Neuroinformatics (2026) 24:5
https://doi.org/10.1007/s12021-025-09754-1

RESEARCH

Impact of Neuron Models on Spiking Neural Network Performance: A Complexity-based Classification Approach

Zofia Rudnicka[1] · **Janusz Szczepanski**[1] · **Agnieszka Pregowska**[1]

Received: 2 September 2025 / Accepted: 4 November 2025

Abstract

This study addresses the important question of how neuron model choice and learning rules shape the classification performance of Spiking Neural Networks (SNNs) in bio-signal processing. By systematically contrasting Leaky Integrate-and-Fire, metaneurons, and probabilistic Levy-Baxter (LB) neurons across spike-timing dependent plasticity, tempotron, and reward-modulated learning, we identify model-rule combinations best suited for capturing the temporal richness of neural data. A novel contribution is the integration of a complexity-driven evaluation into the SNN pipeline. Using Lempel-Ziv Complexity (LZC), an entropy-related measure of spike-train regularity, we provide a consistent and interpretable benchmark of classification outcomes across architectures. To probe neural dynamics under controlled conditions, we employed synthetic datasets with varying temporal dependencies and stochasticity, including Markov and Poisson processes established models of neuronal spike-trains. Moreover, we validated the observed trends on real data by testing the same architectures on an MNIST dataset. Performance trends reveal strong dependence on the interaction between neuron model, learning rule, and network size. The LZC based evaluation highlights configurations resilient to weak or noisy signals. The LB-tempotron combination proved most effective for tasks with complex temporal patterns, leveraging adaptive neuronal dynamics and precise spike-timing exploitation. LIF-based architectures with Bio-inspired Active Learning delivered solid accuracy at lower computational cost, while hybrid models offered a versatile middle ground when paired with appropriate learning algorithms. This work delivers the first systematic mapping of neuron model learning rule synergies in SNNs and introduces complexity-based evaluation framework that sets a robust benchmark for biosignal classification. Beyond benchmarking, our results provide actionable guidelines for building next-generation SNNs capable of handling the variability and complexity of real neural data.

Keywords Spiking neural networks · Neuron models · Learning algorithms · Lempel-Ziv complexity

Introduction

Neural communication is a highly complicated and dynamic process through which neurons convey information within biological systems. The fundamental mechanism of this communication involves the propagation of electrical signals, known as action potentials (Gerstner et al., 2014). Artificial Neural Networks (ANNs) are designed to emulate aspects of this biological communication process through a system of interconnected computational nodes, or neurons. These nodes process information via mathematical operations, and their interconnections are modifiable based on learning algorithms, mirroring the adaptive mechanisms observed in biological neural networks. The connections in neural networks are adjusted based on feedback, allowing them to "learn" from data and optimize their performance,

Zofia Rudnicka and Agnieszka Pregowska contributed equally to this work.

✉ Agnieszka Pregowska
aprego@ippt.pan.pl

Zofia Rudnicka
zrudnick@ippt.pan.pl

Janusz Szczepanski
jszczepa@ippt.pan.pl

[1] Institute of Fundamental Technological Research, Polish Academy of Sciences, Pawinskiego 5B, Warsaw 02-106, Poland

Published online: 08 January 2026

similar to the synaptic modifications in the brain. ANNs utilize simplified mathematical models that simulate the underlying processes of neural communication. These models operate within a hidden layer, where they are connected to the output layer of the network. The computations within these nodes are based on mathematical operations, and their output is combined with the weights of the connections between nodes, which are adjusted during training, mainly with algorithms such as backpropagation (Wu et al., 2018). In this framework, the training process involves iteratively adjusting the weights of connections based on the discrepancy between the predicted outputs and the actual outputs. This adjustment allows the network to "learn" and improve its performance over time. Although artificial neurons are designed to mimic certain aspects of biological neurons, they do so in a much more abstract and simplified manner. The complexity of biological neurons, influenced by a wide range of internal and external factors, is reduced to a basic computational model that focuses on input-output relationships.

While both biological and artificial neurons process information, the mechanisms and complexity differ significantly (Seguin et al., 2023). Biological neurons operate in highly dynamic environments, with their activity influenced by a multitude of biochemical processes and external stimuli. In contrast, artificial neurons, governed by specific architectures and learning algorithms, are simplified representations of this complexity. This difference highlights the inherent contrast between the adaptive, biochemical complexity of biological systems and the structured, algorithmic framework of computational models. Thus, the limitations of classical Artificial Intelligence, particularly in models based on perceptrons and traditional ANNs naturally forced the exploration of alternative neural network architectures that can improve computational efficiency. One of the most promising candidates for overcoming these limitations seems to be Spiking Neural Networks (SNNs), which are considered to be a more energy-efficient option for complex calculations (Datta et al., 2022). The key distinction between SNNs and conventional ANNs lies in their output dynamics. Unlike ANNs, SNNs utilize a spiking mechanism, where information is transmitted as discrete temporal spikes rather than as continuous signals. This dynamic spike-based signaling more closely emulates the way information flows through biological synapses, allowing SNNs to represent features in spatiotemporal data more effectively. Consequently, SNNs exhibit a promising ability to perform computations in a manner that approaches the temporal processing capabilities of the human brain, enabling richer and more efficient representations of time-dependent data (Sun et al., 2025).

When studying the efficiency of information transmission in SNN, the selection of both neuron and network architecture models is crucial. A model that accurately replicates the spike response of a neuron to any input current is fundamental for both constructing brain simulators and understanding the computational mechanisms of neural activity (Izhikevich, 2003; Shirsavar et al., 2023). Several approaches to neuron modeling are being developed (Gerstner et al., 2014), with two primary lines of development being most prominent. The first approach involves detailed biophysical modeling, such as Hodgkin and Huxley-like models, which describe the dynamics of ion channels within the spatially structured tree-like morphology of neurons. The second approach includes the integrate-and-fire (LIF) models (Dutta et al., 2017), which treat neuronal electrical activity as a threshold-based process. As network building blocks (single-neuron models), first, we will assume perceptorns, than recently used in SNNs, Leaky Integrate-and-Fire neuron model, metaneurons (Weng et al., 2020) and probabilistic Levy-Baxter neuron model (Levy & Baxter, 2002; Paprocki et al., 2020), which provide results consistent with physiological observed values. In this paper, we analyze how the performance of the SNN is influenced by the neuron model used to build it. We introduce a novel hybrid framework that integrates the temporal precision and biological plausibility of SNNs with the Lempel-Ziv complexity (LZC) measure (Ziv & Lempel, 1976) to improve the classification of spatiotemporal neural data. By quantifying the structural complexity of spike patterns, the proposed method offers interpretable and noise-robust classification, particularly effective for data exhibiting variable temporal dynamics, such as Poisson-distributed signals.

Main Contribution

The main contributions of the paper include a comprehensive analysis of how different neuron models: LIF, metaneurons, and probabilistic Levy-Baxter, affect SNN classification performance across varying network sizes and tasks, as well as a systematic evaluation of multiple learning algorithms, including unsupervised (STDP, SDSP), supervised (tempotron, backpropagation), and hybrid reward-modulated approaches, to quantify their interaction with neuron model choice. The study further demonstrates the effectiveness of bio-inspired neuron models in bio-signal classification, achieving high accuracy, sensitivity, and specificity on synthetic datasets modeled with Poisson and Markov processes. In addition, it introduces a complexity-based evaluation by integrating Shannon's information theory with SNN output, using LZC to capture subtle temporal structures in spike trains and improve the detection of weak or noisy signals. The results identify performance trends showing how optimal neuron model-learning rule combinations depend

on signal characteristics and network size, with tempotron and reward-based learning showing notable advantages in specific regimes. Finally, the paper highlights the potential of SNNs as a biologically plausible and computationally efficient framework for processing complex spatio-temporal data, particularly biosignals.

Basics Notation

In Table 1 the notation used is presented.

Models of Neurons

The perceptron operates in an *n*-dimensional real vector space $\mathbb{R}^n$. The input vector is $\mathrm{x} = [x_1, x_2, \ldots, x_n]$, and the weight vector is $\mathrm{w} = [w_1, w_2, \ldots, w_n] \in \mathbb{R}^n$. The perceptron computes a weighted sum of inputs with an added bias *b*, as given by:

$$z = \sum_{i=1}^{n} w_i x_i + b \tag{1}$$

where $b \in \mathbb{R}$ is the bias term, and the output is determined by a threshold function *f*(*z*):

$$f(z) = \begin{cases} 1 & \text{if } z \geq \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where θ is the activation threshold (Parlos et al., 1994).

The Leaky Integrate-and-Fire model describes the membrane potential dynamics of a neuron (Dutta et al., 2017). The membrane potential $U(t)$ evolves over time according to the equation:

$$\tau_m \frac{dU}{dt} = -U(t) + R_m I(t) \tag{3}$$

where τ_m is the membrane time constant, R_m is the membrane resistance, and $I(t)$ is the input current at time *t*. When the membrane potential exceeds a threshold U_{th}, a spike occurs, and the potential is reset to a lower value U_r. This process models the gradual accumulation and leakage of membrane potential.

Table 1 Basic notation used in the study

Description	Notation
input vector	$\mathrm{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$
weight vector	$\mathrm{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$
bias	b
threshold	θ
weighted sum	$z = \mathrm{w}^\top \mathrm{x} + b \in \mathbb{R}$
activation function	$f(z),\ f : \mathbb{R} \to \mathbb{R}$ or $f : \mathbb{R} \to \{0, 1\}$

The metaneuron is a higher-level computational unit that abstracts the activity of multiple neurons or neural processes (Cheng et al., 2023). It introduces a modular approach to neural network design, facilitating large-scale modeling by representing groups of neurons or collective behaviors. Like the perceptron, it computes a weighted sum of inputs, but with greater flexibility in activation functions, supporting binary step, sigmoid, or ReLU. Unlike the perceptron, the metaneuron can model dynamic neuron behaviors, including spiking dynamics, by processing time-varying inputs and evolving internal states, akin to LIF model. This generalization allows it to capture complex neural phenomena such as oscillations and synchrony, making it highly adaptable for large-scale networks.

The Levy-Baxter (LB) model incorporates probabilistic dynamics to capture synaptic transmission variability (Levy & Baxter, 2002; Paprocki et al., 2020). The inputs to the neuron are represented by the vector $\mathrm{x} = [x_1, x_2, \ldots, x_n]$, with each x_i modeled as a binary stochastic process. The transformation of each input is governed by a Bernoulli random variable ϕ_i with success probability *s*, and the amplitude is scaled by a random variable Q_i uniformly distributed over [0, 1]. The transformed input is given by:

$$\mathrm{y} = [\phi_1 Q_1 x_1, \phi_2 Q_2 x_2, \ldots, \phi_n Q_n x_n] \tag{4}$$

The total excitation σ is the sum of the transformed inputs:

$$\sigma = \sum_{i=1}^{n} \phi_i Q_i x_i \tag{5}$$

The output of the neuron is determined by a threshold function $f(\sigma)$:

$$z = \begin{cases} 1 & \text{if } \sigma \geq 0 \\ 0 & \text{if } \sigma < 0 \end{cases} \tag{6}$$

where $z = 1$ indicates the neuron has fired, and $z = 0$ indicates no spike. This probabilistic approach models a synaptic variability, with x_i as inputs, ϕ_i as quantal release probabilities, and Q_i as the scaling factor representing amplitude fluctuations.

Each neuron model described here represents a different abstraction of neural behavior. The perceptron is a simple threshold-based model used for binary classification tasks, operating deterministically. The spiking phenomenon is not included. The LIF model incorporates temporal dynamics, modeling the gradual integration of inputs and natural

leakage of membrane potential. It introduces time-dependent behavior and spiking. The metaneuron abstracts the collective activity of multiple neurons into a higher-level computational unit, enabling modular and hierarchical network design. The spiking phenomenon is optional. The LB model introduces stochastic and quantal variability in synaptic transmission, providing a probabilistic framework for understanding neural responses. While stochastic provides more realistic noise modeling.

The LB neuron introduces probabilistic variability into synaptic transmission, modeling the inherently noisy and quantal nature of biological synapses. This stochastic mechanism allows the neuron to represent uncertainty and adapt its response to fluctuating input conditions, thereby capturing a key physiological property of biological neural systems often overlooked in deterministic models. The probabilistic firing behavior enhances the diversity and richness of temporal activity patterns, increasing the model's coding capacity through variability in spike timing and amplitude. Such variability supports distributed and redundant representations, improving robustness to input perturbations and synaptic noise.

