



Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

journal homepage: www.elsevier.com/locate/chaos



Understanding the training dynamics of CoLaNET by its simplified model

O.A. Goryunov^a, O.V. Maslennikov^a, M.V. Kiselev^b, Igor Franović^{c,*},
V.V. Klinshov^{a,d,*}

^a A.V. Gaponov-Grekhov Institute of Applied Physics of the Russian Academy of Sciences, 46 Ulyanova Street, 603950 Nizhny Novgorod, Russia

^b Chuvash State University, 15 Moskovsky Avenue, 428015 Cheboksary, Russia

^c Scientific Computing Laboratory, Center for the Study of Complex Systems, Institute of Physics Belgrade, University of Belgrade, Pregrevica 118, 11080 Belgrade, Serbia

^d HSE University, 25/12 Bolshaya Pecherskaya street, 603155 Nizhny Novgorod, Russia

ARTICLE INFO

Keywords:

Spiking Neural Networks (SNNs)

Learning

Synaptic plasticity

Reduction approach

ABSTRACT

Training complex, biologically plausible Spiking Neural Networks (SNNs) with local learning rules is a significant challenge for theoretical analysis. Here we address this problem by developing a comprehensive analytical theory for the learning dynamics of CoLaNET, a recently proposed columnar SNN. In particular, we consider a simplified model that captures the core algorithmic logic of CoLaNET's training process. We derive closed-form expressions that accurately predict the number of emergent microcolumns, the complete temporal evolution of synaptic weights, and the model's learning curve. Our theoretical predictions show excellent quantitative agreement with numerical simulations of the simplified model and successfully capture the qualitative behavior of the full spiking CoLaNET. Furthermore, our analysis reveals that the spike-based, asynchronous competition in the full model delays neuronal specialization and slows down the training process relative to its simplified counterpart. This work provides a scalable theoretical foundation for understanding and configuring biologically inspired SNNs and highlights a key difference introduced by spike-based dynamics.

1. Introduction

Spiking Neural Networks (SNNs) are a class of brain-inspired computational models that process information using discrete, asynchronous events, or spikes [1,2]. This event-driven paradigm is fundamentally different from the continuous-valued activations of traditional Artificial Neural Networks (ANNs) and offers significant advantages, most notably in energy efficiency. By activating computational units only when necessary, SNNs are exceptionally well-suited for implementation in low-power neuromorphic hardware, paving the way for next-generation edge AI and autonomous systems [3,4]. The inherent ability of SNNs to encode information in the precise timing of spikes also makes them a strong candidate for processing complex temporal data [5]. This computational power arises from complex, biologically-plausible dynamics, where intrinsic mechanisms like autaptic spiking can regulate neural reactivity and activation patterns [6,7]. Furthermore, principles of synchronization and network topology are crucial, as the specific organization of connections profoundly influences information processing [8–10]. It has been demonstrated that SNNs can use intricately organized spike sequences to generate even simple low-dimensional signals, highlighting the rich representational capacity of these systems [11].

* Corresponding authors.

E-mail addresses: mkiselev@chuvsu.ru (M.V. Kiselev), franovic@ipb.ac.rs (I. Franović), vladimir.klinshov@gmail.com (V.V. Klinshov).

<https://doi.org/10.1016/j.chaos.2025.117663>

Received 4 November 2025; Received in revised form 20 November 2025; Accepted 23 November 2025

Available online 28 November 2025

0960-0779/© 2025 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

Despite such promise, the practical application of SNNs has been hindered by the challenge of developing effective training algorithms [12]. The discontinuous, non-differentiable nature of the spike generation event prevents the direct application of backpropagation, the cornerstone of modern deep learning. To overcome this, two main approaches have emerged. The first involves adapting backpropagation for SNNs through techniques like surrogate gradients, which approximate the derivative of the spiking threshold function [13,14]. This approach has led to state-of-the-art performance on complex benchmarks by enabling the training of deep SNN architectures [15]. However, these methods often require significant computational resources for training and can be complex to implement on-chip, potentially deviating from the goal of fully event-driven, local learning.

The second approach remains closer to the biological roots of SNNs, focusing on local, biologically plausible learning rules. These rules, such as Spike-Timing-Dependent Plasticity (STDP), modify synaptic weights based only on the activity of pre- and post-synaptic neurons [16–19]. Such local mechanisms are ideal for online, on-chip learning in neuromorphic systems [20,21]. To perform supervised tasks, STDP is often combined with other mechanisms, such as reward modulation (R-STDP), where a global dopamine-like signal guides plasticity towards a desired outcome [22], or structural plasticity, where the network's connectivity dynamically adapts during training [23–26]. While these methods warrant biological plausibility and hardware compatibility, designing architectures that effectively solve complex tasks using only local rules remains an active and challenging area of research [27–30].

A recent example of the paradigm based on local rules is CoLaNET (Columnar Layered Network), which has already been proven useful for classification tasks [31]. CoLaNET features a bio-inspired columnar organization where each class is represented by a dedicated column. Within each column, a certain number of “microcolumns” are specified to accommodate for different variations of the features of this class. Learning is governed entirely by local plasticity rules based on dopamine reinforcement and anti-Hebbian mechanisms. While this design makes CoLaNET highly suitable for neuromorphic implementation, its intricate dynamics, involving multiple specialized neuron types and feedback loops, makes it difficult for formal theoretical analysis. Understanding how and why the learning process converges, and how network parameters influence the final learned representation, is crucial for its principled application and improvement.

To address the latter problem, we analyze a simplified analogue of CoLaNET, first proposed in [32]. This model deliberately abstracts the fine-grained temporal dynamics of individual spikes to concentrate on the fundamental algorithmic logic of the CoLaNET learning process. By treating spike counts as integer inputs, we can construct an analytically tractable framework to study the core mechanics: the dynamic creation of microcolumns, the competitive selection of “winner” neurons, and the reward-based update of synaptic weights. This reduction approach allows us to dissect the learning dynamics and derive insights that would be obscured in a full, complex SNN simulation.

The primary contribution of this work is the development of a comprehensive analytical theory that describes the training process of the simplified CoLaNET model in a canonical binary classification task. Using a standard rate-based encoding with Gaussian receptive fields [33], we derive mathematical expressions that accurately predict the emergent properties and performance of the network. In particular, we provide a theoretical estimate for the number of microcolumns that emerge to represent the target class, linking this number to the properties of the input data and the encoding scheme. We also derive the complete temporal evolution of the synaptic weights, capturing the initial linear growth phase and the subsequent saturation as the network becomes proficient. Finally, we formulate an equation for the model's learning curve, predicting its classification accuracy as a function of the number of training iterations.

The theoretical predictions are validated against numerical experiments, demonstrating excellent quantitative agreement. This approach not only allows for a deep understanding of the CoLaNET learning algorithm but also provides a methodological paradigm of how simplified models can be used to develop a principled, theoretical foundation for complex, bio-inspired SNN architectures.

