International Journal of Neural Systems, Vol. 0, No. 0 (2005) 1–15
© World Scientific Publishing Company

A Layeblack Spiking Neural System for Classification Problems

Gexiang Zhang
School of Control Engineering,
Chengdu University of Information Technology, Chengdu 610225, China
E-mail: zhgxdylan@126.com

Xihai Zhang
School of Electrical and Information Engineering,
Tianjin University, Tianjin, 300072, China
Email: xihaizhang@tju.edu.cn

Haina Rong
School of Electrical Engineering,
Southwest Jiaotong University, Chengdu 610031, China
Email: ronghaina@126.com

Prithwineel Paul, Ming Zhu
School of Control Engineering,
Chengdu University of Information Technology, Chengdu 610225, China
Email: prithwineelpaul@gmail.com; zhuming@cuit.edu.cn

Ferrante Neri*
NICE Group, Department of Computer Science,
University of Surrey, United Kingdom
Email: f.neri@surrey.ac.uk

Yew-Soon Ong
School of Computer Science and Engineering, School of Physical and Mathematical Science,
Nanyang Technological University, Singapore
Email: ASYSOng@ntu.edu.sg

Biological brains have a natural capacity for resolving certain classification tasks. Studies on biologically plausible spiking neurons, architectures and mechanisms of artificial neural systems that closely match biological observations while giving high classification performance are gaining momentum. Spiking neural P systems (SN P systems) are a class of membrane computing models and third-generation neural networks that are based on the behaviour of biological neural cells and have been used in various engineering applications. Furthermore, SN P systems are characterised by a highly flexible structure that enables the design of a machine learning algorithm by mimicking the structure and behaviour of biological cells without the over-simplification present in neural networks. Based on this aspect, this paper proposes a novel type of SN P system, namely, layeblack SN P system (LSN P system), to solve classification problems by supervised learning. The proposed LSN P system consists of a multi-layer network containing multiple weighted fuzzy SN P systems with adaptive weight adjustment rules. The proposed system employs specific ascending dimension techniques and a selection method of output

*Corresponding author.

neurons for classification problems. The experimental results obtained using benchmark datasets from the UCI machine learning repository and MNIST dataset demonstrated the feasibility and effectiveness of the proposed LSN P system. More importantly, the proposed LSN P system presents the first SN P system that demonstrates sufficient performance for use in addressing real-world classification problems.

Keywords: Spiking neural networks, Spiking neural P systems, layeblack weighted fuzzy spiking neural P systems, supervised learning.

1. Introduction

Following the advent of the perceptron algorithm,[1] three generations of artificial neural networks have been developed. With regard to the first-generation, examples Hopfield network.[2] Meanwhile, in terms of the second-generation networks, examples include backpropagation [3,4] and ensemble learning[5] , while third-generation networks include spiking neural networks (SNNs).[6] It should be mentioned that Deep Learning Networks (DLNs), which have multiple hidden layers between input and output, belong to the second generation, as stated in.[7] Furthermore, the authors of Ref[7] report that researchers are currently working at the design of SNNs capable to match the performance of DLNs.

Throughout the history of neural networks, various models have been proposed for voice recognition,[8] autonomous vehicle issues,[9] and brain tumor classification.[10] Since classification problems are generally easy tasks to the human brain, artificial neural network models attempt to mimic the functioning of biological neural networks. However, the simplifications of the synthetic neural models present serious limitations in terms of the capability of artificial intelligence to resolve this type of problem.[11]

For example, the feed value of the first-generation neural networks must be discrete, which limits to Boolean functions only.[12,13] Furthermore, while, unlike first-generation neural networks, second-generation neural networks are able to process continuous values of input and output, they adopt frequency coding that is not suited to certain biological neurons.[12,13] To address this problem, SNNs encode the information by spikes [14,15]. On the other hand, the "integrate-and-fire" type of spiking neurons are used primarily, [6,16] while in neural-biological systems, other types of brain cells, such as astrocytes,[17] exists besides the spiking neurons.