Moreover, stochasticity contributes to energy-efficient signaling: rather than maintaining continuous high-frequency activity, the LB neuron exhibits adaptive firing rates governed by probabilistic thresholds, reducing metabolic load while preserving informational throughput. Relative to the LIF model, which integrates inputs in a fixed, deterministic manner, the LB neuron offers a more flexible and biologically plausible framework that links noise to computation rather than treating it as an unwanted artifact. When compared to metaneurons, the LB model emphasizes fine-grained, probabilistic mechanisms at the single-neuron level, whereas metaneurons abstract this variability into ensemble-level stability and modular computation. Together, these perspectives highlight the role of stochastic dynamics as both a source of biological realism and a functional advantage, enabling adaptive coding, efficient energy utilization, and resilience in complex neural architectures. In contrast, metaneurons aggregate multiple neural responses into higher-level computational units, effectively smoothing out micro-level noise. This suggests a potential mechanism for emergent stability and energy efficiency in large-scale networks. Incorporating stochastic or hierarchical features into neuron models not only affects computational performance but also aligns with physiologically relevant principles of efficiency, variability, and adaptive coding.

Spiking Neural Network Architecture

Spiking Neural Networks process information by considering spike signals, making them particularly promising for handling complex tasks (Naderi et al., 2025). These networks excel at encoding intricate spatiotemporal information through spike patterns. The design of SNNs is often based on models like LIF neuron model, which is a simplified representation of how biological neurons process information. In these models, the arrival of a spike at a presynaptic neuron triggers an input current $I(t)$ that influences the membrane potential of the postsynaptic neuron. For simplicity, we can express the input current as a convolution of the spike signal $S_j(t)$ rom a presynaptic neuron with an exponential decay function, representing the temporal filtering of the spike signal

$$I(t) = \int_0^\infty S_j(s-t) exp(\frac{-s}{\tau_s}) ds, \tag{7}$$

where $S_j(s-t)$ represents the spike train from the j-th presynaptic neuron, and τ_s is a synaptic time constant, dictating the decay of the signal over time. This equation models the temporal dynamics of the input signal, which integrates over time, decaying with rate τ_s.

The neural network consists of three layers of neurons: input, hidden, and output, each containing n neurons, as shown in Fig. 1. We considered neural networks comprising

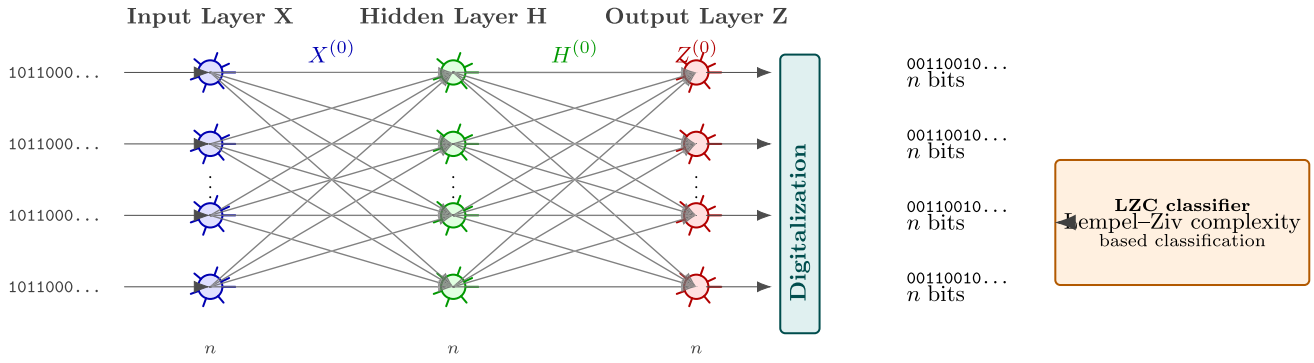


Fig. 1 The basics scheme of the conducted classification task

16, 32, 64, 128, 256, 512, and 1024 neurons per layer. The best of the results obtained are presented in Tables 1, 2, 3, 4 and 5. Sequences of binary values (strings of zeros and ones), each 1024 bits long, are fed into the network. Then, n-bit-long sequences of action potentials (spike trains) are generated by the network and subsequently converted back into sequences of zeros and ones. To store and manipulate spike times efficiently at scale, we adopt a bit-encoding data structure specialized for event-based signals, which substantially reduces memory footprint and improves runtime for large spike matrices (Ljungquist et al., 2018).

Shannon's Information Theory establishes a mathematical framework for quantifying and analyzing the transmission of information within communication systems (Shannon, 1948). Thus, entropy rate estimators provide a rigorous mathematical approach to approximating information transmission rates, offering an alternative to traditional firing rate analysis. Notably, Lempel-Ziv complexity, as defined by Ziv and Lempel (1976), has been successfully employed as an effective estimator. The Lempel-Ziv complexity is a widely used metric for estimating entropy, and consequently the informational content of sequences. Given a sequence $\mathrm{x}_n^1 := [x_1, x_2, \ldots, x_n]$, where each x_i belongs to a finite source alphabet (e.g., $x_i \in \{0, 1\}$), the complexity $C_\alpha(\mathrm{x}_n^1)$ counts the number of distinct blocks (or patterns) in the sequence. A new block is defined when a substring starting from the current position has not appeared before. The normalized complexity, $c_\alpha(\mathrm{x}_n^1)$, which measures the rate at which new patterns are generated, is defined as:

$$c_\alpha(\mathrm{x}_n^1) = \frac{C_\alpha(\mathrm{x}_n^1)}{n} \log_\alpha n, \tag{8}$$

where $\alpha = 2$ for binary sequences. Asymptotically, $c_2(\mathrm{x}_n^1) \to 1$ for random sequences and $c_2(\mathrm{x}_n^1) \to 0$ for deterministic sequences. For random sequences, the normalized complexity tends to 1, and for deterministic sequences, it approaches 0. The LZ complexity serves as an effective estimator of entropy for ergodic stochastic processes (Ziv & Lempel, 1976; Arnold et al., 2013; Pregowska et al., 2016, 2019). In neural networks, Lempel-Ziv complexity is applied to classify output sequences based on their informational content and unpredictability. The output sequences are classified using the Lempel-Ziv 1976 complexity-based classifier. Various neuron models were used to construct the SNN, including the perceptron, LIF, metaneuron, and LB neuron models. Moreover, different types of learning algorithms, including unsupervised methods (e.g., spike-timing dependent plasticity STDP Zhao et al., 2025), spike-driven synaptic plasticity SDSP He et al., 2025), supervised methods (e.g., tempotron Gutig & Sompolinsky, 2006; Yu et al., 2014, backpropagation Wuet al., 2018), and hybrid methods (e.g., reward-based learning with active learning Zhan et al., 2023), have been widely investigated to optimize network performance.

To ensure the reproducibility and transparency of our results, simulations were carried out within a controlled environment, where all parameters and procedures were explicitly defined. The architecture of the networks, the choice of neuron models (perceptron, LIF, metaneuron, LB neuron) and the learning rules were consistently specified across experiments. In order to minimize variability, fixed random seeds were used during initialization, ensuring that repeated runs yielded comparable results (Beyeler et al., 2014; Richmond et al., 2014). The input data consisted of binary sequences of length 1024, processed by networks of different sizes ranging from 16 to 1024 neurons per

Table 2 Representative configurations: best accuracy and best time-accuracy trade-off per input process

Input	Model + Learining Rule	Neurons/layer	Epochs	Accuracy (%)	Time
Poisson (best trade-off)	Perceptron + Tempotron	64	10	100.00	8.5 s
Bernoulli (fast, high acc.)	Meta + BAL	32	10	99.50	6.7 s
Bernoulli (max acc.)	LB + BAL	32	10	100.00	22.2 s
Markov (max acc.)	Perceptron + Tempotron	128	10	100.00	53.6 s

Full grids in Tables 5-9 in Appendix

Table 3 Per-run accuracy, normalized LZC c_2 (bin=4), and spike-rate (Conv only)

arch	dataset	acc	c_2 (bin=4)
Conv-SNN	Poisson	0.917	1.254
Conv-SNN	Markov	1.000	1.082
Conv-SNN	Bernoulli	1.000	0.936
Reservoir-SNN	Poisson	0.667	0.208
Reservoir-SNN	Markov	0.938	1.010
Reservoir-SNN	Bernoulli	1.000	0.431

Table 4 Accuracy and fractional LZC (bin=4) aggregated over seeds (mean ± std)

arch	dataset	acc	c_2 (mean ± std)
Conv-SNN	Bernoulli	1.000 ± 0.000	0.156 ± 0.000
Conv-SNN	Markov	1.000 ± 0.000	0.168 ± 0.000
Conv-SNN	Poisson	0.950 ± 0.000	0.200 ± 0.000
Reservoir-SNN	Bernoulli	1.000 ± 0.000	0.076 ± 0.000
Reservoir-SNN	Markov	0.938 ± 0.000	0.173 ± 0.000
Reservoir-SNN	Poisson	0.750 ± 0.000	0.036 ± 0.000

layer. Identical pre-processing and spike-encoding procedures were applied to all datasets, making the experimental pipeline uniform and comparable. For every network configuration, training and evaluation followed standardized protocols, including the number of epochs, the selection of learning algorithms (e.g., STDP, SDSP, backpropagation), and the relevant hyperparameters. Furthermore, simulation scripts, together with parameter settings, were designed in a portable manner, so that the experiments can be reproduced on different computational platforms without loss of generality.

Input Datasets

Neuronal action potentials and spike sequences are often modeled as point processes, providing a statistical framework for analyzing discrete events over time (Daley et al., 2003; Wojcik et al., 2009). Poisson point processes are particularly useful for representing spike trains with minimal temporal dependencies, effectively capturing the stochastic nature of neuronal firing (Rieke et al., 1997). Their Markov properties make them suitable for modeling short-term memoryless behavior in neural dynamics (Papoulis et al., 2002).

These insights emphasize the need for probabilistic and temporally structured input data to evaluate the capacity of computational models to represent neural signals accurately. To this study uses three types of synthetic binary sequences as input datasets: Bernoulli sequences, first-order Markov sequences, and Poisson-based spike trains. Each of these allows for controlled manipulation of randomness, temporal dependence, and rate-based variability, i.e. factors critical for evaluating the interplay between neural dynamics and classification mechanisms based on complexity.

Bernoulli sequences are generated from independent Bernoulli processes, where each binary element B_i is drawn independently with probability p. For our experiments, we generate two sets of binary sequences B_1 and B_2, each of length 1024, using different values of p. The independence of events allows us to assess the system's performance under random binary outcomes with varying probabilities.

Markov sequences are generated from a first-order Markov process, where the probability of each element M_i depends on the previous state M_{i-1}. For our experiments, two sets of binary sequences $M_1 = \{m_{1i}\}, i = 1, 2, \ldots, 1000$ and $M_2 = \{m_{2i}\}, i = 1, 2, \ldots, 1000$ are generated with transition probabilities defining the dependency structure. We explore different transition probability configurations to evaluate the impact of state dependencies on classification accuracy.

Poisson sequences are generated from a Poisson process, where events occur independently at a constant average rate λ. For each Poisson process, spike trains are generated with rate parameters λ_x and λ_y. These sequences are tested under various rate configurations to observe the effect of spike timing variability on system behavior and classification performance.

To explore neural dynamics, we use Poisson and Markov processes to model neuronal spike trains, a well-established method for simulating the stochastic firing behavior of biological neurons.

The rationale for using synthetic sequences with controlled stochastic properties is to enable a rigorous evaluation of how SNNs and biologically inspired learning algorithms respond to different temporal structures and statistical dependencies. Unlike real-world benchmarks, these synthetic inputs allow for precise manipulation of noise, memory, and event distributions, i.e. factors that are crucial for understanding the encoding capabilities of SNNs. Moreover, the use of interpretable, parameterized inputs facilitates a clearer attribution of classification outcomes to underlying neural dynamics and complexity-based decision mechanisms.