The paper is organized as follows. Section 2 briefly outlines the full CoLaNET architecture. Section 3 introduces the simplified model and its training algorithm. The classification task is defined in Section 4. Section 5 presents the numerical simulation results, which are then theoretically explained in Section 6. In Section 7 we compare our results obtained for the simplified model with those for the full CoLaNET architecture. Finally, Section 8 discusses the implications of our results and outlines problems for future study.

2. CoLaNET - brief description

The CoLaNET model is described in detail in [31], and here we give a brief overview. CoLaNET consists of several identically organized groups called columns, see Fig. 1. In the context of classification task, each column corresponds to a single class. Each column has a layered structure with five layers, containing different types of neurons called *L*, *WTA*, *R*, *BIAS* and *OUT*. Note that the two latter types (*BIAS* and *OUT*) are present in the column in one copy, while the three former types (*L*, *WTA* and *R*) form triplets, the so-called microcolumns, which may exist in several copies. Different microcolumns are intended to react on substantially different representatives of the same class.

All neurons in CoLaNET are spiking LIF neurons [34] governed by:

$$\frac{dU}{dt} = -\frac{U}{\tau} + \sum_{i,j} w_i \delta(t - t_{ij}), \quad (1)$$

where U is the membrane voltage, τ is the membrane time constant, w_i is the weight of the i th synapse and t_{ij} are moments of spikes arriving to the i th synapse. When the membrane potential reaches the threshold Θ , the neuron generates a spike, and its membrane potential resets to V_{res} . Note that the threshold Θ may be variable.

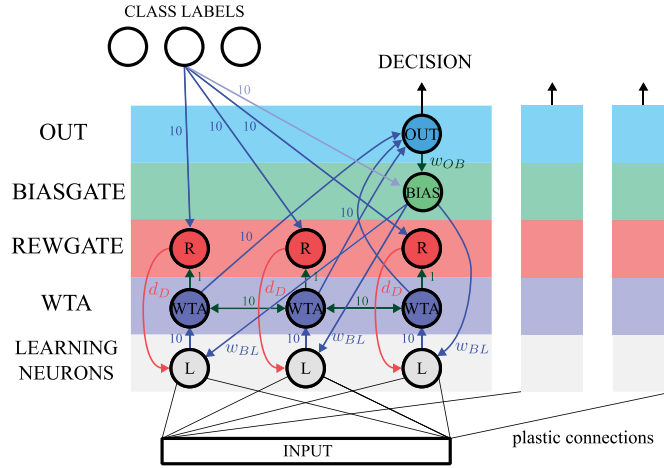


Fig. 1. Architecture of CoLaNET. The column in the left part of the figure is plotted in detail, the other two columns have the same composition. Each column is responsible for its own target class and contains several microcolumns ($L - WTA - R$ triplets), which allows recognizing very different examples of the same class.

In CoLaNET most of the synapses have fixed weights and serve an auxiliary function, and only the synapses on L neurons are plastic and change during the training. An important feature is that the plasticity rules are applied not to the weights w of the synapses directly, but to the so-called synaptic resources W related to the weights via:

$$w = w_{\min} + \frac{(w_{\max} - w_{\min}) \max(W, 0)}{w_{\max} - w_{\min} + \max(W, 0)}, \quad (2)$$

where $w_{\min} < 0$ and $w_{\max} > 0$ are the minimal and the maximal possible weights. The nonlinear part in (2) allows to keep the synaptic weights limited.

The input patterns are encoded via rate code, where each input neuron generates a Poisson spike train with the frequency f_i . In the inference regime, the dynamics of the network is straightforward: patterns belonging to a certain target class should activate a corresponding L neuron, which in turn activates a WTA neuron and an OUT neuron, whereby the latter indicates the decision of the network.

The dynamics of the training is much more sophisticated, and here we provide just an overview, while all the details can be found in [31]. The training examples (input patterns) are presented to the network together with their class labels encoded as strong activity of the corresponding *CLASS LABEL* neuron. The latter one activates the *BIASGATE* neuron after some delay. Importantly, L neurons can be activated either by the input spikes or by the *BIASGATE* spikes. The L neuron that fires earlier than the others is called “a winner”; it receives the so-called dopamine spike from the R neuron, and its plastic synapses change according to the rule of so-called dopamine plasticity: those synapses that have received spikes shortly before are potentiated. This mechanism prompts the L neuron to react on the presented stimulus and similar ones, and when they are presented next time, this neuron will have high chances to become a winner again and to get the same set of synapses potentiated. Once the synapses after some time become sufficiently strong, the L neuron becomes trained and acquires the ability to react on proper input stimuli without the additional input from the *CLASS LABEL* (though the *BIASGATE*). At this moment the potentiation of its synapses is terminated because the dopamine plasticity is compensated by the so-called anti-Hebbian plasticity which depresses all the plastic synapses having received a spike shortly before. Thus, when the L neuron becomes trained and reacts on the input patterns properly, its synaptic weights stabilize. Moreover, if the neuron reacts by mistake, the corresponding synaptic weights are suppressed by the anti-Hebbian plasticity thereby preventing further errors.

Concerning the above description, it is necessary to add a remark about two mechanisms which maintain the specialization of L neurons. The first mechanism is the competition between plastic synapses of a single neuron: whenever some synapses are potentiated, all the other ones are uniformly depressed so that the total synaptic resource is preserved. The second mechanism is the threshold variation: the neuron threshold increases proportionally to the sum of positive weights of the neuron’s plastic synapses, which makes trained neurons less excitable. Together, these two mechanisms ensure that different L neurons in one column react on significantly different instances of the target class.

3. A simplified model of the CoLaNET

The structure and the dynamics of CoLaNET are quite sophisticated and include many components whose diversity complicates the study of the learning process. To focus on the most important aspects of the learning dynamics, a simplified algorithm was suggested in [32] which simulates the learning process in CoLaNET. Let us briefly describe this algorithm for the binary classification task corresponding to a CoLaNET with a single column. The simplified model also includes a single column with several

microcolumns, and each microcolumn consists of a single L neuron and nothing else. Moreover, the number of the microcolumns N can increase during the learning process by adding the new ones (microcolumns not yet added correspond to those not yet activated in CoLaNET).

The input is received through n input neurons and represents a set of nonnegative integer numbers m_i , $i = 1, \dots, n$, corresponding to the number of spikes delivered through each neuron. It is assumed that m_i are small and most of them (but not all) equal zero. Note that the timing of the spikes is not considered. By using spike counts over a time interval as input and disregarding the precise spike timing, the simplified model effectively acts as a rate-based abstraction of the full spiking network. This simplification is crucial, as it transforms the complex temporal dynamics into a discrete, algorithmic process that is amenable to formal mathematical analysis, while still preserving the core learning mechanisms of competition and reward-based updates.

The state variables of the simplified model comprise the $N \times n$ matrix of resources W_{ij} , and the $N \times n$ matrix of weights w_{ij} calculated through the corresponding resources via Eq. (2). The algorithm has several hyperparameters: the minimum (maximum) synaptic weight $w_{\min}$ ($w_{\max}$), and the learning rate d (corresponding to both anti-Hebbian and dopamine plasticity in CoLaNET). Note that the model in [32] contains additional parameters n_s and α , as well as the time-dependent threshold potentials h_a , which are omitted in the present study.

