admits a wide range of extensions. A reader who looks at the literature will find dozens of variants that modify the threshold, the reset, the synaptic kernel, the leak, the noise, or the input encoding. Section 3 introduced the variation axes informally; here we present a systematic taxonomy of the variants that modify the spike-generation rule itself through a hazard function, together with the input-adaptive Liquid extension of that family.

The taxonomy is drawn from a public, patent-scoped reference implementation of the H-LIF family and its Liquid extension [11], which provides PyTorch implementations under a common interface, a custom CUDA kernel for TH-LIF with an analytic natural-gradient formulation, and a post-route FPGA validation database on the Digilent Arty A7-35T. We use this implementation as the source of the family list because it provides a single source of truth tied to the patent application; comparable lists could be assembled from the literature, but they would not share the implementation details and hardware measurements. The two families are summarized in Table 2, with the per-family status label explained below.

The present scope is intentionally restricted to the families covered by the patent application. A broader taxonomy of LIF variants is in preparation as a companion v2 of this article. The v2 will cover input-transform families that modify the current reaching the membrane through filter banks (multi-spectral), time-frequency atoms (wavelet, Gabor, Hilbert), or control laws (Kalman, MPC, PID); memory-kernel families that replace the exponential leak by fractional, Hawkes, or Lévy memory; information-theoretic objective families that derive the spike rule from free-energy, mutual-information, or surprise objectives; and domain-specific gating families such as cardiac variants that introduce phenomenological gating motivated by particular signal classes. The structural status labels for these additional families and their representative members are deferred to that companion article.

Table 2. Status-labeled taxonomy of the two patent-covered LIF variant families; the status label refers to each family's core extension, not implementation details. Broader variant axes are deferred to a companion v2.

Family	Count	Modification axis	Status	Representative models
H-LIF (Hazard)	13	threshold mechanism (deterministic, stochastic, history-dependent hazards)	mixed	TH-LIF, BH-LIF, DB-LIF, RT-LIF, MoE-LIF
Liquid	2	state-dependent hazard parameters	*heuristic*	L-BH-LIF, L-TH-LIF

8.1. H-LIF: Hazard-Modified Threshold Mechanism

The H-LIF family modifies the spike-generation rule. Baseline LIF uses a deterministic hard threshold (Section 3.6); H-LIF replaces the threshold with a hazard function $h(V, t, \mathcal{H}_t)$ that depends on the membrane potential, time, and possibly the spike history. The general form is

$$\Pr\{\text{spike in } [t, t+dt] \mid V(t), \mathcal{H}_t\} = h(V(t), t, \mathcal{H}_t)\, dt. \tag{38}$$

[*exact: definition of a generalized hazard function.*]

The spike decision is taken as an inhomogeneous Poisson rule: the probability of a spike in the interval $[t, t+\Delta t]$ is $1 - \exp(-h(t)\,\Delta t)$, compared against a uniform random sample. This unifies all members of the family under a single decision rule; what changes between members is how $h(t)$ is computed. Different choices of h produce different family members.

 doi:10.20944/preprints202605.1130.v1

- **Deterministic** (Det-LIF): $h = \delta(V - V_{\text{th}})$ is the hard-threshold limit; this recovers the baseline LIF. *exact*.
- **Stochastic** (Stoch-LIF) and **Boltzmann-Hazard** (BH-LIF): $h \propto \exp(\beta(V - V_{\text{th}}))$ is the escape-rate form (29). *exact* as a definition.
- **Tunneling-Hazard** (TH-LIF): h contains a quantum-tunneling-inspired term that allows spikes below V_{th} with probability that decays exponentially in the gap. *heuristic*: the form is suggested by analogy with quantum tunneling and is justified empirically; no rigorous derivation from a microscopic model.
- **Dual-barrier** (DB-LIF): two thresholds, with one above and one below, and different hazard rates. *heuristic*.
- **Refractory-tail** (RT-LIF): the hazard is reduced for a fixed time after the last spike. *exact* as a definition; equivalent to adding a refractory period on top of the baseline LIF dynamics.
- **MoE-LIF** (Mixture of Experts hazard): the hazard is a mixture of several hazard components, gated by the input. *exact* as a definition; the mixture structure is well-defined.
- **Attention** (Att-LIF) and **HyperNet-LIF**: the hazard depends on a learned modulating signal. *heuristic*.
- **Thermal-LIF**, **Spin-LIF**, **Piezo-LIF**, **Chaotic-Hazard** (CH-LIF): hazard forms inspired by physical analogies. *heuristic*.