Our use of Bernoulli/Markov/Poisson generators is deliberate: these canonical regimes approximate real dynamics while providing controllable ground truth for neuron-level analyses. In particular, inhomogeneous Poisson point processes are standard for event-based neural/biomedical data (e.g., spike trains), and Poisson-intensity GLMs are a workhorse in neural decoding (Truccolo et al., 2004).The time-rescaling theorem maps any simple point process to a unit-rate Poisson via its conditional intensity, and Ogata's thinning supplies unbiased simulation/diagnostics-hence Poisson-family baselines. Although EEG/EMG are continuous, many derived features are event-like (spindles, microstate transitions, thresholded band-power), so Poisson naturally models event times/counts for multi-class temporal classification with clear probabilistic semantics.

Biologically, cortical firing often approximates inhomogeneous Poisson processes over mesoscale windows, capturing ISI variability and trial-to-trial jitter. The resulting controlled stochasticity meaningfully stress-tests SNN encoding and decision mechanisms.

In addition to synthetic spike-train inputs, we evaluate the networks on an event-based MNIST benchmark derived from the standard handwritten digit classification task. Pixel intensities are converted into spike trains using a rate-based encoding, providing a structured but non-random input with spatially meaningful correlations. This dataset serves as an intermediate step between fully controlled synthetic inputs and real biosignals, allowing us to assess whether the

observed model-rule synergies generalize to a widely used classification task.

Related Works

In the paper by Dan et al. (2025), a Spiking Neural Network architecture based on LIF neuron model was proposed. The authors introduced a residual-based SNN architecture with dynamic threshold adjustment, which combines direct encoding (frequency-based neuronal representation) with multineuronal population decoding. The MNIST dataset was considered, and the architecture achieved notable performance in just 6 time-steps, with accuracy improvements ranging between 1.00% and 7.50%, depending on the dataset. Similarly, Luo et al. (2025) proposed an SNN based on a current-based adaptive LIF (CuAdLIF) neuron model featuring delayed responses and membrane potential adaptation. This design improves temporal correlations and maintains long short-term memory. On the other hand, Zhao et al. (2025) advanced neuromorphic computing by developing a high-performance neuromorphic processing unit (NPU) tailored for high data throughput and robust SNN processing. This NPU utilized LIF neurons with a backpropagation-based spike-timing-dependent plasticity learning algorithm, achieving an accuracy of 91.00% on the MNIST dataset. This approach set a new benchmark in neuromorphic computing, offering superior data throughput and neural processing precision compared to systems like SpiNNaker 2.

Study (Liu et al., 2025) applied the STDP learning algorithm to an SNN based on LIF neurons for classification tasks on the MNIST dataset, showing promising results and highlighting the potential of this research direction. Another bio-inspired learning algorithm tested in SNNs is the tempotron, as reported by Yu et al. (2014), who achieved high classification accuracy on the MNIST dataset. Building on this, Patankar et al. (2025) employed a tempotron learning rate method for training and standard STDP for optimization in an SNN based on LIF neurons for medical data applications. This approach significantly improved processing speed and reduced complexity compared to other SNN methods. Various cross-validation techniques were used to validate the robustness of the model, demonstrating its superiority over existing state-of-the-art SNNs. Additionally, Luo et al. (2025) addressed the limitations of SNNs based on meta neuron models by improving the You-Only-Look-Once (YOLO) algorithm. Their SpikeYOLO architecture, i.e. a simplified version of YOLO incorporating meta SNN blocks, which minimized spike degradation and improved detection accuracy.

Despite these advances, none of the aforementioned studies considered the impact of neuron models on SNN performance, particularly the effects on accuracy and computation time in relation to the number of neurons per layer and different neural network learning algorithms. All the reviewed works relied on the LIF neuron model or its variations, such as the meta neuron, and there is a clear trend toward implementing biologically inspired algorithms in SNNs based on LIF neurons. However, these studies did not investigate the influence of network size or the potential of alternative neuron models. Moreover, to our knowledge, no existing research has explored combining SNNs with concepts derived from Information Theory to enhance network accuracy while reducing computation time. This gap in the literature underscores the novelty of our proposed approach.

Results

The considered datasets were divided into two subsets: 90.00% for training and 10.00% for testing, ensuring that the models had sufficient data to learn while retaining a separate evaluation set. Accuracy was used the primary metric for evaluating classification performance. All computations were provided on Intel(R) Core(TM) i7-14700F, 2.10 GHz.

First, we consider four spiking neural networks made of LIF, perceptron, meta neurons, LB neuron model as well as the hybrid network, in which input layer was construed by LIF neurons, while the hidden and output layers were made of perceprtrons, respectively. In and Fig. 2 and Table 5 in Appendix the influence of neuron model on commonly used learning algorithm, i.e. BP learning algorithm was presented. It turned out that all cases gave accuracies above 90.00%, except for the use of a network composed of the LIF model to the Poisson source, however, the computation times differed significantly. The application of the compared to a neural network composed of LIF neurons, the use of the meta nueron model in the case of BP learning algortihms gives higher accuracy in comparable computation time. Surprisingly, for input data in the form of the Bernoulli process (actually the simplest data set), a neural network consisting of 512 LIF neuron in input layer and 512 perceptrons in hidden and output layers was needed to achieve high accuracy. In comparison, architectures composed of the remaining neurons models required only 64 neurons per layer. Moreover, in the case of the Poisson process, the network based on LIF neurons also required four times more epochs to achieves lower accuracy than in other cases.

Figure 3 and Table 6 in Appendix show the influence of neuron model on bio-inspired learning algorithms like tempotron learning rule, Bio-inspired Active Learning (BAL),

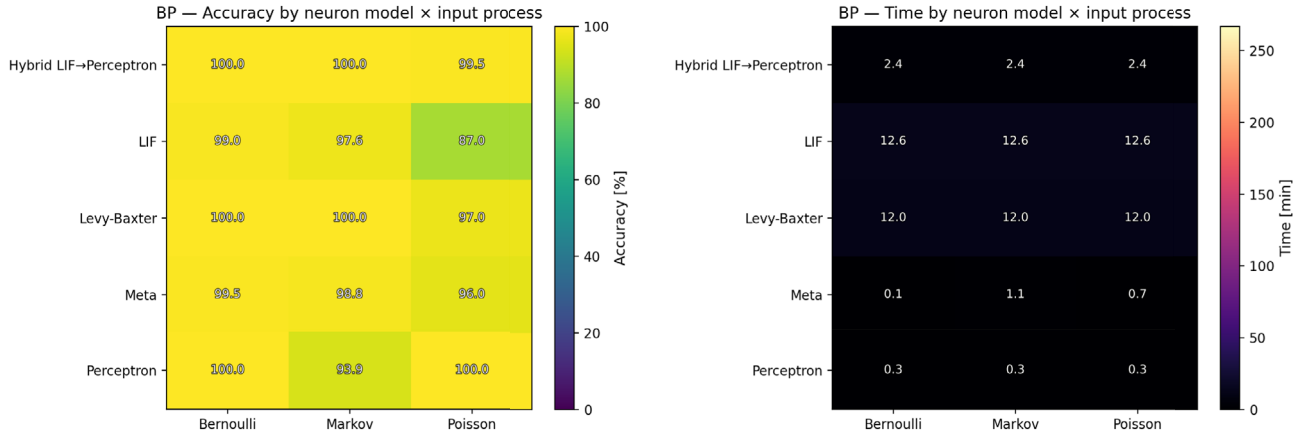
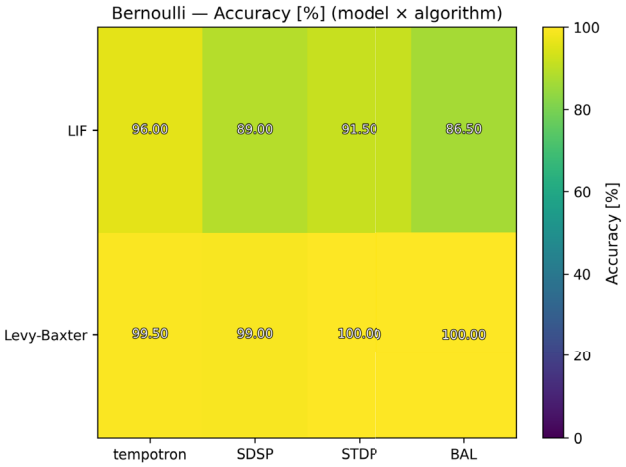


Fig. 2 Comparison of commonly used learning algorithms across neuron models (rows: LIF, LIF input + perceptron hidden/output, Perceptron, Meta, LB) and input processes (columns: Bernoulli, Markov, Poisson). (**a**) Accuracy [%]; (**b**) computation time [min]

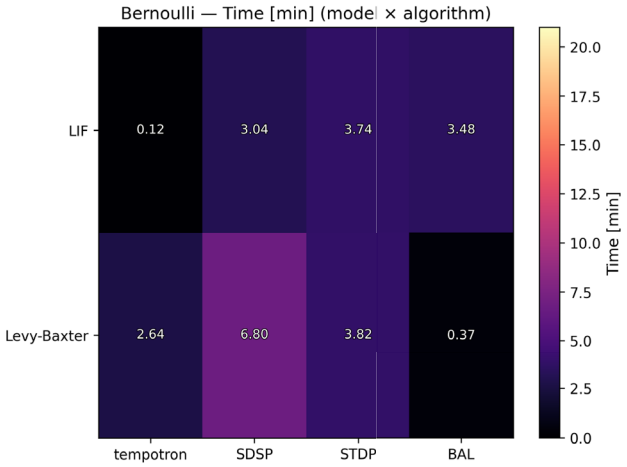
STDP, and SDSP. We consider the same architectures as in Table 5. The results obtained show that involving biologically inspired learning algorithms in the process of training a neural network allows for significantly shortened computation time, especially when the neural network is built from LIF, meta, and LB neuron models. The meta and LB neurons models consistently demonstrated high efficiency in handling Bernoulli processes across different bio-inspired learning algorithms. It achieved a balance between accuracy and computational time, making it well-suited for time-sensitive applications requiring moderate accuracy. The first approach demonstrates high efficacy for Bernoulli sequences, where the independence between events reduces the necessity for complex temporal integration. The Levy-Baxter model, although slightly slower, consistently achieved perfect accuracy across all Bernoulli datasets. The introduction of perceptrons into neural networks presented trade-offs: while they occasionally improved processing speed, particularly in the BAL and tempotron algorithms, they generally led to reduced accuracy and significantly increased training times, as observed in the case of STDP-based algorithms. The Levy-Baxter neural model exhibited superior accuracy, consistently reaching 99.00-100.00% in all scenarios tested. Despite requiring slightly longer training times compared to simpler neuron models, its adaptability and robustness in BAL scenarios make it a strong candidate for accuracy-critical applications. The application of the tempotron learning algorithm enables high-accuracy computations within optimal time constraints. Furthermore, meta-neurons demonstrated an accuracy range of 90 to 100% when implemented in SNNs with 32 to 64 neurons, while other neuron models typically required 128 neurons or more to achieve comparable performance.

The analysis of neuron models under BAL, tempotron, and STDP learning algorithms for Markov sequences revealed distinct performance trends, emphasizing the trade-offs between accuracy and computational efficiency. The perceptron model consistently excelled, achieving perfect accuracy (i.e., 100.00%) in BP and STDP scenarios, demonstrating its effectiveness in capturing Markovian dependencies. Hybrid models integrating LIF neurons with perceptrons exhibited significant potential under BP and bio-inspired learning algorithms, balancing low computational cost with high accuracy. However, these models showed inefficiencies under SDPD, where accuracy declined to 83.00%. The biologically inspired LB model demonstrated exceptional robustness, achieving near-perfect accuracy (93.75–100.00%) across tasks. However, its substantially longer runtimes make it more suitable for precision-critical applications rather than time-sensitive computations. Conversely, meta neurons provided a compelling balance between accuracy and efficiency, maintaining high accuracy while requiring only 32-64 neurons per layer, significantly reducing computational overhead compared to alternative neuron models.