The output of L neurons is calculated as

$$U_j = \sum_{i=1}^n w_{ji} m_i. \quad (3)$$

The neuron is said to fire if its membrane potential U_j exceeds the threshold $\Theta = 1$. All the L neurons project to a single OUT neuron, and the latter fires if at least one of the L neurons fires. In the trained state, the OUT neuron should fire if input representations belong to the target class and should not spike otherwise. Let us describe how the model is trained to perform this binary classification task by supervised learning.

In the beginning of the learning process, $N = 0$. For each teaching example $\vec{m} = (m_1, \dots, m_n)$ the output of each L neuron is calculated via (3), and further actions depend on whether the example belongs to the target class as follows.

1. If $\vec{m}$ is from the target class, the algorithm checks whether there are L neurons which have reacted to the input positively ($U_j > 0$).

1.1. If there are no such L neurons (or it is the first step and $N = 0$), a new L neuron is added and N increases by one. After adding, the resources of all the synapses are set to

$$W_0 = -w_{\min} \left(1 + \frac{-w_{\min}}{w_{\max}} \right), \quad (4)$$

so that the synaptic weights are zero. Then the newborn N th neuron is rewarded is immediately assigned with

$$W_{ji} \leftarrow \begin{cases} W_{ji} + d, & m_i > 0 \\ W_{ji} - d \frac{n_p}{n_0}, & m_i = 0, \end{cases} \quad (5)$$

where $n_p = |\{i : m_i > 0\}|$ is the number of positive elements in $\vec{m}$, while $n_0 = |\{i : m_i = 0\}|$ is the number of zero elements. The positive update of synapses corresponding to non-zero components of $\vec{m}$ prompts the neuron to react to similar inputs, while the negative update of silent synapses ensures the synaptic balance so that the total resource of all neuron's synapses remains constant.

1.2. If there are already L neurons which react on the input positively with $U_j > 0$, the algorithm chooses the neuron k with maximal U_k . Further actions depend on whether the reaction is sub- or super-threshold.

1.2.1. If the reaction is subthreshold ($U_k < \Theta$), the k th neuron is rewarded according to (5) in order to facilitate this reaction.

1.2.2. If the reaction is already superthreshold and the neuron fires ($U_k \geq \Theta$), the k th neuron is not updated. However, if there are other L neurons reacting to the same input with $U_j > 0$ and $j \neq k$, they are punished according to

$$W_{ji} \leftarrow \begin{cases} W_{ji} - d, & m_i > 0 \\ W_{ji} + d \frac{n_p}{n_0}, & m_i = 0. \end{cases} \quad (6)$$

This is done in order to ensure the specialization of neurons so that several L neurons do not react to the same input.

2. If $\vec{m}$ is not in the target class, the algorithm checks whether some L neurons fired (which leads to the incorrect answer of the model) and punishes them according to (6).

The described algorithm is illustrated in Fig. 2. As discussed in [32], the training of this simplified model resembles the training of CoLaNET except for a few minor details. At the same time, the dynamics of the simplified model is more amenable to theoretical analysis and numerical simulation.

4. One-dimensional data classification task

We proceed by analyzing the training of the simplified model on a toy problem of binary classification of one-dimensional data. We are given numbers $x \in [0, 1)$ (for convenience, the points $x = 0$ and $x = 1$ are treated as identical) which are encoded using Gaussian receptive fields [33]. The whole interval $[0, 1)$ is covered equidistantly by the receptive fields of n input neurons

$$f_i(x) = f_{\max} \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right), \quad (7)$$

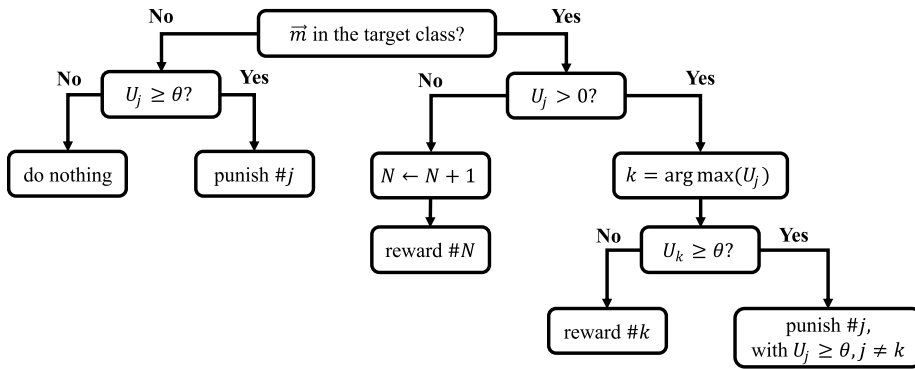


Fig. 2. Block diagram illustrating the logic of the simplified model. This simplified algorithm simulates the operation of two main learning mechanisms of CoLaNET: on one branch - an analogue of dopamine plasticity, on the other - the rule of anti-Hebbian plasticity.

where the receptive field of the i th neuron is centered at $x_i = (i - 0.5)/n$ and has a half-width σ . Each neuron generates spikes according to the Poisson process with the frequency $f_i(x)$ during a certain time interval T_p , so that the probability of generating m_i spikes is

$$p[m_i] = \frac{(f_i T_p)^{m_i}}{m_i!} \cdot e^{-f_i T_p}. \quad (8)$$

The components m_i of the encoding vector are the numbers of spikes generated by each neuron.

The model has to be trained to perform an in-range check, i.e., to react to inputs $x \in [a, b]$ and not react to inputs $x \notin [a, b]$, where $0 < a < b < 1$ are given numbers.

5. Learning dynamics - numerical results

Let us now assume that the training examples x_k are drawn randomly so that the probability of a positive example equals 0.5, and the positive examples are drawn from a uniform distribution $x \in [a, b]$. A typical learning process is illustrated in Fig. 3. Fig. 3(a) shows the birth of L neurons and evolution of their receptive fields in time. Here, by the receptive field of the L neuron we mean the set of input neurons most strongly connected to it. One finds that the L neurons are born in the first several steps, and their receptive fields do not change significantly for most of the neurons, although they may shrink (as for neuron 2) or expand (as for neuron 1) for some of them. However, the strength of the connections grows persistently for all the neurons, as can be seen from subsequent snapshots of the synaptic weights profiles in Figs. 3(b–d). The shape of all the profiles remains bell-like, while the amplitude grows consistently (except for neuron 2 whose synapses no longer gain weight after the first few steps). The dynamics of the receptive fields is further illustrated in Figs. 3(e–f) which show the dynamics of their width and strength, respectively. While the width quickly stabilizes and remains (almost) constant, the strength (the maximal synaptic weight) grows persistently.