The family-level status label in Table 2 is "mixed" because some H-LIF members (Det-LIF, Stoch-LIF, BH-LIF, RT-LIF, MoE-LIF) are exact extensions, while others (TH-LIF, DB-LIF, Att-LIF, CH-LIF, Thermal-LIF, Spin-LIF, Piezo-LIF, HyperNet-LIF) are heuristic. The TH-LIF variant is the most-used member in our applied work and is worth a separate paragraph.

TH-LIF (Tunneling-Hazard):

In the TH-LIF model, the hazard combines two independent pathways: a threshold-crossing term $h_{\text{thr}}(t) = \exp(\eta(V - V_{\text{th}} - \phi_0))$ that dominates above threshold, and a barrier-penetration term $h_{\text{bar}}(t) = \lambda_0 \exp(-\kappa(V_{\text{th}} - V))$ that allows spikes below threshold with probability decaying exponentially in the voltage gap, summed as $h_{\text{total}}(t) = h_{\text{thr}}(t) + h_{\text{bar}}(t)$. The four parameters $(\lambda_0, \kappa, \eta, \phi_0)$ are learnable, with the three rate-like parameters stored in log-space to enforce positivity. The motivation is by analogy with quantum tunneling through an energy barrier. The TH-LIF model is empirically observed to have better gradient flow under surrogate-gradient training because the smooth transition from no-spike to spike across the threshold reduces the size of the surrogate-gradient correction; the hazard formulation admits an analytic natural gradient, which the reference CUDA kernel exploits to remove the surrogate function entirely [11]. Hardware feasibility has been verified on a Digilent Arty A7-35T FPGA (Xilinx Artix-7 `xc7a35ticsg324-1L`) with a Q4.12 fixed-point implementation: the single-neuron design occupies 171 LUTs, 81 FFs, and 1 DSP48E1, runs at 50 MHz with six cycles per step, and consumes at most 240 pJ per neuron step under post-route SAIF-annotated power analysis with bit-exact agreement across five independent references (float, fixed-point Python, Icarus Verilog, Xilinx `xsim`, and the hardware itself) [11]. The TH-LIF formulation, its dual-pathway hazard architecture, and the Liquid extension described below are the subject of Turkish patent application 2026/007632, filed 14 May 2026 by İstinye University. It remains an instance of *heuristic*: the form is suggested by physical analogy, the empirical and hardware performance support it, but a rigorous derivation from a microscopic model is not available.

8.2. Liquid: State-Dependent Hazard Parameters

The Liquid family modifies the hazard parameters themselves to depend on the instantaneous neuron state. Baseline H-LIF members hold the hazard parameters $(\lambda_0, \kappa, \eta, \phi_0)$ fixed as learnable scalars; Liquid variants make each parameter a learned function of the membrane potential $V(t)$ and input current $I(t)$ at every time step, via $\theta_j(t) = \sigma_j(a_j V(t) + b_j I(t) + c_j)$ with positivity-preserving activations σ_j (softplus for the rate-like parameters, $\phi_{\max}$ tanh for the threshold offset) and learnable

 doi:10.20944/preprints202605.1130.v1

coefficients (a_j, b_j, c_j). Members include L-BH-LIF (Liquid Boltzmann-Hazard) and L-TH-LIF (Liquid Tunneling-Hazard); they are the state-adaptive analogs of BH-LIF and TH-LIF from the H-LIF family. When all coefficients (a_j, b_j, c_j) are zero, a Liquid variant reduces exactly to its parent H-LIF variant.

A characteristic feature of the Liquid construction is that the functions producing the hazard parameters are direct functions of the neuron's own state, not the outputs of a separate hypernetwork. This keeps the additional parameter count small (twelve scalars per neuron in the L-TH-LIF case) and allows end-to-end training of the coefficients alongside synaptic weights under standard surrogate-gradient backpropagation. The Liquid family is the subject of Turkish patent application 2026/007632 [11].

The family-level status is *heuristic*. Making the hazard parameters state-dependent is a useful empirical modification but it removes the time-invariant structure of the standard LIF. The lack of a time-invariant fixed point makes analytical results from Section 4 (echo state property, balanced regime) inapplicable in their standard forms.