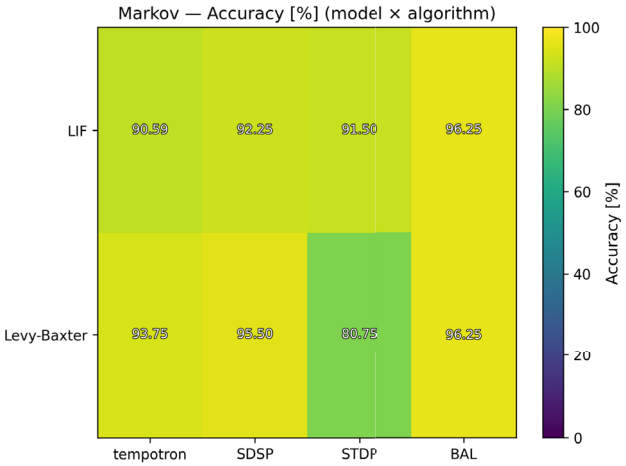
For datasets containing Poisson processes, the perceptron-based neural network model consistently demonstrated the highest efficiency and effectiveness, achieving perfect accuracy (i.e. 100.00%) in minimal time under the tempotron learning algorithm and SDPS. However, in the case of BAL, the computational time increased by an order of magnitude while maintaining the same accuracy. Hybrid models, such as those combining LIF neurons with perceptrons, exhibited significantly lower accuracy, ranging from 68.00% to 89.00%, but with relatively low computational costs. While this approach benefits from step dynamics and linear decision boundaries, it requires careful tuning to



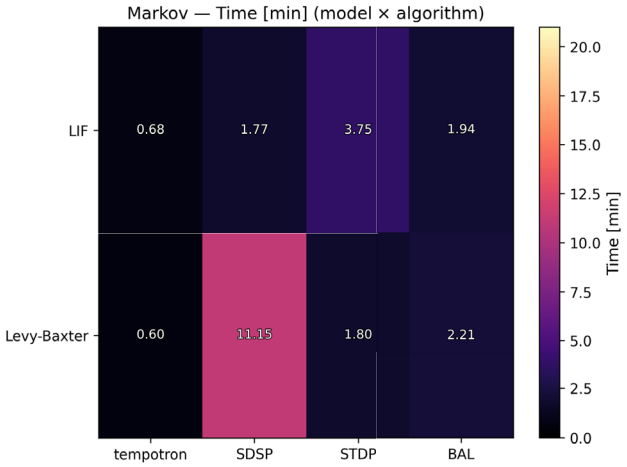
(a) Accuracy (%) (Bernoulli)



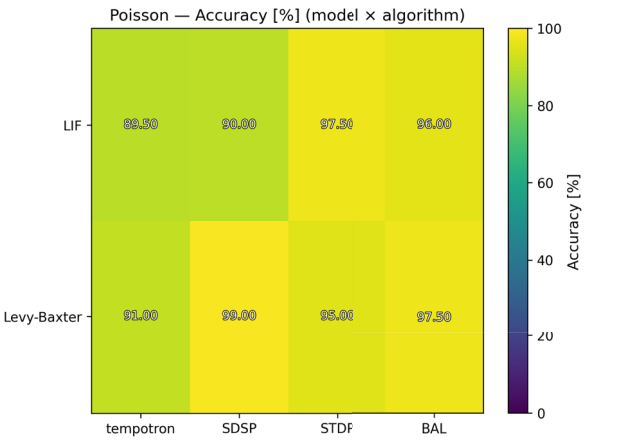
(b) Time (min) (Bernoulli)



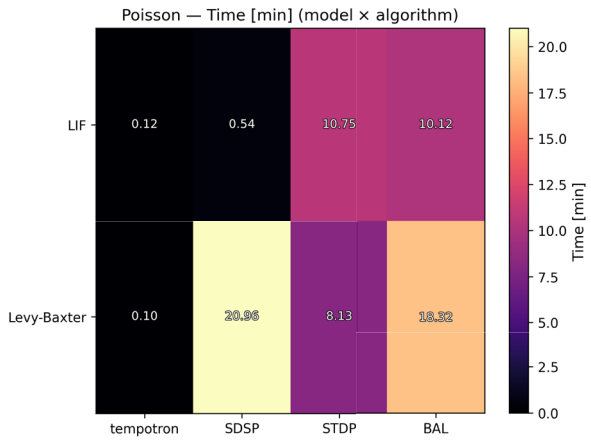
(c) Accuracy (%) (Markov)



(d) Time (min) (Markov)



(e) Accuracy (%) (Poisson)

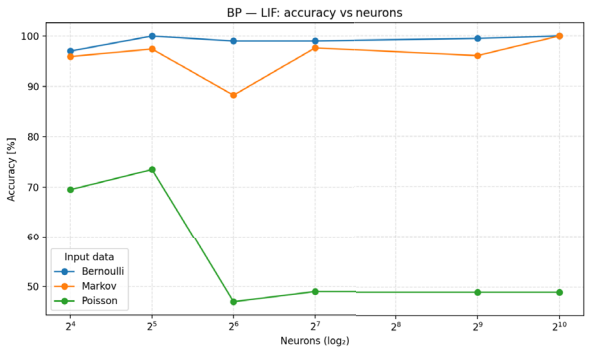


(f) Time (min) (Poisson)

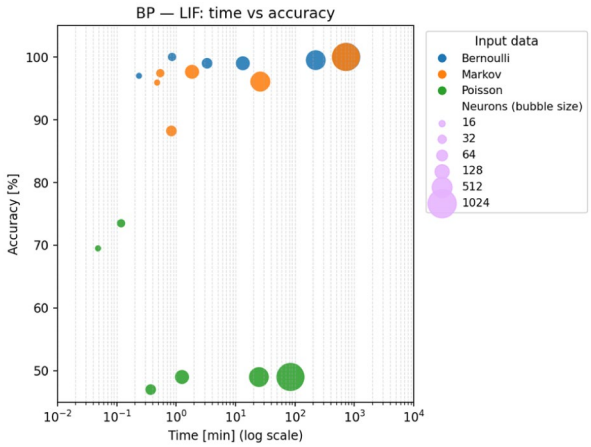
Fig. 3 Bio-inspired algorithms across neuron models and input processes. Panels (**a**,**c**,**e**) show accuracy (%); panels (**b**,**d**,**f**) show computation time (min). Rows of each heatmap correspond to neuron models (LIF, LB), columns to algorithms (Tempotron, SDSP, STDP, BAL)

prevent inefficiencies, particularly with increased learning periods or larger network sizes. Under the SDSP algorithm, a configuration with 16 neurons and 10 epochs achieved an accuracy of 68.00% with a runtime of 28.1 seconds. Similarly, under the STDP algorithm, the same configuration yielded the same accuracy (i.e. 68.00%) but with a significantly reduced runtime of 2.7 seconds, highlighting the computational efficiency of STDP. In contrast, applying the BAL algorithm to a larger network configuration (64 neurons, 20 epochs) resulted in a significantly higher accuracy of 88.00%, with a runtime of 1 minute and 48.0 seconds. Biologically inspired models, such as the LB model, exhibited robustness and representational richness, achieving accuracy levels between 91.00% and 99.00% with 64–128 neurons under bio-inspired learning algorithms, albeit at the cost of increased computational time. In turn, metaneuron networks consistently provided high accuracy with low computational costs for 64 neurons per layer across all bio-inspired learning algorithms, except for tempotron. In the case of the tempotron learning algorithm, achieving comparable accuracy required 128 neurons per layer. Nevertheless, the computational cost remained an order of magnitude lower than in previous cases, reinforcing the efficiency of metaneuron networks.

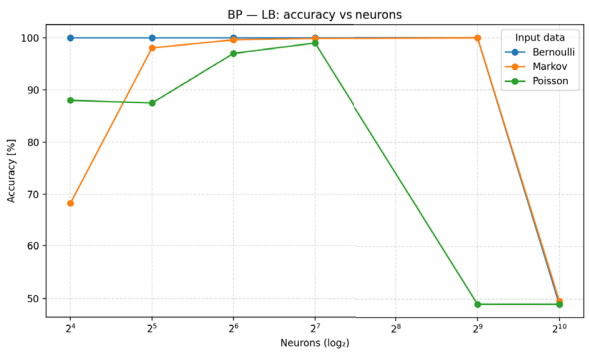
The optimal neuron model and learning algorithm depend on the application's accuracy and efficiency requirements. LB models excel in accuracy-critical tasks, while LIF-based architectures with BAL provide efficient solutions. Hybrid models offer a promising middle ground, performing well when paired with appropriate learning algorithms (Fig. 4).



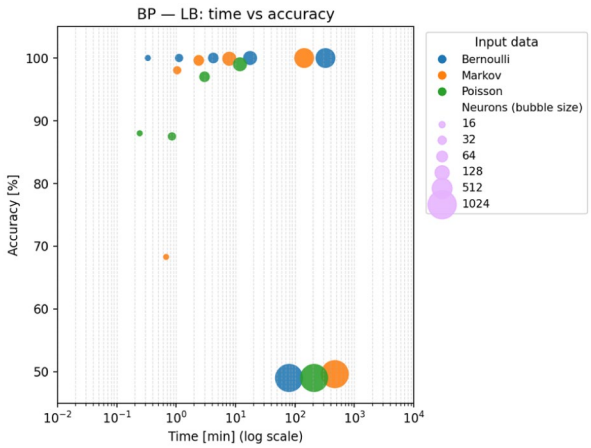
(a) Accuracy (%)



(b) Time (min)



(c) Accuracy (%)



(d) Time (min)

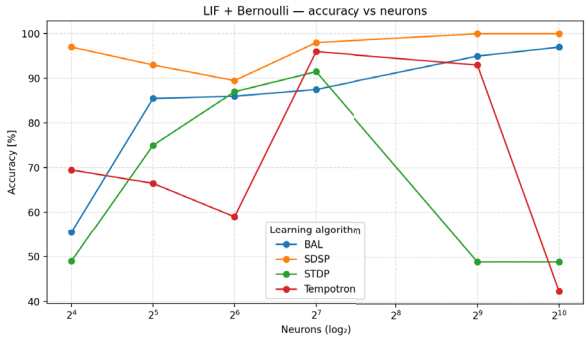
Fig. 4 Results for the LIF and LB neuron models across three input processes: Bernoulli, Markov, Poisson. Rows correspond to input processes. The left column reports accuracy (%) versus the number of neurons per layer ($\log_2$ scale) for a commonly applied learning rule (e.g., BP). The right column shows time–accuracy trade-offs (log-scaled time). Bubble area is proportional to neuron count ($\propto N$), and color encodes the learning algorithm. Colors and size mapping are kept consistent across panels for direct comparison

Also taking into account the results obtained in the Paprocki et al. (2024) paper, namely that a large number of neurons in the network does not necessarily lead to significant improvements in transmission efficiency but can enhance the reliability of the system, we examined the influence of the number of neurons in individual layers on accuracy and computation time. Figures 5 and 6 present time-accuracy trade-off for neuron model with input process. Each marker corresponds to one network configuration trained for 10 epochs. Color encodes the learning algorithm (BP, BAL, SDSP, STDP/STPD, Tempotron). The *x*-axis shows wall-clock training time in seconds on a logarithmic scale (tick marks are powers of ten), and the *y*-axis reports classification accuracy (%). Bubble size encodes model scale: the area of each bubble is proportional to the number of neurons per layer ($\text{area} \propto N$). Consequently, doubling N doubles the bubble area, while the bubble diameter grows only with $\sqrt{N}$. This area-based encoding preserves perceptual proportionality and avoids over-emphasizing very large models. For orientation, the legends include three reference sizes (16 / 128 / 1024 neurons), and all other sizes are interpolated proportionally. The same color and size mappings are used consistently across figures, enabling direct visual comparison. Points closer to the upper-left corner indicate more favorable configurations on the empirical Pareto front (higher accuracy at lower time). In Tables 7, 8, 9 in Appendix, the influence of numbers of neurons on learning algorithms, taking account BP algorithms and bio-inspired learning algorithms were presented. The neural architecture made by LIF and LB neuron model, respectively, were widely investigated. Variants of neural networks that had 16, 32, 64, 128, 512 and 1024 nuerons in each layer were tested, respectively. In the case of BP learning algorithm (see, Table 7) all computation was provided in the 10 epochs. Both networks composed of LIF and LB neurons achieved high accuracy in the case of data from Bernoulli and Markov processes, however, in the case of the Poisson process, the network model based on LIF neurons achieved a maximum accuracy of 73.50 percent with the number of neurons in the layer also 32. Then, as the number of neurons in the layers increased, the accuracy dropped below 50.00%. The network based on the Levy-Baxter neuron model achieved an accuracy of more than 97. 00% at 64, but the computation time was longer than when using the neural network based on the LIF model. In other cases, there is a visible trend towards an increase in precision as the number of neurons in the layers increases.