6. Theory

In this section we develop the theory which allows to describe qualitatively and quantitatively the dynamics of learning and to estimate the performance of the simplified model from Section 3. Let us first estimate the number of active neurons with $m_i > 0$ for a given input x . The expected number of spikes generated by the i th neuron equals $f_i(x)T_p$, and the number of neurons which are expected to generate at least one spike can be found from (7) as $2\sigma\sqrt{2\ln f_{\max}T_p}$. For convenience, input neurons can be characterized by their preferred coordinates x_i rather than indexes i . Then the length of the interval activated by a certain input x equals

$$l = 2\sigma\sqrt{2\ln f_{\max}T_p}. \quad (9)$$

In the very beginning, the process of learning proceeds as follows. The first positive training example x_k activates the input neurons on the interval of length $\approx l$, and they get into the receptive field of the newborn L neuron. The next positive example activates another interval of length $\approx l$, and if this interval intersects with the receptive field of the first L neuron, the activated neurons join this receptive field. Otherwise, a new L neuron is added, and the activated interval forms its receptive field. The process repeats until all the input neurons get into receptive field of some L neuron. Obviously, the maximal number of L neurons is created if the subsets activated by first positive training examples do not intersect, and it equals $(b - a)/l$. The opposite case is when all subsets activated by first positive training examples intersect (for instance, the positive examples x_k move gradually and slowly from a to b), and in this case only one L neuron is created. We can estimate the expected number of created L neurons as the average of these two limiting cases, namely as

$$N_0 \approx \frac{b - a}{2l}. \quad (10)$$

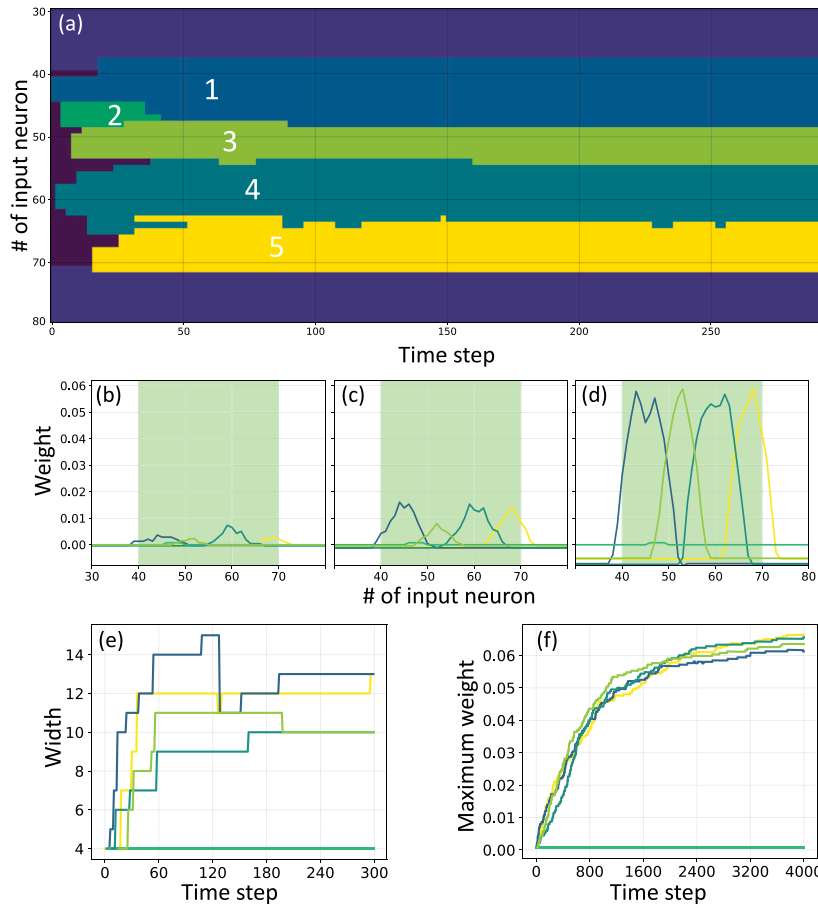


Fig. 3. Typical learning process in the simplified model. (a) Receptive fields of L neurons. (b–d) Snapshots of the distributions of synaptic weights of L neurons (have a trapezoidal shape). (e) Time traces of the widths of the receptive fields. As can be seen, the widths of the receptive fields quickly stabilize and do not change. (f) Time traces of the maximal weights of each L neuron. In all panels, different colors correspond to different neurons.

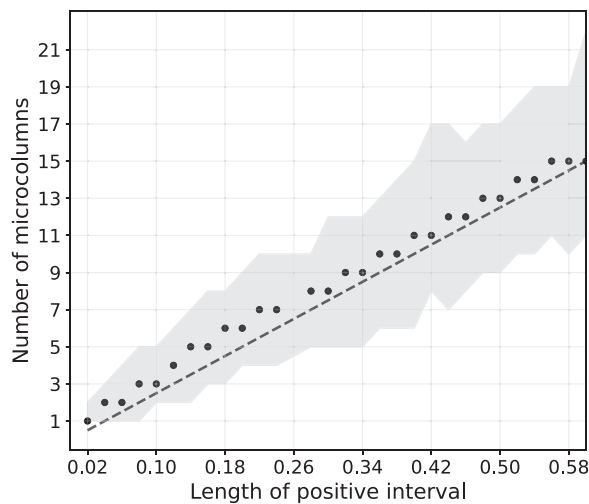


Fig. 4. The most probable number of generated L neurons versus the length of the interval $[a; b]$. Black dots: numerical results, dashed line: theoretical prediction. Gray area: the minimal and the maximal number of L neurons observed in all experiments. Parameters values: $d = 0.003$, $w_{max} = -w_{min} = 0.3$, $N = 200$, $\sigma = 1/N$, $f_{max}T_p = 10$.

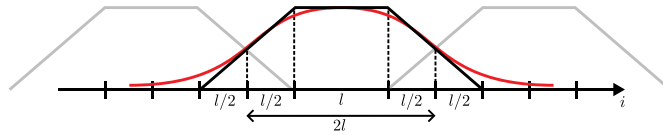


Fig. 5. Trapezoid shape of the weight distribution and its approximation by a Gaussian.

This simple estimate turns out to be surprisingly accurate which is demonstrated by varying the length of the positive interval $[a; b]$. For each length of the interval, we performed the experiment 1000 times and saved the number of created L neurons. The minimal, the maximal and the most probable number of L neurons are plotted in Fig. 4. The estimate (10) happens to be very close to the most probable number, and the actual number for each experiment is always of the same order.

Now let us study the dynamics of training, i.e. evolution of synaptic weights in time. Recall that the profiles of weights for all L -neurons are bell-like with intersecting tales. Let us first estimate the shape of these profiles. Taking into account (10), assume that all the receptive fields are of the width $2l$. As long as the synapses are weak and cannot make L neurons fire, some L neuron is rewarded after each positive training example. The synapse from the input neuron with the preferable coordinate x_i to some L neuron is rewarded if the training example x_k gets into the receptive field of this very L neuron, and $|x_i - x_k| \leq l$. Due to this, the synapses in the centers (near the edges) of the receptive field are rewarded more (less) often. Provided that the relation between the weight and resource is close to linear, this leads to the weight profiles of trapezoid shape with the upper base of length l and the lower base of the length $3l$, as illustrated in Fig. 5.