8.3. Reading the Taxonomy

The taxonomy in Table 2 is restricted to the hazard-based H-LIF family and its Liquid extension, the two families covered by the patent application and the corresponding FPGA-validated reference implementation. Several other family axes also extend baseline LIF in well-defined ways and are deferred to the companion v2: input-transform families (multi-spectral, wavelet, control-theoretic), memory-kernel families (fractional, Hawkes, Lévy), information-theoretic objective families (free energy, mutual information, surprise), and domain-specific gating families including cardiac variants. Beyond these structured families, several individual models in the wider literature also fall outside the present scope, notably the Izhikevich quadratic model [43], the Mihalas-Niebur generalized LIF, and various Markov-chain-based models.

The status labels in the table reflect a deliberate point. The H-LIF family is "mixed": some members (Det-LIF, Stoch-LIF, BH-LIF, RT-LIF, MoE-LIF) are *exact* extensions, while others (TH-LIF, DB-LIF, Att-LIF, CH-LIF, Thermal-LIF, Spin-LIF, Piezo-LIF, HyperNet-LIF) are *heuristic*. The Liquid family is uniformly *heuristic*. A reader designing a new SNN can use the table to ask: which axis is being modified, and is the modification *exact*, *reduction*, *approximation*, or *heuristic*? This question structures the design choice and tells the reader what kind of justification will be expected at review.

The TH-LIF model is in the H-LIF family with a *heuristic* status. It is a useful illustration: a heuristic modification of the standard hazard can produce empirical improvements in surrogate gradient training and verifiable hardware feasibility on commodity FPGA, but the lack of a rigorous derivation from a microscopic model means the result must be supported by experiments and measurement rather than by a theorem. This is an honest description of the current state and is consistent with how most novel neuron models in the literature should be classified.

9. Outlook and Open Problems

A foundations paper closes naturally with a list of open problems. We are not exhaustive; we describe the problems that follow most directly from the material developed here.

Surrogate gradient theory:

The empirical robustness of surrogate gradient training [22] suggests that there is a structural reason why the method works, beyond the heuristic argument of replacing a Dirac delta with a smooth function. A rigorous analysis would clarify why arctan and fast-sigmoid surrogates produce more stable gradient flow than sigmoid surrogates, why the choice of β is forgiving, and how the surrogate gradient relates to the true gradient that EventProp [44] computes. A possible angle is to view surrogate gradient training as a smoothed version of an event-driven adjoint method, with the smoothing controlled by β. Whether this view leads to a useful theorem is open.

 doi:10.20944/preprints202605.1130.v1

Recurrent SNN dynamics:

Section 4 gave a mostly heuristic treatment of the echo state property for spiking reservoirs. A rigorous extension of the contraction-mapping criterion to threshold-and-reset dynamics, with explicit conditions on the synaptic time constants, would put recurrent SNN design on firmer theoretical ground. The balanced-network analysis [20,21] provides one rigorous framework, but it covers a specific scaling regime (sparse $1/\sqrt{K}$ scaling) that is not the regime of trained deep SNNs.

Generalization bounds for SNNs:

VC-dimension bounds for SNNs [42] are loose by polynomial factors and do not account for the temporal structure of the input. PAC-Bayes and Rademacher approaches that exploit the spike-train sparsity might give tighter bounds. The relationship between the spike-count budget at inference and the generalization gap is unexplored.

Energy complexity:

A formal complexity-theoretic treatment of the energy cost of SNN computation, in terms of spike counts and synaptic operations, would clarify when SNNs offer practical advantages. The Stanojevic 2024 result [38] is a striking empirical demonstration of low spike counts under TTFS coding; in order to obtain a matching complexity-theoretic statement, lower bounds on spike counts to achieve a given accuracy on a given problem class would be needed. This is open.

Compositional architectures:

The Stanojevic equivalence (Theorem 7) covers feedforward ReLU networks with simple connectivity. Extending the construction to attention layers, normalization, residual connections, and recurrent skip connections is open. Each architectural ingredient introduces its own subtleties: attention requires soft normalization, batch normalization shifts the operating point of LIF dynamics, residual connections create coupled time recurrences. A principled treatment that handles these ingredients in a unified framework is missing.