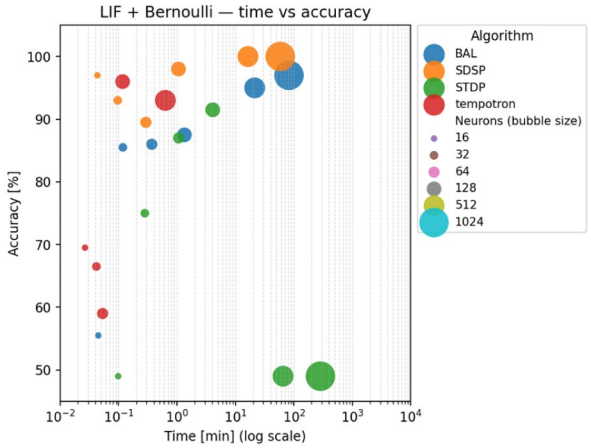
In Table 8 we consider the influence of numbers of neurons in neural networks, which consists of LIF neurons on bio-inspired learning algorithms. Calculations were performed for 10 epochs. In the case of Bernoulli process and temporton learning algorithm increasing the number of neurons in the range 16-64 does not result in a significant increase in accuracy. For 128 and 512 neurons in each layer, we obtain an accuracy of over 93.00%, while for 1024 we obtained only 42.50%. When we classify a two-state Markov process, we get such accuracy over 90.00% in the case of 128, 512 and 1024 neurons in layers. However, for the number of neurons in layers 64 it reaches 82.00%. In turn, this learning algorithm does not work for the Poisson process. Only an accuracy greater than 80.00% is achieved for 128 neurons in each layer. The SDSP algorithm allows to achieve high accuracy regardless of the number of neurons in the layers when classifying Bernoulli and Markov processes, while in the case of the Poisson process only when each layer has 64 neurons, i.e. 88.00%. In the case of the BAL algorithm accuracy increases as the number of neurons in layers increases. Surprisingly, the BAL and SDSP learning algorithms can achieve accuracy above 80.00% only for 64 neurons in layers when classifying Poisson processes. The STPD algorithm gives high accuracy for networks composed of a larger number of neurons, however not exceed 128 neurons in each layer. In turn, the computation times for the tempotron, BAL and SDSP learning algorithms are similar for all data. However, in the case of the STDP algorithm, with a large number of neurons in the layers, the computation time is more than twice as high.

In turn, in Table 9 the results obtained for neural network that consist of LB neuron was shown. Calculations were performed for 10 epochs. It turned out that the use of the tempotron learning algorithm gave high accuracy for all considered data, while the neural network architecture had 128, and 512 neurons in each layer. When the number of neurons in the layers was higher, the accuracy dropped below 50.00%. A similar situation occurred when the number of neurons in the layers was smaller. For example, in the case of tempotron learning rule, the small layer sizes (16, 32 neurons) may lack sufficient representational capacity to capture the temporal and probabilistic dependencies in Bernoulli, Markov, and Poisson processes. This results in underfitting, where the model cannot adequately learn the patterns in the data. On the other hand, large layer sizes like 512, 1024 neurons may overfit the data, particularly for simpler processes. Overfitting can occur when the model memorizes specific patterns instead of generalizing, leading to poor performance on test data. A similar situation is with other learning algorithms, only BAL is exception.

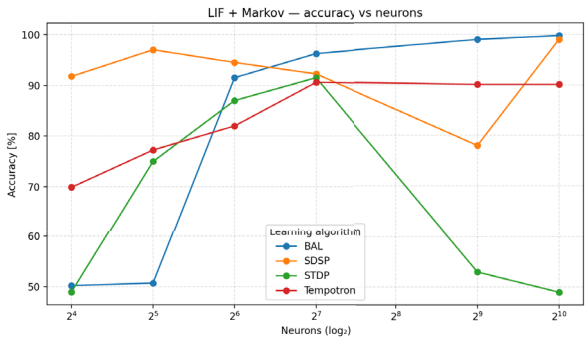
To summarize, our representative results show clear, model-dependent trade-offs in both accuracy and runtime. It turned out that for Poisson-distributed inputs, the most optimal configuration is a perceptron trained with the Tempotron rule, achieving 100.00% accuracy in just 8.5 seconds.



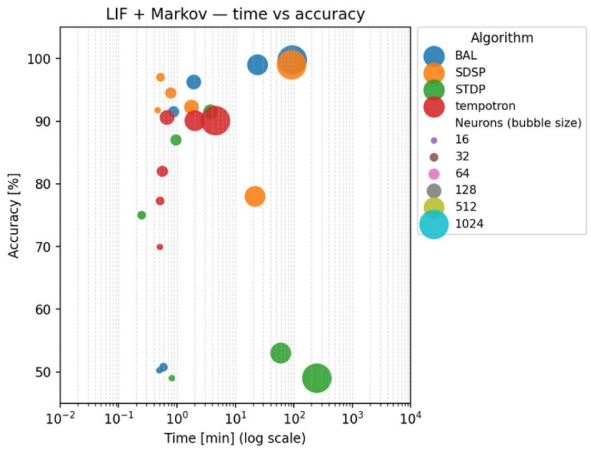
(a) Accuracy (%) (Bernoulli)



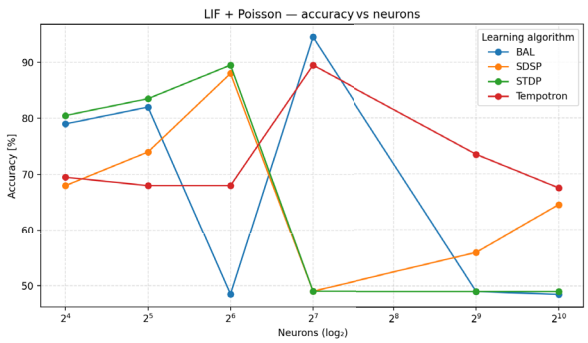
(b) Time (min) (Bernoulli)



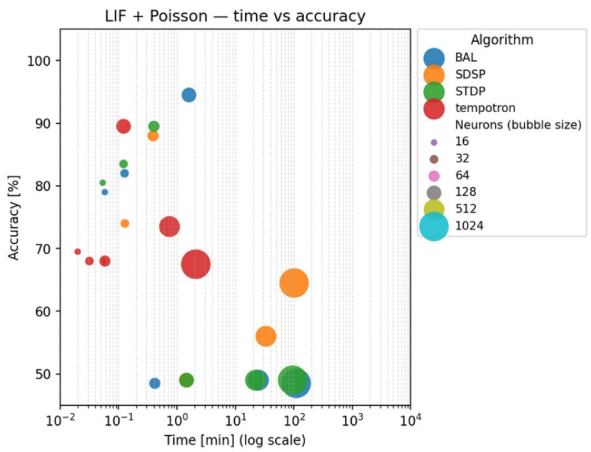
(c) Accuracy (%) (Markov)



(d) Time (min) (Markov)

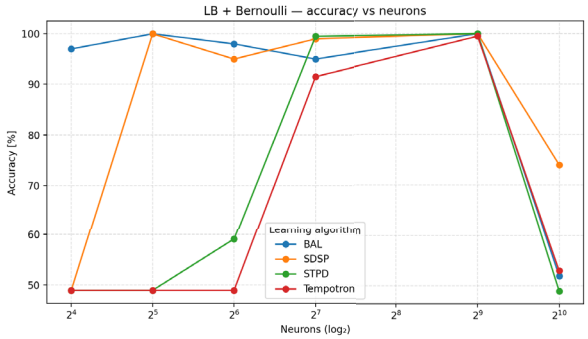


(e) Accuracy (%) (Poisson)

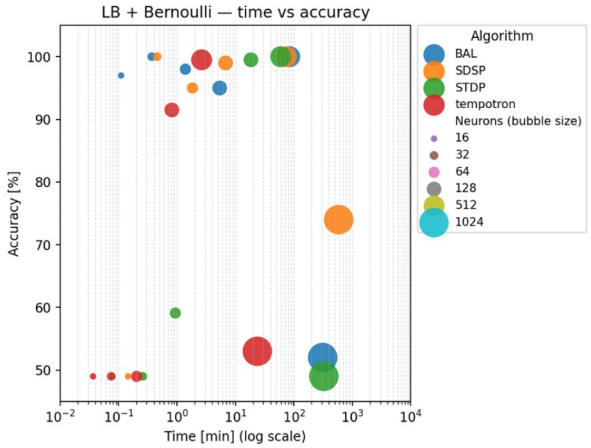


(f) Time (min) (Poisson)

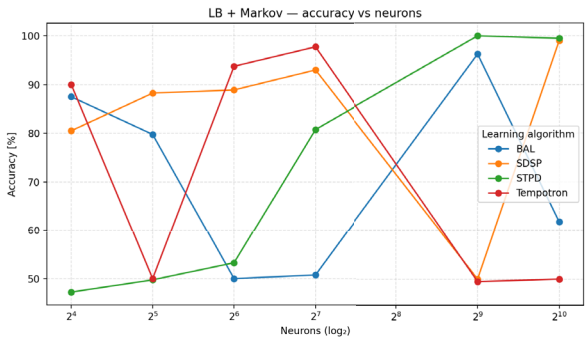
Fig. 5 Results for the LIF neuron model across three input processes: Bernoulli, Markov, Poisson. (**a**,**c**,**e**) Accuracy (%) against number of neurons ($\log_2$); (**b**,**d**,**f**) Time–accuracy trade-offs (log time). Bubble area $\propto N$ and color encodes the algorithm



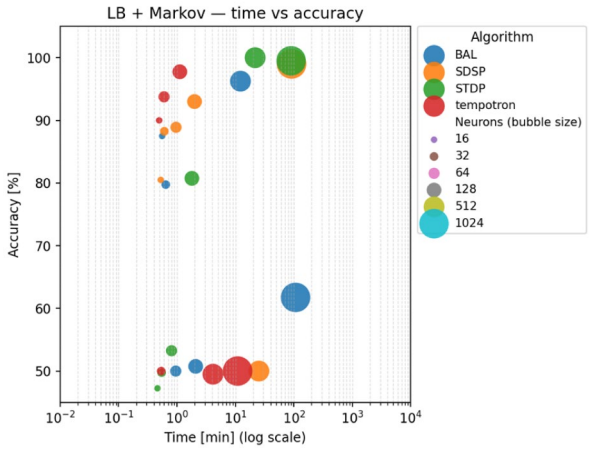
(a) Accuracy (%) (Bernoulli)



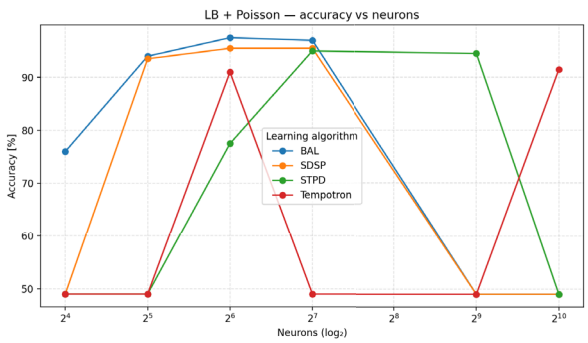
(b) Time (min) (Bernoulli)



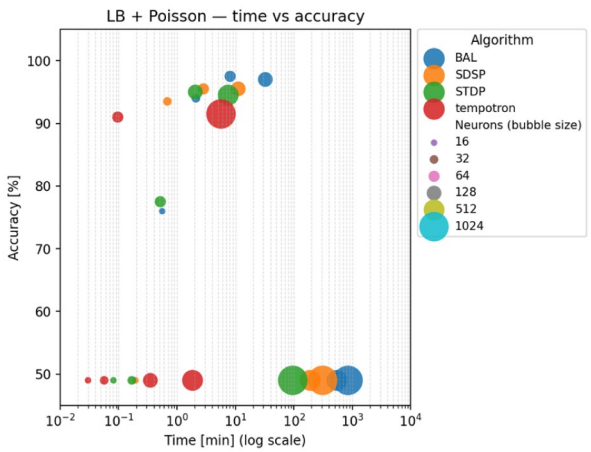
(c) Accuracy (%) (Markov)



(d) Time (min) (Markov)



(e) Accuracy (%) (Poisson)



(f) Time (min) (Poisson)

Fig. 6 Results for the LB neuron model across three input processes: Bernoulli, Markov, Poisson. (**a**,**c**,**e**) Accuracy (%) against number of neurons ($\log_2$); (**b**,**d**,**f**) Time–accuracy trade-offs (log time). Bubble area $\propto N$ and color encodes the algorithm

For Bernoulli data, meta networks trained with BAL or Tempotron offer the best speed-accuracy trade-off under tight computational budgets. If accuracy is prioritized over runtime for Bernoulli inputs, LB models trained with BAL or Tempotron are preferable. In the case of Markov data, LB networks trained with BAL or Tempotron yield the highest accuracy (up to 100.00%) but at a higher computational cost. When balancing speed and performance for Markov input, Tempotron perceptrons remain a strong choice. Overall, model selection should consider both data structure and resource constraints, as optimal configurations vary significantly across input types.