Let us denote the weight at the top of the trapezoid as w_c and consider the weight outside the trapezoid as zero (in reality it will be slightly negative due to synaptic balance, but we currently neglect this). Then we can calculate the speed of growth of the trapezoid's height. The probability to get the training example within the interval with $|x_i - x_k| \leq l$ equals $l/(b-a)$, so the synaptic resource in the center of the receptive field grows with

$$dW_c/dt \approx \frac{dl}{b-a}. \quad (11)$$

This linear growth continues until the synapses strengthen enough so that L neurons acquire capability to fire in response to some inputs x_k . In order to understand when this happens, one needs to calculate the voltage of j th L neuron after the presentation of input x_k . To do so, we replace the sum in (3) by an integral:

$$\begin{aligned} U_j &= \sum_{i=1}^n w_{ji} m_i \approx \sum_{i=1}^n w_{ji} T_p f_i(x_k) \approx \\ &\approx T_p f_{\max} \int_{-\infty}^{\infty} w_j(x) \exp\left(-\frac{(x-x_k)^2}{2\sigma^2}\right) n dx, \end{aligned} \quad (12)$$

where $w_j(x)$ is the weight profile of the j th L -neuron. To evaluate the integral, we approximate this trapezoid profile by a Gaussian function of the same height and integral:

$$w_j(x) \approx w_c \exp\left(-\frac{(x-x_j)^2}{2\chi^2}\right), \quad (13)$$

where x_j is the center of the receptive field and $\chi = l\sqrt{\frac{2}{\pi}}$ (see Fig. 5). Then the integral (12) equals

$$U_j = T_p f_{\max} n w_c \sqrt{\frac{2\pi\sigma^2\chi^2}{\sigma^2 + \chi^2}} \exp\left(-\frac{(x_k - x_j)^2}{2(\sigma^2 + \chi^2)}\right). \quad (14)$$

Obviously, U_j is maximal for $x_j = x_k$, and it reaches the threshold θ for

$$w_0 = \frac{\theta}{T_p f_{\max} n} \sqrt{\frac{\sigma^2 + \chi^2}{2\pi\sigma^2\chi^2}}. \quad (15)$$

Then, taking into account the relation between the weight and the resource (2) and given the speed of the resource growth (11), one can find the time when L neurons start to fire:

$$t_0 = \frac{w_0}{d} \frac{(b-a)}{l} \frac{(w_{\max} - w_{\min})^2}{w_{\max}(w_{\max} - w_0)}. \quad (16)$$

This time is proportional to the required weight w_0 and the task complexity $(b-a)$, and inversely proportional to the learning rate d . After this time L neurons start to fire only if the inputs get in the very center of their receptive field. When the weights grow further, the interval of inputs which make the neuron fire expands. Note, however, that the speed of the resource growth decreases

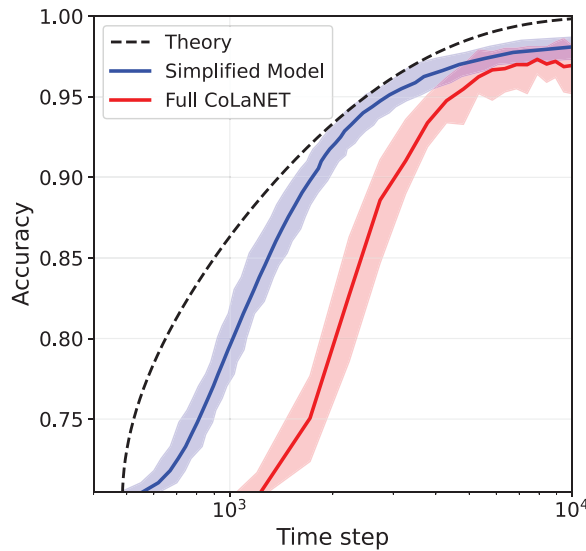


Fig. 6. Learning curves for the simplified model (blue curve), full CoLaNET model (red curve), and theoretical prediction (dashed black curve). Solid curves represent the results averaged over 100 experiments, while shading indicates the entire set of observed values. The theoretical curve fits well with the simplified model, and the qualitative behavior is preserved for the full model, although the latter learns more slowly due to spike competition. Parameters values for simplified model: $d = 0.003$, $w_{\max} = -w_{\min} = 0.3$, $N = 100$, $\sigma = 1/N$, $f_{\max} T_p = 10$. The parameters for the CoLaNET simulation were fixed taking into account the scaling factor β according to Eq. (25).

since the inputs which make the L neuron fire do not lead to potentiation of its synapses. For given w_l , the width of the interval of inputs which make the L neuron fire equals

$$l_f(w_l) = 2\sqrt{2(\sigma^2 + \chi^2) \ln \frac{w_l}{w_{l0}}}, \quad (17)$$

so the resource grows as

$$dW/dt \approx \frac{dl}{b-a} \left(1 - \frac{l_f}{2l}\right). \quad (18)$$

According to this ODE, the weight grows until it reaches the value

$$w_1 = w_0 \exp\left(\frac{l^2}{2(\sigma^2 + \chi^2)}\right). \quad (19)$$

To describe the temporal dynamics of its growth, we linearize Eq. (18) near the equilibrium point. Given the initial conditions $w_c(t_0) = w_0$ we finally obtain for $t > t_0$

$$W(t) = W_1 + (W_0 - W_1) \exp(-(t - t_0)/t_1), \quad (20)$$

where

$$W_0 = \frac{(w_{\max} - w_{\min})(w_0 - w_{\min})}{(w_{\max} - w_0)}, \quad (21)$$

$$W_1 = \frac{(w_{\max} - w_{\min})(w_1 - w_{\min})}{(w_{\max} - w_1)}, \quad (22)$$

$$t_1 = \frac{w_1}{d} \frac{(b-a)}{l} \frac{l^2}{\sigma^2 + \chi^2} \frac{(w_{\max} - w_{\min})^2}{(w_{\max} - w_1)^2}. \quad (23)$$

The dynamics of the resource allows to understand how the performance of the model grows during the training progress. Assuming that the system does not make false positive decisions, its accuracy can be found as

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} = 1 - (b-a) \left(1 - \frac{l_f}{2l}\right), \quad (24)$$

where tp , tn , fp and fn are the numbers of true positive, true negative, false positive and false negative replies, and l_f is given by (17). The learning curve obtained from our analysis is plotted in Fig. 6 along with the results of 100 experiments with the simplified model from Section 3. Theoretical estimates are in good agreement with the numerical results.