Point process models for trained SNNs:

The PP-GLM machinery (Section 5) is well-suited to fitting models to recorded spike trains. Whether trained SNNs, viewed through the same lens, exhibit identifiable history kernels and stimulus filters is largely unexplored. A possible application is interpretability: a trained SNN whose neurons are characterized by PP-GLM kernels can be examined for the temporal structure of the responses, potentially exposing the computation in a way that ANN feature visualization does not.

Status labels as a community convention:

The status-label system used in this article is a reading aid for one paper. Whether it is useful as a broader convention, in the way that papers in mathematics use "Theorem", "Lemma", and "Conjecture", remains to be seen. We suggest it as one possible answer to the notation problem in SNN literature; alternative conventions are also possible.

These problems are open in different senses. Some (surrogate gradient theory, recurrent dynamics) are mathematical questions where progress requires new techniques. Others (energy complexity, generalization bounds) require empirical work before a theorem is even formulable. The surface area of the field is large enough that no single paper will close them; the pointer here is meant to orient a reader who finishes this article and wonders what to work on next.

10. Conclusions

This article has presented the mathematical foundations of spiking neural networks in a unified formalism, with status labels attached to every major equation. The substantive content was the reduction chain from Hodgkin-Huxley to LIF (Section 3), the network theory of recurrent SNNs (Section 4), the point process formulation with the time-rescaling theorem and the PP-GLM (Section 5),

 doi:10.20944/preprints202605.1130.v1

the information-theoretic analysis of rate and TTFS codes (Section 6), the three lines of capacity results (Section 7), and a status-labeled taxonomy of the hazard-based H-LIF family and its Liquid extension (Section 8), with the broader taxonomy of LIF variant axes deferred to a companion v2.

The status labels make a methodological point. The same equation can describe an exact identity, a parameter limit, a working approximation, or a heuristic that is empirically supported but not derived. Distinguishing these cases is part of reading SNN literature carefully. The first installment of the series [1] uses the same labels at a more elementary level; the two articles share notation and a reader can move between them with the cross-references provided.

We have left out of this article the topics that would have stretched its focus: hardware implementations, software frameworks, application benchmarks, and the open problems of the field beyond a focused list in Section 9. The selection is deliberate. A foundations paper should be the place where a researcher comes to understand what is known and how to read what is being claimed; it is not the place to survey everything that is currently being worked on. We hope this article serves the first purpose.

Acknowledgments: This article is the second installment in a series on spiking neural networks. The first installment [1] addresses the practical side of training SNNs. Aspects of the H-LIF family hazard formulations and the Liquid extension described in Section 8 are the subject of Turkish patent application 2026/007632, filed 14 May 2026, applicant: İstinye University. The patent-scoped public reference implementation [11], which provided the source material for the taxonomy and includes a custom CUDA kernel for TH-LIF with an analytic natural-gradient formulation together with a post-route FPGA validation database on the Arty A7-35T (171 LUTs, $\leq$ 240 pJ per neuron step, bit-exact across five independent references), is available under Apache 2.0 with an accompanying patent notice; commercial use of the patented inventions should be coordinated through the İstinye University Technology Transfer Office. We thank the SNN open-source community, in particular the developers of snnTorch, SpikingJelly, Norse, and Tonic, whose work makes engineering progress in this area possible.

References

1. İsmail Can Dikmen. Spiking Neural Networks: A Tutorial on Models, Coding, and Training. *Preprints* **2026**. https://doi.org/10.20944/preprints202605.0827.v1.
2. Lapicque, L. Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation. *Journal de Physiologie et de Pathologie Générale* **1907**, *9*, 620–635. No DOI available for this 1907 publication.
3. Hodgkin, A.L.; Huxley, A.F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology* **1952**, *117*, 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764.
4. Cox, D.R.; Isham, V. *Point Processes*; Chapman and Hall, 1980. https://doi.org/10.1201/9780203743034.
5. Daley, D.J.; Vere-Jones, D. *An Introduction to the Theory of Point Processes. Volume I: Elementary Theory and Methods*, 2 ed.; Springer, 2003. https://doi.org/10.1007/b97277.
6. Snyder, D.L.; Miller, M.I. *Random Point Processes in Time and Space*, 2 ed.; Springer, 1991. https://doi.org/10.1007/978-1-4612-3166-0.
7. Maass, W. Networks of spiking neurons: The third generation of neural network models. *Neural Networks* **1997**, *10*, 1659–1671. https://doi.org/10.1016/S0893-6080(97)00011-7.
8. Maass, W. Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons. In Proceedings of the Advances in Neural Information Processing Systems 9 (NIPS 1996); Mozer, M.C.; Jordan, M.I.; Petsche, T., Eds. MIT Press, 1997, pp. 211–217. No DOI available for pre-2000 NeurIPS proceedings.
9. Neftci, E.O.; Mostafa, H.; Zenke, F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine* **2019**, *36*, 51–63. https://doi.org/10.1109/MSP.2019.2931595.
10. Zenke, F.; Ganguli, S. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation* **2018**, *30*, 1514–1541. https://doi.org/10.1162/neco_a_01086.
11. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* **2018**, *38*, 82–99. https://doi.org/10.1109/MM.2018.112130359.