Across datasets, a small set of model-rule pairs consistently dominated, see Table 2. For Poisson inputs, a perceptron trained with the Tempotron rule achieved 100.00% accuracy in 8.5 s with 64 neurons and 10 epochs, offering the best time-accuracy trade-off. For Bernoulli, meta neurons with BAL reached 99.50% in 6.7 s using 32 neurons (fastest high-accuracy option), whereas LB + BAL achieved 100.00% at a modestly higher cost (22.2 s, 32 neurons). For Markov sequences, perceptron + Tempotron attained 100.00% accuracy in 53.6 s with 128 neurons. Increasing layer width beyond 128 neurons rarely helped and sometimes degraded performance (e.g., LIF on Poisson dropped to $\leq 49\%$ for $N \geq 64$). Overall, LB models favor accuracy at higher runtime, meta models strike an efficiency-accuracy balance, and hybrid LIF + perceptron can be competitive on Bernoulli/Markov but lags on Poisson.

On MNIST reference dataset, all architectures except the metaneuron exceed 99.00% accuracy, and the practical differentiator is training cost, see Table 1 in Appendix. A pure LIF network attains the top score ($\approx 99.90\%$) but with the longest wall time, reflecting the expense of simulating membrane dynamics and resets. A hybrid LIF + perceptron (LIF input, perceptron hidden/output) delivers almost the same accuracy with a markedly shorter runtime, indicating that precise temporal encoding at the front end is sufficient while a linear/readout layer can close the decision boundary. A plain perceptron provides a very strong, fast baseline ($\approx 99.30\%$) on this static vision task. The LB model matches LIF-level accuracy but trains slower due to its stochastic synaptic mechanism. For Poisson inputs, the LB neuron reaches 97.00% already with 64 neurons and 10 epochs, underscoring its robustness to stochastic spike trains.

Discussion

The results clearly show that classification performance in SNNs depends strongly on network scale and the statistical structure of input data. Hyperparameters such as learning rate and firing threshold must be tuned to network size; uniform settings often lead to underfitting in small networks or instability in large ones. In high-dimensional architectures, the increased number of synaptic weights expands the parameter space, making optimization more sensitive to initialization and regularization.

The choice of neuron model shapes both computational capacity and task suitability. Simple models like perceptrons are computationally efficient for basic tasks but cannot capture temporal dynamics or nonlinear boundaries without deeper architectures (Parlos et al., 1994). Their performance is especially limited for stochastic inputs such as Poisson processes, where spike variability does not align well with their linear decision structure (Dutta et al., 2017).

More biologically plausible spiking models, e.g., LIF neurons capture temporal coding through spiking dynamics and refractory periods, improving performance on temporal classification tasks. However, they simplify real neuron behavior by omitting channel and synapse dynamics. The Levy-Baxter model combines deterministic and stochastic dynamics, enabling it to represent irregular neural firing more realistically (Levy & Baxter, 2002; Paprocki et al., 2020), at the expense of computational complexity. Meta-neurons (Cheng et al., 2023) aggregate neuron populations into efficient computational units, improving scalability for complex tasks but reducing biological interpretability.

Learning algorithm choice is equally critical: supervised approaches (backpropagation, tempotron) offer high accuracy and fast convergence for well-labeled datasets, while unsupervised rules (STDP, SDSP) are advantageous for limited labels or latent structure discovery. Optimal performance often arises from tailoring the learning rule to both the neuron model and the input statistics.

The input process type Bernoulli, Poisson, or Markov has a measurable effect on model-algorithm combinations. Smaller networks struggle with the temporal complexity of Poisson and Markov sequences, whereas larger networks risk overfitting simpler Bernoulli data. LB neurons combined with tempotron learning perform well for two-state Markov data, effectively capturing transition probabilities (p_{10}, p_{01}).

Moroever, LB neurons behave as renewal units driven by heavy-tailed fluctuations, yielding super-Poisson count variability and burst-quiescence regimes without explicit adaptors. This matches irregularity commonly observed in cortical recordings and explains why LB maintains high accuracy on Poisson/Markov inputs at moderate widths. LIF, by contrast, is near-Poisson unless augmented with history/adaptation, which we empirically observe as either lower accuracy or the need for wider layers. Meta neurons approximate LB-like benefits via learned internal dynamics at lower compute, but with reduced physiological

interpretability. Together with the rate-controlled Δc_2 results, these findings indicate that neuron-level temporal richness (not just network size) is a key driver of downstream performance and efficiency.

For accuracy-critical and temporally complex inputs (Poisson/Markov), LB is preferred. However, for tight compute budgets on simpler inputs like Bernoulli, meta neurons deliver the best accuracy per Multiply–accumulate operations (MACs). In turn, hybrid architecture, which is built with LIF neurons and perceptron is competitive on Bernoulli/Markov but degrades on Poisson unless widened or tuned for refractoriness/history.

It is worth nothing that a key novel aspect of this study is the application of Lempel-Ziv Complexity as a classification mechanism for SNN outputs. By quantifying the structural complexity of spike trains, the LZC approach provides a lightweight and interpretable decision tool that avoids additional classifier layers, aligning well with low-power, real-time neuromorphic applications.

Limitations

This study evaluates SNNs exclusively on synthetic inputs (Bernoulli, Markov, and Poisson spike trains) and on the MNIST benchmark. While such datasets enable controlled manipulation of randomness and temporal dependence, they do not capture several properties of real biosignals, including nonstationarity across sessions and subjects,sensor/physiology artifacts (e.g., motion, electrode drift, burst noise), class imbalance and label uncertainty, and device constraints (quantization, memory, power).

Scalability Across Architectures

We evaluate complexity using the normalized Lempel-Ziv measure c_2 (8), computed on binary spike rasters from any layer or population. Given a spike tensor $S \in \{0,1\}^{T\times C}$ (time $\times$ channels), we form multi-scale temporal bins $b \in \{1,2,4,8\}$, pool spikes within each bin (logical OR or spike counts), compute $c_2 = (C/n)\log_2 n$ per channel (with $n = T/b$), and average over channels. The computational cost is $O(T \cdot C)$ and does not depend on synapse count or layer type. In convolutional SNNs, we treat feature maps as channels and compute c_2 on post-spike tensors; in recurrent/reservoir models, we compute c_2 on reservoir population spikes (optionally layer-wise). In graph-based SNNs, we compute c_2 per node or community and aggregate (mean, trimmed mean, or attention-weighted). Hence, the analysis is architecture-agnostic and directly comparable across models. To illustrate, we include a brief study on a convolutional SNN (STDP layer + linear/ridge readout) and a reservoir SNN (recurrent dynamics + linear readout), where c_2 increases with temporal richness and positively correlates with accuracy (see Tables 3 and 4).

Reproducibility Details

All computations were run on an Intel(R) Core(TM) i7–14700F (2.10 GHz, 20 threads), 32 GB RAM, Windows 10 (10.0.26100, x86_64). Python 3.10.15 (conda– forge); packages: NumPy 2.2.6, SciPy 1.14.1, Matplotlib 3.9.2, Seaborn 0.13.2, scikit– learn 1.6.0, Optuna 4.1.0. NumPy was linked against OpenBLAS 0.3.29. Wall– clock time was measured with `time.perf_counter()` and averaged over 5 seeds ({42,43,44,45,46}). Multiply– accumulate operations (MACs), floating-point operations (FLOPs) are reported as 2×MACs when used.

Conclusions

This study provides a systematic evaluation of how neuron model selection, network size, and learning rule jointly affect SNN classification efficiency. A key contribution is the introduction of a complexity-based classification method that uses Lempel-Ziv Complexity to analyze the structural regularity of spike trains. This method serves as a lightweight, interpretable decision mechanism, replacing conventional output layers and showing strong performance for noisy or temporally irregular signals. Our results show that the interaction between neuron model, network size, and learning algorithm is critical to classification accuracy. Moreover, data characteristics dictate the most effective combination of model-algorithm, with LB neurons and tempotron learning particularly effective for temporally complex data. Also, the LZC-based decision approach complements biologically inspired models by enabling robust, low-power classification, opening paths for neuromorphic applications in biosignal processing. Our results are obtained on synthetic inputs and further validated on the MNIST dataset. However, validating the LZC-based decision mechanism and model-rule trade-offs on real biosignal datasets is therefore an essential next step and a current limitation of this study. Future work will extend this framework to other real biosignal datasets, explore adaptive LZC thresholds, and investigate hardware implementation for real-time SNN deployment.

Appendix A: Summary of Numerical Results

Table 5 Influence of neuron model on commonly used learning algorithm

Input data	Neuron model	Neurons / layer	Epochs	Time	Accuracy [%]
Bernoulli	LIF	128	10	12m36.8s	99.00
Bernoulli	LIF in input layer + perceptron in hidden and output layers	512	20	2m26.7s	100.00
Bernoulli	Perceptron	128	10	17.0s	100.00
Bernoulli	Meta	32	10	7.8s	99.50
Bernoulli	Levy-Baxter	128	20	11m58.6s	100.00
Markov	LIF	128	10	12m36.8s	97.64
Markov	LIF in input layer + perceptron in hidden and output layers	64	20	2m26.7s	100.00
Markov	Perceptron	32	20	17.0s	93.94
Markov	Meta	32	10	1m7.4s	98.75
Markov	Levy-Baxter	64	20	11m58.6s	100.00
Poisson	LIF	64	40	12m36.8s	87.00
Poisson	LIF in input layer + perceptron in hidden and output layers	64	20	2m26.7s	99.50
Poisson	Perceptron	64	10	17.0s	100.00
Poisson	Meta	32	10	40.7s	96.00
Poisson	Levy-Baxter	64	10	11m58.6s	97.00
MNIST	LIF	128	10	27m3.3s	99.86
MNIST	LIF in input layer + perceptron in hidden and output layers	784	20	4m34.5s	99.66
MNIST	Perceptron	256	20	4m41.05s	99.36
MNIST	Meta	32	20	62m55.0s	87.69
MNIST	Levy-Baxter	784	20	16m59.4s	99.86

The BP learning algorithm was applied

Table 6 Influence of neuron model on bio-inspired learning algorithms

Input	Neuron model	Algorithm	Neurons / layer	Epochs	Time	Acc. [%]
Bernoulli	LIF	Tempotron	128	10	7.02	96.00
Bernoulli	LIF (in) + Perceptron (hid/out)	Tempotron	128	10	19.9s	100.00
Bernoulli	Perceptron	Tempotron	128	10	19.6s	100.00
Bernoulli	Meta	Tempotron	64	40	1m24.4s	90.50
Bernoulli	Levy-Baxter	Tempotron	128	10	2m38.4s	99.50
Bernoulli	LIF	SDSP	128	45	3m2.3s	89.00
Bernoulli	LIF (in) + Perceptron (hid/out)	SDSP	128	10	12m1.7s	87.50
Bernoulli	Perceptron	SDSP	128	10	17s	100.00
Bernoulli	Meta	SDSP	32	10	3m56.0s	100.00
Bernoulli	Levy-Baxter	SDSP	128	10	6m48.0s	99.00
Bernoulli	LIF	STPD	128	10	3m44.4s	91.50
Bernoulli	LIF (in) + Perceptron (hid/out)	STPD	128	10	11m50.9s	87.50
Bernoulli	Perceptron	STPD	512	10	86m48.8s	100.00
Bernoulli	Meta	STPD	32	10	20.2s	92.50
Bernoulli	Levy-Baxter	STPD	128	10	3m49s	100.00
Bernoulli	LIF	BAL	128	30	3m28.9s	86.50