To check the generality of our results, we performed a series of experiments with different tasks varying the length of the positive interval $[a; b]$. For each interval length, we have trained the model until it reached accuracy of 90% and have saved the time it takes

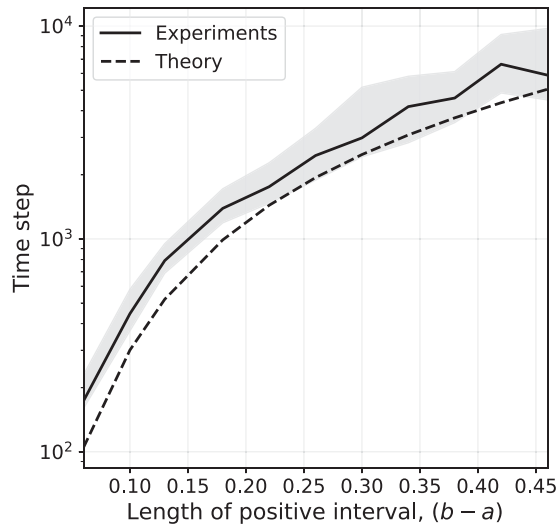


Fig. 7. Training time versus the length of the interval $[a, b]$. Solid line: numerical results with a simplified model, dashed line: theoretical results. Parameters values: $d = 0.003$, $w_{\max} = -w_{\min} = 0.3$, $N = 100$, $\sigma = 1/N$, $f_{\max} T_p = 10$.

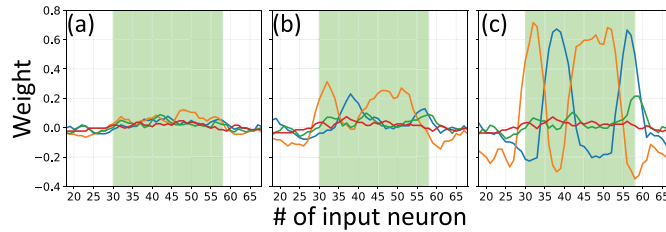


Fig. 8. Weights distributions for the full CoLaNET model at (a) $t = 500$, (b) $t = 1000$, and (c) $t = 2500$. Slow specialization in the full model contrasts with rapid stabilization in the simplified model and is explained by the stochastic nature of spike competition.

to reach such the performance. The results of the experiments demonstrate remarkable agreement with theoretical predictions, see Fig. 7.

7. Comparison with the full CoLaNET learning dynamics

Having verified the derived theory for the training of the simplified model, we examine to what extent it is applicable to the full CoLaNET network. The simplified model should reproduce the key features of the full one, therefore one expects that the theory also applies to the latter one as well. In order to test this, we performed a series of experiments with the full CoLaNET model training it for the very same task described in Section 4. Note that the hyperparameters of the full model were not exactly the same as in the simplified model, but were rather tuned in to ensure the best match between them. In particular, in the simplified model m incoming spikes make a neuron fire if and only if the synaptic weight $w \geq \Theta/m$. In the full model, the timing of the spikes matters. The spikes arrive to the L neurons during the time interval T_p . If their number $m \gg 1$ and they arrive asynchronously, the neuron will fire if and only if the synaptic weight $w \geq \beta\Theta/m$, where

$$\beta = \frac{T_p}{\tau(1 - \exp(-T_p/\tau))}. \quad (25)$$

This scaling factor β effectively bridges the gap between the *rate-based* logic of the simplified model and the *temporal integration dynamics* of the full spiking model. Taking this into account, we rescaled the parameters of the full CoLaNET model ($w_{\min}$, $w_{\max}$, and the plasticity rates) by the factor β .

The results of the experiments with the full CoLaNET network are shown in Fig. 6 together with the theoretical results and the results for the simplified model. The qualitative correspondence between all three learning curves is obvious which corroborates the applicability of the developed theory to the description of the full model. However, the full model trains noticeably slower than the simplified one, which prompted us to investigate out the reasons for this discrepancy.

To gain insight into this problem, we plotted the distributions of the synaptic weights in the subsequent moments of time in Fig. 8. The comparison of these distributions with those for the simplified model (Fig. 3) reveals the key difference in the training

process between the two models. In the simplified model, the receptive fields of L neurons are narrow and uni-modal from the very beginning, while in the full model they are narrow and jagged-shape for quite long time (up to $t = 500$). Only as late as at $t = 1000$ the weight distributions acquire pronounced peaks, and even then there is a striking difference with the simplified model in that the distributions of the full model may be multimodal, i.e. receptive fields of an L neuron may include several separated intervals of the input space.

The reason of the observed difference in the training process of the simplified and the full model is the spiking nature of the latter one. In particular, in the simplified model each L neuron is specialized on a particular subclass of inputs. When a new input is presented, the neuron specialized on similar inputs gets the most excitation U_j , and due to this advantage this very neuron is rewarded, no matter how small the advantage is. In the full model, a small advantage may not be sufficient to guarantee that the neuron spikes earlier than other L neurons, and if it spikes simultaneously with other neurons, the one to be rewarded is selected randomly. Thus, neurons only begin to specialize in the later stages of learning, once their synaptic weights have become strong enough to establish a significant advantage. This effect slows down the training process and decreases the number of involved microcolumns.

8. Discussion

In this study, we have developed and validated an analytical theory for the learning dynamics of a simplified model of the CoLaNET architecture. By abstracting the complex spiking dynamics, we were able to derive closed-form expressions for key emergent properties, including the number of microcolumns, the temporal evolution of synaptic weights, and the learning curve. Our theoretical predictions show excellent agreement with numerical simulations of the simplified model and capture the qualitative behavior of the full spiking CoLaNET. Such agreement demonstrates the usefulness of simplified models as a powerful tool for obtaining a principled understanding of complex, bio-inspired SNNs.

The core insights from our theory allow us to extrapolate the model's behavior to a broader class of classification tasks beyond the simple 1D case studied here. The fundamental driving force of the learning process is the competition among L -neurons to partition the input space of the target class. Our analysis identifies two key parameters that govern this process: the size of the activated input interval for a single stimulus, l , and the size of the target class in the input space, here represented by $(b - a)$. Eq. (10) provides a general heuristic for the network's representational capacity. It implies that the number of specialized feature detectors (microcolumns) scales linearly with the "size" of the target class and inversely with the "resolution" of a single neuron's receptive field. For multi-dimensional data, $(b - a)$ would be replaced by the volume of the target manifold, and l would be replaced by the diameter of the activated region in the input space. This suggests that complex classes requiring high representational diversity will naturally lead to the emergence of more microcolumns. Obviously, if the input space is very complex and high-dimensional, the number of microcolumns becomes excessive, and one needs additional preprocessing and feature extraction layers in order to keep the model working.

We have also derived the values of the synaptic weights indicating the beginning (w_0) and the end (w_1) of the network training. The former one corresponds to the moment when the model starts to respond on *some* of the target inputs, while the later one corresponds to the state when the model responds to *all* of them. Expression (15) for w_0 is quite simple: it just states that the L neuron should fire when the input gets *exactly* into the center of its receptive field. Expression (19) is less trivial and, in fact, states that the weight w_1 is larger than w_0 but of the same order (since l , σ and χ are all of the same order). These results provide important indications for the choice of the model hyperparameters: one should calculate w_0 and set $w_{\max}$ an order of magnitude larger, so that the training process occurs beyond the saturation.