 doi:10.20944/preprints202605.1130.v1

12. Brette, R.; Gerstner, W. Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity. *Journal of Neurophysiology* **2005**, *94*, 3637–3642. https://doi.org/10.1152/jn.00686.2005.
13. Gerstner, W.; Kistler, W.M.; Naud, R.; Paninski, L. *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*; Cambridge University Press, 2014. https://doi.org/10.1017/CBO9781107447615.
14. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press, 2002. https://doi.org/10.1017/CBO9780511815706.
15. Maass, W.; Natschläger, T.; Markram, H. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation* **2002**, *14*, 2531–2560. https://doi.org/10.1162/089976602760407955.
16. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks. *GMD Report* **2001**, *148*, 1–47. German National Research Center for Information Technology.
17. Bertschinger, N.; Natschläger, T. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation* **2004**, *16*, 1413–1436. https://doi.org/10.1162/089976604323057443.
18. Boedecker, J.; Obst, O.; Lizier, J.T.; Mayer, N.M.; Asada, M. Information processing in echo state networks at the edge of chaos. *Theory in Biosciences* **2012**, *131*, 205–213. https://doi.org/10.1007/s12064-011-0146-8.
19. Lukoševičius, M.; Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review* **2009**, *3*, 127–149. https://doi.org/10.1016/j.cosrev.2009.03.005.
20. van Vreeswijk, C.; Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **1996**, *274*, 1724–1726. https://doi.org/10.1126/science.274.5293.1724.
21. Brunel, N. Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *Journal of Computational Neuroscience* **2000**, *8*, 183–208. https://doi.org/10.1023/A:1008925309027.
22. Zenke, F.; Vogels, T.P. The Remarkable Robustness of Surrogate Gradient Learning for Instilling Complex Function in Spiking Neural Networks. *Neural Computation* **2021**, *33*, 899–925. https://doi.org/10.1162/neco_a_01367.
23. Eshraghian, J.K.; Ward, M.; Neftci, E.O.; Wang, X.; Lenz, G.; Dwivedi, G.; Bennamoun, M.; Jeong, D.S.; Lu, W.D. Training Spiking Neural Networks Using Lessons From Deep Learning. *Proceedings of the IEEE* **2023**, *111*, 1016–1054. https://doi.org/10.1109/JPROC.2023.3308088.
24. Brown, E.N.; Barbieri, R.; Ventura, V.; Kass, R.E.; Frank, L.M. The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation* **2002**, *14*, 325–346. https://doi.org/10.1162/089976602252741149.
25. Ogata, Y. Statistical models for earthquake occurrences and residual analysis for point processes. *Journal of the American Statistical Association* **1988**, *83*, 9–27. https://doi.org/10.1080/01621459.1988.10478560.
26. Paninski, L. Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems* **2004**, *15*, 243–262. https://doi.org/10.1088/0954-898X/15/4/002.
27. Hawkes, A.G. Spectra of some self-exciting and mutually exciting point processes. *Biometrika* **1971**, *58*, 83–90. https://doi.org/10.1093/biomet/58.1.83.
28. Brémaud, P.; Massoulié, L. Stability of nonlinear Hawkes processes. *The Annals of Probability* **1996**, *24*, 1563–1588. https://doi.org/10.1214/aop/1065725193.
29. Truccolo, W.; Eden, U.T.; Fellows, M.R.; Donoghue, J.P.; Brown, E.N. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology* **2005**, *93*, 1074–1089. https://doi.org/10.1152/jn.00697.2004.
30. Pillow, J.W.; Shlens, J.; Paninski, L.; Sher, A.; Litke, A.M.; Chichilnisky, E.J.; Simoncelli, E.P. Spatio-temporal correlations and visual signaling in a complete neuronal population. *Nature* **2008**, *454*, 995–999. https://doi.org/10.1038/nature07140.
31. Pillow, J.W.; Paninski, L.; Uzzell, V.J.; Simoncelli, E.P.; Chichilnisky, E.J. Prediction and decoding of retinal ganglion cell responses with a probabilistic spiking model. *Journal of Neuroscience* **2005**, *25*, 11003–11013. https://doi.org/10.1523/JNEUROSCI.3305-05.2005.
32. Strong, S.P.; Koberle, R.; de Ruyter van Steveninck, R.R.; Bialek, W. Entropy and information in neural spike trains. *Physical Review Letters* **1998**, *80*, 197–200. https://doi.org/10.1103/PhysRevLett.80.197.
33. Rieke, F.; Warland, D.; de Ruyter van Steveninck, R.; Bialek, W. *Spikes: Exploring the Neural Code*; MIT Press, 1997.
34. Stanojevic, A.; Woźniak, S.; Bellec, G.; Cherubini, G.; Pantazi, A.; Gerstner, W. An exact mapping from ReLU networks to spiking neural networks. *Neural Networks* **2023**, *168*, 74–88. https://doi.org/10.1016/j.neunet.2023.09.011.