Table 6 (continued)

Input	Neuron model	Algorithm	Neurons / layer	Epochs	Time	Acc. [%]
Bernoulli	LIF (in) + Perceptron (hid/out)	BAL	128	10	4.7s	87.50
Bernoulli	Perceptron	BAL	128	10	5m38.5s	99.00
Bernoulli	Meta	BAL	32	10	6.7	99.50
Bernoulli	Levy-Baxter	BAL	32	10	22.2s	100.00
Markov	LIF	Tempotron	128	10	40.5s	90.59
Markov	LIF (in) + Perceptron (hid/out)	Tempotron	64	10	29.1s	96.75
Markov	Perceptron	Tempotron	128	10	53.6s	100.00
Markov	Meta	Tempotron	64	10	1m8.2s	92.00
Markov	Levy-Baxter	Tempotron	64	10	36.0s	93.75
Markov	LIF	SDSP	128	10	1m46.4s	92.25
Markov	LIF (in) + Perceptron (hid/out)	SDSP	256	10	93m2.4s	83.00
Markov	Perceptron	SDSP	128	20	3m44.7s	100.00
Markov	Meta	SDSP	64	10	53.1s	97.53
Markov	Levy-Baxter	SDSP	128	10	11m9.2s	95.50
Markov	LIF	STPD	128	10	3m45.1s	91.50
Markov	LIF (in) + Perceptron (hid/out)	STPD	64	10	1m17.3s	85.00
Markov	Perceptron	STPD	512	10	195m34.2s	100.00
Markov	Meta	STPD	64	10	51.3s	99.20
Markov	Levy-Baxter	STPD	128	10	1m47.9s	80.75
Markov	LIF	BAL	128	10	1m56.2s	96.25
Markov	LIF (in) + Perceptron (hid/out)	BAL	256	20	11m12.9s	95.00
Markov	Perceptron	BAL	512	10	266m55.1s	99.75
Markov	Meta	BAL	32	10	7.6s	99.50
Markov	Levy-Baxter	BAL	64	10	2m12.4s	96.25
Poisson	LIF	Tempotron	128	10	7.3s	89.50
Poisson	LIF (in) + Perceptron (hid/out)	Tempotron	128	20	3m23.9s	89.00
Poisson	Perceptron	Tempotron	64	10	8.5s	100.00
Poisson	Meta	Tempotron	128	10	7.8s	89.50
Poisson	Levy-Baxter	Tempotron	64	10	5.8s	91.00
Poisson	LIF	SDSP	64	10	32.2s	90.00
Poisson	LIF (in) + Perceptron (hid/out)	SDSP	16	10	28.1s	68.00
Poisson	Perceptron	SDSP	16	10	3.1s	100.00
Poisson	Meta	SDSP	64	10	26.1s	91.00
Poisson	Levy-Baxter	SDSP	128	10	20m57.7s	99.00
Poisson	LIF	STPD	128	20	10m44.7s	97.50
Poisson	LIF (in) + Perceptron (hid/out)	STPD	16	10	2.7s	68.00
Poisson	Perceptron	STPD	64	10	10.7s	91.00
Poisson	Meta	STPD	64	10	31.9s	88.00
Poisson	Levy-Baxter	STPD	128	10	8m7.9s	95.00
Poisson	LIF	BAL	128	30	10m6.9s	96.00
Poisson	LIF (in) + Perceptron (hid/out)	BAL	64	20	1m48.0s	88.00
Poisson	Perceptron	BAL	64	10	1m54.0s	100.00
Poisson	Meta	BAL	64	10	31.4s	91.00
Poisson	Levy-Baxter	BAL	64	10	18m19.0s	97.50

Table 7 Influence of numbers of neurons on commonly applied learning algorithms

Neuron model	Input data	Neurons / layer	Epochs	Time	Accuracy [%]
LIF	Bernoulli	16	10	14.2s	97.00
LIF	Bernoulli	32	10	51.2s	100.00
LIF	Bernoulli	64	10	3m19s	99.00
LIF	Bernoulli	128	10	13m16.5s	99.00
LIF	Bernoulli	512	10	223m53.5s	99.50
LIF	Bernoulli	1024	10	729m36s	100.00
LIF	Markov	16	10	28.7s	95.92
LIF	Markov	32	10	32.4s	97.42
LIF	Markov	64	10	49.9s	88.22
LIF	Markov	128	10	1m51.0s	97.64
LIF	Markov	512	10	26m9.0s	96.08
LIF	Markov	1024	10	720m4.8s	100.00
LIF	Poisson	16	10	2.9s	69.50
LIF	Poisson	32	10	7.1s	73.50
LIF	Poisson	64	10	22.3s	47.00
LIF	Poisson	128	10	1m15.2s	49.00
LIF	Poisson	512	10	24m48.0s	49.00
LIF	Poisson	1024	10	84m10.5s	49.00
LB	Bernoulli	16	10	20.0s	100.00
LB	Bernoulli	32	10	1m7.1s	100.00
LB	Bernoulli	64	10	4m12.9s	100.00
LB	Bernoulli	128	10	17m34.1s	100.00
LB	Bernoulli	512	10	326m32.5s	100.00
LB	Bernoulli	1024	10	79m48.4s	49.00
LB	Markov	16	10	40.6s	68.29
LB	Markov	32	10	1m2.3s	98.05
LB	Markov	64	10	2m24.2s	99.62
LB	Markov	128	10	7m51.3s	99.88
LB	Markov	512	10	143m9.1s	100.00
LB	Markov	1024	10	470m1.1s	49.62
LB	Poisson	16	10	14.6s	88.00
LB	Poisson	32	10	51.0s	87.50
LB	Poisson	64	10	3m0.1s	97.00
LB	Poisson	128	10	11m51.5s	99.00
LB	Poisson	512	10	19m45.6s	49.00
LB	Poisson	1024	10	210m11.0s	49.00

The BP learning algorithm was applied

Table 8 Influence of numbers of neurons on bio-inspired learning algorithms

Input data	Algorithm	Neurons / layer	Epochs	Time	Acc. [%]
Bernoulli	Tempotron	16	10	1.6s	69.50
Bernoulli	Tempotron	32	10	2.5s	66.50
Bernoulli	Tempotron	64	10	3.2s	59.00
Bernoulli	Tempotron	128	10	7.02s	96.00
Bernoulli	Tempotron	512	10	38.0s	93.00
Bernoulli	Tempotron	1024	10	1m41.9s	42.50
Bernoulli	BAL	16	10	2.7s	55.50
Bernoulli	BAL	32	10	7.1s	85.50
Bernoulli	BAL	64	10	22.2s	86.00
Bernoulli	BAL	128	10	1m20.3s	87.50
Bernoulli	BAL	512	10	21m21.6s	95.00
Bernoulli	BAL	1024	10	82m27.9s	97.00
Bernoulli	STDP	16	10	5.9s	49.00
Bernoulli	STDP	32	10	16.9s	75.00
Bernoulli	STDP	64	10	1m3.9s	87.00
Bernoulli	STDP	128	10	4m5.3s	91.50
Bernoulli	STDP	512	10	65m21.0s	49.00
Bernoulli	STDP	1024	10	285m6.4s	49.00
Bernoulli	SDSP	16	10	2.6s	97.00
Bernoulli	SDSP	32	10	5.8s	93.00
Bernoulli	SDSP	64	10	17.6s	89.50
Bernoulli	SDSP	128	10	1m3.5s	98.00
Bernoulli	SDSP	512	10	16m23.5s	100.00
Bernoulli	SDSP	1024	10	58m36.4s	100.00
Markov	Tempotron	16	10	30.6s	69.93
Markov	Tempotron	32	10	30.9s	77.27
Markov	Tempotron	64	10	33.7s	82.00
Markov	Tempotron	128	10	40.5s	90.59
Markov	Tempotron	512	10	2m1.7s	90.08
Markov	Tempotron	1024	10	4m33.1s	90.08
Markov	STDP	16	10	49s	49.00
Markov	STDP	32	10	14.9s	75.00
Markov	STDP	64	10	57.5s	87.00
Markov	STDP	128	10	3m45.1s	91.50
Markov	STDP	512	10	59m33.4s	53.00
Markov	STDP	1024	10	247m21.1s	49.00
Markov	BAL	16	10	30.0s	50.25
Markov	BAL	32	10	35.1s	50.75
Markov	BAL	64	10	52.5s	91.50
Markov	BAL	128	10	1m56.2s	96.25
Markov	BAL	512	10	23m57s	99.00

Table 8 (continued)

Input data	Algorithm	Neurons / layer	Epochs	Time	Acc. [%]
Markov	BAL	1024	10	93m53.3s	99.75
Markov	SDSP	16	10	28.1s	91.75
Markov	SDSP	32	10	31.4s	97.00
Markov	SDSP	64	10	47.0s	94.50
Markov	SDSP	128	10	1m46.4s	92.25
Markov	SDSP	512	10	21m40.8s	78.00
Markov	SDSP	1024	10	91m24.0s	99.00
Poisson	Tempotron	16	10	1.2s	69.50
Poisson	Tempotron	32	10	1.9s	68.00
Poisson	Tempotron	64	10	3.5s	68.00
Poisson	Tempotron	128	10	7.3s	89.50
Poisson	Tempotron	512	10	44.6s	73.50
Poisson	Tempotron	1024	10	2m6s	67.50
Poisson	BAL	16	10	3.5s	79.00
Poisson	BAL	32	10	7.6s	82.00
Poisson	BAL	64	10	25.0s	48.50
Poisson	BAL	128	10	1m36.3s	94.50
Poisson	BAL	512	10	24m48.1s	49.00
Poisson	BAL	1024	10	109m38.3s	48.50
Poisson	STDP	16	10	3.2s	80.50
Poisson	STDP	32	10	7.3s	83.50
Poisson	STDP	64	10	24.1s	89.50
Poisson	STDP	128	10	1m27.5s	49.00
Poisson	STDP	512	10	22m8.5s	49.00
Poisson	STDP	1024	10	95m35.3s	49.00
Poisson	SDSP	16	10	3.4s	68.00
Poisson	SDSP	32	10	7.7s	74.00
Poisson	SDSP	64	10	23.4s	88.00
Poisson	SDSP	128	10	1m26.6s	49.00
Poisson	SDSP	512	10	33m17.0s	56.00
Poisson	SDSP	1024	10	101m5.0s	64.50

The LIF neuron model was applied

Table 9 Influence of numbers of neurons on bio-inspired learning algorithms

Input data	Algorithm	Neurons / layer	Epochs	Time	Acc. [%]
Bernoulli	Tempotron	16	10	2.2s	49.00
Bernoulli	Tempotron	32	10	4.5s	49.00
Bernoulli	Tempotron	64	10	12.2s	49.00
Bernoulli	Tempotron	128	10	49.0s	91.50
Bernoulli	Tempotron	512	10	2m38.4s	99.50
Bernoulli	Tempotron	1024	10	23m45.1s	53.00
Bernoulli	SDSP	16	10	8.7s	49.00
Bernoulli	SDSP	32	10	27.5s	100.00
Bernoulli	SDSP	64	10	1m50.5s	95.00
Bernoulli	SDSP	128	10	6m48.0s	99.00
Bernoulli	SDSP	512	10	74m51.5s	100.00
Bernoulli	SDSP	1024	10	585m52.5s	74.00
Bernoulli	STPD	16	10	4.6s	49.00
Bernoulli	STPD	32	10	15.6s	49.00
Bernoulli	STPD	64	10	56.1s	59.10
Bernoulli	STPD	128	10	18m27.7s	99.50
Bernoulli	STPD	512	10	59m20.9s	100.00
Bernoulli	STPD	1024	10	325m51.4s	49.00
Bernoulli	BAL	16	10	6.6s	97.00

Table 9 (continued)