Finally, the most interesting result of our theory is the derivation of the learning curve of the simplified model, including the time t_0 when the performance starts to grow and the timescale of the performance saturation t_1 . Again, Eq. (16) for t_0 is simple: it is proportional to the time w_0/d it takes to train one L neuron and the number of L neurons $(b - a)/l$, while the nonlinear dependence on w_0 is marginal since $w_0 \ll w_{\max}$. Expression (23) for performance saturation t_1 is more sophisticated, indicating that t_1 may get very large if $w_{\max}$ is set too close to w_1 . This observation would suggest making $w_{\max}$ very large which is, however, undesirable since it also makes the model unstable to noise. Thus, the recommended solution is to set $w_{\max}$ several times larger than w_1 .

Our comparison of the simplified and the full CoLaNET indicates the slower training speed and delayed specialization of neurons in the latter one. As shown in Section 7, this is a direct consequence of the spike-based, asynchronous winner-take-all mechanism. In the simplified model, the neuron with the highest membrane potential is always deterministically selected for reward. In contrast, in the spiking network, a neuron must not only have the highest potential but also exceed its threshold and fire before its competitors. Small advantages in synaptic strength may not translate into a decisive temporal advantage in spike timing, especially in the early stages of learning. This leads to a period of non-specialization where the "winning" neuron is selected almost randomly, delaying the stable partitioning of the input space. This effect is a fundamental property of spike-based competition and represents a key difference from rate-based or analog models. It suggests that the performance of biologically plausible SNNs could be improved by mechanisms that amplify small differences in excitation to ensure a clear and timely winner, especially during the initial phase of learning.

It is important to acknowledge that our theoretical framework was developed and validated using a canonical one-dimensional classification task. While this provides a clear and tractable foundation, a crucial avenue for future research is to extend this theory to high-dimensional and structurally complex data, such as real-world image datasets (e.g., MNIST). This extension is non-trivial and requires a careful generalization of the core concepts. The "size" of the target class, represented by the interval length $(b - a)$ in our 1D case, would have to be reinterpreted as a measure of the class manifold's volume in the high-dimensional input space. Similarly, the

“resolution” of a single stimulus, I , would correspond to the volume or diameter of the region of input neurons activated by a single sample. Successful generalization of our theoretical predictions, especially the estimate for the number of emergent microcolumns (10), to this multi-dimensional context would provide a powerful tool for predicting the required network capacity for complex tasks and would be a significant step towards a general theory of learning in such bio-inspired architectures.

In summary, our theoretical framework provides a scalable and generalizable understanding of CoLaNET’s learning principles. The analytical expressions for characteristics of the training process offer a predictive framework for configuring the network for various classification tasks, estimating its required capacity, and anticipating its learning timeline. The insights into the impact of spike-based dynamics on specialization timing are vital for the future development of not only CoLaNET, but also other biologically plausible SNN architectures that rely on local, competitive plasticity rules. We believe that the future work should focus on extending this analysis to multi-dimensional and hierarchical data structures, and on incorporating the stochastic effects of spike-timing into the theoretical model itself.

CRediT authorship contribution statement

O.A. Goryunov: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **O.V. Maslennikov:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **M.V. Kiselev:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Igor Franović:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Methodology, Conceptualization. **V.V. Klinshov:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing 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.

Acknowledgments

The work on Sections 2, 3 and 7 was supported by the Russian Science Foundation, Russia, grant No. 24-12-10088. The work on Sections 4 and 5 was supported by the Russian Science Foundation, Russia, grant No. 23-72-00245. The work on Section 6 was supported by the project “International academic cooperation” of the HSE University.

Data availability

Data will be made available on request.

References

- [1] Maass W. Networks of spiking neurons: the third generation of neural network models. *Neural Netw* 1997;10:1659. [http://dx.doi.org/10.1016/S0893-6080\(97\)00011-7](http://dx.doi.org/10.1016/S0893-6080(97)00011-7).
- [2] Maslennikov OV, Pugavko MM, Shchapin DS, Nekorkin VI. Nonlinear dynamics and machine learning of recurrent spiking neural networks. *Uspekhi Fiz Nauk* 2022;65:1020. <http://dx.doi.org/10.3367/ufnr.2021.08.039042>.
- [3] Schuman CD, Potok TE, Patton RM, Birdwell JD, Dean ME, Rose GS, Plank JS. A survey of neuromorphic computing and neural networks in hardware. 2017, <http://dx.doi.org/10.48550/arXiv.1705.06963>.
- [4] Pfeiffer M, Pfeil T. Deep learning with spiking neurons: Opportunities and challenges. *Front Neurosci* 2018;12:774. <http://dx.doi.org/10.3389/fnins.2018.00774>.
- [5] Yamazaki K, Vo-Ho V-K, Bulsara D, Le N. Spiking neural networks and their applications: A review. *Brain Sci* 2022;12:863. <http://dx.doi.org/10.3390/brainsci12070863>.
- [6] Guo D, Wu S, Chen M, Perc M, Zhang Y, Ma J, Cui Y, Xu P, Xia Y, Yao D. Regulation of irregular neuronal firing by autaptic transmission. *Sci Rep* 2016;6:26096. <http://dx.doi.org/10.1016/j.neunet.2019.09.036>.
- [7] Guo D, Chen M, Perc M, Wu S, Xia C, Zhang Y, Xu P, Xia Y, Yao D. Firing regulation of fast-spiking interneurons by autaptic inhibition. *Europhys Lett* 2016;114:30001. <http://dx.doi.org/10.1209/0295-5075/114/30001>.
- [8] Majhi S, Ghosh S, Pal PK, Pal S, Pal TK, Ghosh D, Završnik J, Perc M. Patterns of neuronal synchrony in higher-order networks. *Phys Life Rev* 2025;52:144. <http://dx.doi.org/10.1016/j.plrev.2024.12.013>.
- [9] Gosak M, Milojević M, Duh M, Skok K, Perc M. Networks behind the morphology and structural design of living systems. *Phys Life Rev* 2022;41:1. <http://dx.doi.org/10.1016/j.plrev.2022.03.001>.
- [10] Klinshov VV, Kovalchuk AV, Soloviev IA, Maslennikov OV, Franović I, Perc M. Extending dynamic memory of spiking neuron networks. *Chaos Solitons Fractals* 2024;182:114850. <http://dx.doi.org/10.1016/j.chaos.2024.114850>.
- [11] Maslennikov O, Perc M, Nekorkin V. Topological features of spike trains in recurrent spiking neural networks that are trained to generate spatiotemporal patterns. *Front Comput Neurosci* 2024;18:1363514. <http://dx.doi.org/10.3389/fncom.2024.1363514>.
- [12] Taherkhani A, Belatreche A, Li Y, Cosma G, Maguire LP, McGinnity T. A review of learning in biologically plausible spiking neural networks. *Neural Netw* 2020;122:253. <http://dx.doi.org/10.1016/j.neunet.2019.09.036>.