 doi:10.20944/preprints202605.1130.v1

35. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*, 2 ed.; Wiley-Interscience, 2006. https://doi.org/10.1002/047174882X.
36. Zenke, F.; Neftci, E.O. Brain-Inspired Learning on Neuromorphic Substrates. *Proceedings of the IEEE* **2021**, *109*, 935–950. https://doi.org/10.1109/JPROC.2020.3045625.
37. Schuman, C.D.; Kulkarni, S.R.; Parsa, M.; Mitchell, J.P.; Date, P.; Kay, B. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science* **2022**, *2*, 10–19. https://doi.org/10.1038/s43588-021-00184-y.
38. Stanojevic, A.; Woźniak, S.; Bellec, G.; Cherubini, G.; Pantazi, A.; Gerstner, W. High-performance deep spiking neural networks with 0.3 spikes per neuron. *Nature Communications* **2024**, *15*, 6793. https://doi.org/10.1038/s41467-024-51110-5.
39. Goltz, J.; Kriener, L.; Baumbach, A.; Billaudelle, S.; Breitwieser, O.; Cramer, B.; Dold, D.; Kungl, A.F.; Senn, W.; Schemmel, J.; et al. Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nature Machine Intelligence* **2021**, *3*, 823–835. https://doi.org/10.1038/s42256-021-00388-x.
40. Comsá, I.M.; Fischbacher, T.; Potempa, K.; Gesmundo, A.; Versari, L.; Alakuijala, J. Temporal coding in spiking neural networks with alpha synaptic function. *ICASSP 2020 - IEEE International Conference on Acoustics, Speech and Signal Processing* **2020**, pp. 8529–8533. https://doi.org/10.1109/ICASSP40776.2020.9053856.
41. Date, P.; Potok, T.E.; Schuman, C.D.; Kay, B. Neuromorphic Computing is Turing-Complete. In Proceedings of the Proceedings of the International Conference on Neuromorphic Systems (ICONS 2022), New York, NY, USA, 2022. https://doi.org/10.1145/3546790.3546806.
42. Maass, W.; Schmitt, M. On the complexity of learning for spiking neurons with temporal coding. *Information and Computation* **1999**, *153*, 26–46. https://doi.org/10.1006/inco.1999.2806.
43. Izhikevich, E.M. Which model to use for cortical spiking neurons? *IEEE Transactions on Neural Networks* **2004**, *15*, 1063–1070. https://doi.org/10.1109/TNN.2004.832719.
44. Wunderlich, T.C.; Pehle, C. Event-based backpropagation can compute exact gradients for spiking neural networks. *Scientific Reports* **2021**, *11*, 12829. https://doi.org/10.1038/s41598-021-91786-z.