Input data	Algorithm	Neurons / layer	Epochs	Time	Acc. [%]
Bernoulli	BAL	32	10	22.2s	100.00
Bernoulli	BAL	64	10	1m23.5s	98.00
Bernoulli	BAL	128	10	5m22.5s	95.00
Bernoulli	BAL	512	10	85m5.5s	100.00
Bernoulli	BAL	1024	10	311m47s	52.00
Markov	Tempotron	16	10	29.7s	90.00
Markov	Tempotron	32	10	32.2s	50.00
Markov	Tempotron	64	10	36.0s	93.75
Markov	Tempotron	128	10	1m7.0s	97.75
Markov	Tempotron	512	10	4m7.8s	49.50
Markov	Tempotron	1024	10	10m53.1s	50.00
Markov	SDSP	16	10	31.6s	80.50
Markov	SDSP	32	10	36.4s	88.25
Markov	SDSP	64	10	57.65s	88.89
Markov	SDSP	128	10	1m59.65s	93.00
Markov	SDSP	512	10	25m9.65s	50.00
Markov	SDSP	1024	10	91m24.0s	99.00
Markov	STPD	16	10	27.8s	47.25
Markov	STPD	32	10	32.7s	49.75
Markov	STPD	64	10	48.4s	53.25
Markov	STPD	128	10	1m47.9s	80.75
Markov	STPD	512	10	21m44.3s	100.00
Markov	STPD	1024	10	89m45.9s	99.50
Markov	BAL	16	10	33.4s	87.50
Markov	BAL	32	10	38.5s	79.75
Markov	BAL	64	10	57.1s	50.00
Markov	BAL	128	10	2m5.2s	50.75
Markov	BAL	512	10	12m12.4s	96.25
Markov	BAL	1024	10	106m54.7s	61.75
Poisson	Tempotron	16	10	1.8s	49.00
Poisson	Tempotron	32	10	3.4s	49.00
Poisson	Tempotron	64	10	5.8s	91.00
Poisson	Tempotron	128	10	21.0s	49.00
Poisson	Tempotron	512	10	1m50.7s	49.00
Poisson	Tempotron	1024	10	5m39.6s	91.50
Poisson	SDSP	16	10	11.7s	49.00
Poisson	SDSP	32	10	41s	93.50
Poisson	SDSP	64	10	2m49.9s	95.50
Poisson	SDSP	128	10	11m9.2s	95.50
Poisson	SDSP	512	10	190m53.1s	49.00
Poisson	SDSP	1024	10	311m48s	49.00
Poisson	STPD	16	10	4.9s	49.00
Poisson	STPD	32	10	10.1s	49.00
Poisson	STPD	64	10	31.1s	77.50
Poisson	STPD	128	10	2m3.2s	95.00
Poisson	STPD	512	10	7m31s	94.50
Poisson	STPD	1024	10	95m50.7s	49.00
Poisson	BAL	16	10	33.4s	76.00
Poisson	BAL	32	10	2m6.3s	94.00
Poisson	BAL	64	10	8m6.7s	97.50
Poisson	BAL	128	10	32m29.2s	97.00
Poisson	BAL	512	10	546m25.4s	49.00
Poisson	BAL	1024	10	845m38.7s	49.00

The LB neuron model was applied

Information Sharing Statement

Author Contributions All authors contributed to the conception and design of the study. All authors performed material preparation, data collection, and analysis. The first draft of the manuscript was written by all authors who commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Not applicable.

Data Availability No datasets were generated or analysed during the current study.

Materials availability Not applicable.

Code availability Not applicable.

Declarations

Conflicts of Interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethics approval and consent to participate The authors affirm that this work adheres to the highest ethical standards of scientific publication, in accordance with COPE guidelines.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

References

Arnold, M. M., Szczepanski, J., Montejo, N., Amigó, J. M., Wajnryb, E., & Sanchez-Vives, M. V. (2013). Information content in cortical spike trains during brain state transitions. *Journal of Sleep Research, 22*, 13–21. https://doi.org/10.1111/j.1365-2869.2012.01031.x

Beyeler, M., Richert, M., Dutt, N. D., & Krichmar, J. L. (2014). Efficient spiking neural network model of pattern motion selectivity in visual cortex. *Neuroinformatics, 12*, 435–454. https://doi.org/10.1007/s12021-014-9220-y

Cheng, X., Zhang, T., Jia, S., & Xu, B. (2023). Meta neurons improve spiking neural networks for efficient spatio-temporal learning. *Neurocomputing, 531*, 217–225. https://doi.org/10.1016/j.neucom.2023.02.029

Daley, D. J., & Vere-Jones, D. (2003). An introduction to the theory of point processes. Springer-Verlag, New York. https://doi.org/10.1007/b97277

Dan, Y., Sun, C., Li, H., & Meng, L. (2025). Adaptive spiking neuron with population coding for a residual spiking neural network. *Applied Intelligence, 5*, 288. https://doi.org/10.1007/s10489-024-06128-z

Datta, G., Kundu, S., Jaiswal, A. R., & Beerel, P. A. (2022). ACE-SNN: Algorithm-hardware Co-design of energy-efficient & low-latency deep spiking neural networks for 3D image recognition. *Frontiers in Neuroscience,* 815258. https://doi.org/10.3389/fnins.2022.815258

Dutta, S., Kumar, V., Shukla, A., Mohapatra, N. R., & Ganguly, U. (2017). Leaky integrate and fire neuron by charge-discharge dynamics in floating-body MOSFET. *Scientific Reports, 7*, 8257. https://doi.org/10.1038/s41598-017-07418-y

Gerstner, W., Kistler, W., Naud, R., & Paninski, L. (2014). Neuronal dynamics from single neurons to networks and models of cognition. Cambridge University Press. https://doi.org/10.1017/CBO9781107447615

Gutig, R., & Sompolinsky, H. (2006). The tempotron: a neuron that learns spike timing-based decisions. *Nature Neuroscience, 9*(3), 420–428. https://doi.org/10.1038/nn1643

He, S., Peng, Y., Wang, H., & Ma, M. (2025). Discrete memristor synapse-driven spiking neural networks: Dynamics of firing and synaptic plasticity. *Nonlinear Dynamics*. https://doi.org/10.1007/s11071-025-11579-1

Izhikevich, E. M. (2003). Simple model of spiking neurons. *IEEE Transactions on Neural Networks, 14*(6), 1569–1572. https://doi.org/10.1109/TNN.2003.820440

Levy, W., & Baxter, R. (2002). Energy-efficient neuronal computation via quantal synaptic failures. *The Journal of Neuroscience, 22*, 4746–4755. https://doi.org/10.1523/JNEUROSCI.22-11-04746.2002

Liu, X., Mo, L., & Tang, M. (2025). TinySpiking: a lightweight and efficient python framework for unsupervised learning spiking neural networks. *Engineering Research Express, 7*, 015217. https://doi.org/10.1088/2631-8695/ada725

Ljungquist, B., Petersson, P., Johansson, A. J., Schouenborg, J., & Garwicz, M. (2018). A bit-encoding based new data structure for time and memory efficient handling of spike times in an electrophysiological setup. *Neuroinformatics, 16*, 217–229. https://doi.org/10.1007/s12021-018-9367-z

Luo, X., Yao, M., Chou, Y., Xu, B., & Li, G. (2025). Integer-valued training and spike-driven inference spiking neural network for high-performance and energy-efficient object detection. In: A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, & G. Varol (Eds.), *Computer Vision – ECCV 2024*. ECCV 2024. Lecture Notes in Computer Science, vol 15090. Springer, Cham. https://doi.org/10.1007/978-3-031-73411-3_15

Naderi, R., Rezaei, A., Amiri, M., & Peremans, H. (2025). Unsupervised post-training learning in spiking neural networks. *Scientific Reports, 15*, 17647. https://doi.org/10.1038/s41598-025-01749-x

Papoulis, A., & Pillai, S. U. (2002). *Probability, random variables, and stochastic processes*. Fourth Boston: McGraw Hill.

Paprocki, B., Pregowska, A., & Szczepanski, J. (2020). Optimizing information processing in brain-inspired neural networks. *BPAS Technical Sciences, 68*(2), 225–233. https://doi.org/10.24425/bpasts.2020.131844

Paprocki, B., Pregowska, A., & Szczepanski, J. (2024). Does adding of neurons to the network layer lead to increased transmission efficiency? *IEEE Access, 12*, 42701–42709. https://doi.org/10.1016/10.1109/ACCESS.2024.3379324

Parlos, A. G., Chong, K. Y., & Atiya, A. F. (1994). Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks, 5*(2), 255–266. https://doi.org/10.1109/72.279189

Patankar, M., Chaurasia, V., & Shandilya, M. (2025). A novel spiking neural network method for classification of tuberculosis using X-ray images. *Computers and Electrical Engineering, 122*, 110003. https://doi.org/10.1016/j.compeleceng.2024.110003

Pregowska, A., Proniewska, K., van Dam, P., & Szczepanski, J. (2019). Using Lempel-Ziv complexity as effective classification tool of the sleep-related breathing disorders. *Computer Methods and Programs in Biomedicine, 182*, 105052-1–7. https://doi.org/10.1016/j.cmpb.2019.105052

Pregowska, A., Szczepanski, J., & Wajnryb, E. (2016). Temporal code versus rate code for binary Information Sources. *Neurocomputing, 216*, 756–762. https://doi.org/10.1016/j.neucom.2016.08.034

Richmond, P., Cope, A., Gurney, K., & Allerton, D. J. (2014). From model specification to simulation of biologically constrained networks of spiking neurons. *Neuroinformatics, 12*, 307–323. https://doi.org/10.1007/s12021-013-9208-z

Rieke, F., Warland, D., van Steveninck, R. R., & Bialek, W. (1997). Spikes. *Exploring the neural code*. MIT Press, Cambridge. https://doi.org/10.5555/319283

Seguin, C., Sporns, O., & Zalesky, A. (2023). Brain network communication: Concepts, models and applications. *Nature Reviews Neuroscience, 24*(9), 557–574. https://doi.org/10.1038/s41583-023-00718-5

Shannon, C. E. (1984). A mathematical theory of communication. *Bell System Technical Journal, 27*, 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Shirsavar, S. R., Vahabie, A. H., & Dehaqani, M. A. (2023). Models developed for spiking neural networks. *MethodsX, 10*, 102157. https://doi.org/10.1016/j.mex.2023.102157

Sun, P., Wu, J., Devos, P., & Botteldooren, D. (2025). Towards parameter-free attentional spiking neural networks. *Neural Networks, 185*, 107154. https://doi.org/10.1016/j.neunet.2025.107154

Truccolo, W., Eden, U. T., Fellows, M. R., Donoghue, J. P., & Brown, E. N. (2004). A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology, 93*(2), 1074–1089. https://doi.org/10.1152/jn.00697.2004

Weng, J. K., Ding, Y. J., Hu, C. B., Zhu, X. F., Liang, B., Yang, J., & Cheng, J. C. (2020). Meta-neural-network for real-time and passive deep-learning-based object recognition. *Nature Communications, 11*(1), 6309. https://doi.org/10.1038/s41467-020-19693-x

Wojcik, D., Mochol, G., Jakuczun, W., Wypych, M., & Waleszczyk, W. J. (2009). Direct estimation of inhomogeneous Markov interval models of spike trains. *Neural Computation, 21*(8), 2105–2113. https://doi.org/10.1162/neco.2009.07-08-828

Wu, Y., Deng, L., Li, G., Zhu, J., & Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience, 12*, 331. https://doi.org/10.3389/fnins.2018.00331

Yu, Q., Tang, H., Tan, K. C., & Yu, H. (2014). A brain-inspired spiking neural network model with temporal encoding and learning. *Neurocomputing, 138*, 3–13. https://doi.org/10.1016/j.neucom.2013.06.052

Zhan, Q., Liu, G., Xie, X., Zhang, M., & Sun, G. (2023). Bio-inspired Active Learning method in spiking neural network. Knowledge-Based Systems *261*, C. https://doi.org/10.1016/j.knosys.2022.110193

Ziv, J., & Lempel, A. (1976). On the complexity of finite sequences. *IEEE Transactions on information theory, 22*, 75–81. https://doi.org/10.1109/TIT.1976.1055501

Zong, J., Wang, J., Wang, J., Li, G., & Wu, R. (2026). Polaris 23: A high throughput neuromorphic processing element by RISC-V customized instruction extension for spiking neural network (RV-SNN 2.0) and SIMD-style implementation of LIF model with backpropagation STDP. *The Journal of Supercomputing, 81*, 398. https://doi.org/10.1007/s11227-024-06826-y