- [13] Zenke F, Ganguli S. Superspike: Supervised learning in multilayer spiking neural networks. *Neural Comput* 2018;30:151. http://dx.doi.org/10.1162/neco_a.01086.
- [14] Neftci EO, Mostafa H, Zenke F. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process Mag* 2019;36:51. <http://dx.doi.org/10.1109/MSP.2019.2931595>.
- [15] Fang W, Yu Z, Chen Y, Huang T, Masquelier T, Tian Y. Deep residual learning in spiking neural networks. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, editors. *Advances in neural information processing systems*. vol. 34, Curran Associates, Inc.; 2021, p. 21056–69, URL https://proceedings.neurips.cc/paper_files/paper/2021/file/afe434653a898da20044041262b3ac74-Paper.pdf.
- [16] Tavanaei A, Ghodrati M, Kheradpisheh SR, Masquelier T, Maida A. Deep learning in spiking neural networks. *Neural Netw* 2019;111:47. <http://dx.doi.org/10.1016/j.neunet.2018.12.002>.
- [17] Kheradpisheh SR, Ganjtabesh M, Thorpe SJ, Masquelier T. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Netw* 2018;99:56. <http://dx.doi.org/10.1016/j.neunet.2017.12.005>.
- [18] Diehl PU, Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front Comput Neurosci* 2015;9:99. <http://dx.doi.org/10.3389/fncom.2015.00099>.
- [19] Caporale N, Dan Y. Spike timing-dependent plasticity: A hebbian learning rule. *Annu Rev Neurosci* 2008;31:25. <http://dx.doi.org/10.1146/annurev.neuro.31.060407.125639>.
- [20] Querlioz D, Bichler O, Dollfus P, Gamrat C. Immunity to device variations in a spiking neural network with memristive nanodevices. *IEEE Trans Nanotechnol* 2013;12:288. <http://dx.doi.org/10.1109/TNANO.2013.2250995>.
- [21] Wang Z, Joshi S, Savel'ev SE, Jiang H, Midya R, Lin P, Hu M, Ge N, Strachan JP, Li Z, Wu Q, Barnell M, Li G-L, Xin HL, Williams RS, Xia Q, Yang JJ. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nat Mater* 2017;16:101. <http://dx.doi.org/10.1038/nmat4756>.
- [22] Frémaux N, Gerstner W. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front Neural Circuits* 2016;9:85. <http://dx.doi.org/10.3389/fncir.2015.00085>.
- [23] Panda P, Srinivasa N. Learning to recognize actions from limited training examples using a recurrent spiking neural model. *Front Neurosci* 2018;12:126. <http://dx.doi.org/10.3389/fnins.2018.00126>.
- [24] Bennett SH, Kirby AJ, Finnerty GT. Rewiring the connectome: Evidence and effects. *Neurosci Biobehav Rev* 2018;88:51. <http://dx.doi.org/10.1016/j.neubiorev.2018.03.001>.
- [25] Butz M, Wörgötter F, van Ooyen A. Activity-dependent structural plasticity. *Brain Res Rev* 2009;60:287. <http://dx.doi.org/10.1016/j.brainresrev.2008.12.023>.
- [26] Van Ooyen A, Butz-Ostendorf M. *The rewiring brain: a computational approach to structural plasticity in the adult brain*. Academic Press; 2017.
- [27] Yuan Y, Zhu Y, Wang J, Li R, Xu X, Fang T, Huo H, Wan L, Li Q, Liu N, Yang S. Incorporating structural plasticity into self-organization recurrent networks for sequence learning. *Front Neurosci* 2023;17:1224752. <http://dx.doi.org/10.3389/fnins.2023.1224752>.
- [28] Tiddia G, Sergi L, Golosio B. Theoretical framework for learning through structural plasticity. *Phys Rev E* 2024;110:44311. <http://dx.doi.org/10.1103/PhysRevE.110.044311>.
- [29] Knight JC, Senk J, Nowotny T. A flexible framework for structural plasticity in GPU-accelerated sparse spiking neural networks. 2025, [arXiv:2510.19764](https://arxiv.org/abs/2510.19764).
- [30] Feketa P, Meurer T, Kohlstedt H. Structural plasticity driven by task performance leads to criticality signatures in neuromorphic oscillator networks. *Sci Rep* 2022;12:15321. <http://dx.doi.org/10.1038/s41598-022-19386-z>.
- [31] Kiselev M. CoLaNET – a spiking neural network with columnar layered architecture for classification. 2025, [arXiv:2409.01230](https://arxiv.org/abs/2409.01230).
- [32] Kiselev M. Numeric model of spiking neural network CoLaNET learning process. In: 2025 9th scientific school dynamics of complex networks and their applications. IEEE; 2025, p. 56–9. <http://dx.doi.org/10.1109/DCNA67486.2025.11199949>.
- [33] Wang J, Belatreche A, Maguire L, McGinnity TM. An online supervised learning method for spiking neural networks with adaptive structure. *Neurocomputing* 2014;144:526. <http://dx.doi.org/10.1016/j.neucom.2014.04.017>.
- [34] Gerstner W, Kistler WM. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press; 2002.

Update

Journal of Nonlinear Science and Nonequilibrium Thermodynamics: the interdisciplinary journal of Nonlinear Science, and Nonequilibrium Thermodynamics

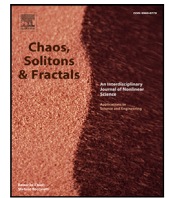
Volume 203, Issue , February 2026, Page

DOI: <https://doi.org/10.1016/j.chaos.2025.117751>



Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

journal homepage: www.elsevier.com/locate/chaos

Corrigendum to “Understanding the training dynamics of CoLaNET by its simplified model” [Chaos Solitons Fractals 203 (2026) 117663]

O.A. Goryunov^a, O.V. Maslennikov^a, M.V. Kiselev^b, Igor Franović^{c,*},
V.V. Klinshov^{a,d,*}

^a A.V. Gaponov-Grekhov Institute of Applied Physics of the Russian Academy of Sciences, 46 Ulyanova Street, Nizhny Novgorod, 603950, Russia

^b Chuvash State University, 15 Moskovsky Avenue, Cheboksary, 428015, Russia

^c Scientific Computing Laboratory, Center for the Study of Complex Systems, Institute of Physics Belgrade, University of Belgrade, Pregrevica 118, Belgrade, 11080, Serbia

^d HSE University, 25/12 Bol'shaya Pecherskaya street, Nizhny Novgorod, 603155, Russia

The Authors would like to apologize for an unfortunate error in stating the project grant numbers in the first two sentences of the Acknowledgement. Instead of writing “The work on sections 2, 3 and 7 was supported by the Russian Science Foundation, grant No. 24-12-10088. The work on sections 4 and 5 was supported by the Russian Science Foundation, grant No. 23-72-00245.” we should have stated “The work on sections 2, 3 and 7 was supported by the Russian Science Foundation, grant No. 23-72-10088. The work on sections 4 and 5 was supported by the Russian Science Foundation, grant No. 24-12-00245.” We regret any inconvenience caused by this error.

DOI of original article: <https://doi.org/10.1016/j.chaos.2025.117663>.

* Corresponding authors.

E-mail addresses: mkiselev@chuvsu.ru (M.V. Kiselev), franovic@ipb.ac.rs (I. Franović), vladimir.klinshov@gmail.com (V.V. Klinshov).

<https://doi.org/10.1016/j.chaos.2025.117751>