Membrane computing models (generally known as P systems) are computational models first proposed by Gh. Păun,[18] which are formalised on the basis on an inspiration from biological cells. Some examples of modified instances of P systems include tissue-like P systems,[19] cell-like P systems,[18] and neural-like P systems.[20] Among the various models, spiking neural P systems (SN P systems) present an interesting option since they combine the potentials of both the SNN and the P systems, thus generating a model that accurately mimics the biological functions of brains[21]. SN P systems have become increasingly popular due to their similarity to SNNs. However, unlike SNNs, which generally perform specific simplifications, P systems can easily represent different types of neurons and the communication mechanisms among them (see[20]).

Several variants on the SN P systems, with different biological phenomena, have been proposed. Here, examples include the SN P systems with astrocytes,[22] systems with extended channel rules,[23] systems with communication on request,[24,25] systems with polarizations,[26] and coupled neural P systems[27]. These variants have solved many theory and applications problems.[28–30] As well, SN P systems have been successfully applied in various real-world scenarios, including combinatorial optimisation,[31,32] fault diagnosis[33] and arithmetic calculator.[34]

The capabilities of SN P systems for addressing classification problems have also been investigated (see e.g.[35–37]). However, while these pioneering studies indicate the potential of SN P systems for classification, they do not propose classification algorithms that can be used in real life, albeit that the SN P system in[35] is endowed with a dynamic fuzzy reasoning algorithm and learning algorithm, with the theoretical results indicating the viability of the approach in principle. However, the model only analyses the changing weights rule and cannot be used to resolve real-world problems.

Other papers[36,37] have proposed SN P systems that employ an element with the capacity to strengthen and weaken the connections among neurons to address classification problems. The main limitation of this approach is its fixed structure, which requires the pre-determination of the input and output sizes, meaning they can only be applied to specific classification problems. Elsewhere, in,[36, 37] successful applications to the problems pertaining to nuclear export signal identification and binary image recognition have been presented.

In contrast to the existing models[35–37] that are based on a fixed structure or theoretical research, to the best of our knowledge, SN P systems to perform classification ability have not been proposed yet. In the present work, we propose a supervised learning algorithm that makes use of membrane computing theory to achieve classification ability, namely, layeblack weighted fuzzy SN P system (LSN P system). The proposed LSN P system can be flexibly used on an array of classification problems and can process inputs and outputs of any size by maintaining the same network structure.

The proposed LSN P system is composed of multiple instances of a modified implementation of the weighted fuzzy spiking neural P system (WFSN P system)[38] and a novel learning element. We chose to base our classifier on a WFSN P system since it allows for both real-valued and integer input and thus presents a flexible building block to address classification problems. It should be noted that, much like biological brains, WFSN P systems use different types of neurons. The Widrow–Hoff learning law[39] is used to ensure the model learns from data efficiently. Moreover, nonlinear mixed selectivity[40] is used to mix different input features so as to generate higher-dimensional information and to transform non-linearly separable data into linearly separable data.

With this architecture, specific classification problems can be resolved by the LSN P system. In fact, the proposed LSN P system is the first supervised learning algorithm based on a WFSN P system for performing classifications.

The remainder of this paper is organised as follows. In section 2, the WFSN P system is briefly outlined before the proposed LSN P system is described in detail in section 3. The experimental results are then presented in section 4. The conclusions of this work are then presented in section 5.

2. Weighted Fuzzy Spiking Neural P Systems

This section formally defines WFSN P systems and explains how they are the building blocks for LSN P systems.

Definition 1.

A WFSN P system[38] of degree $m(\geq 1)$ is constructed as follows:

$$\Pi = (\{a\}, N_p, N_r, syn, IN, OUT)$$

where

(i) a denotes the spike.

(ii) The set $N_p = \{\sigma_{p1}, \sigma_{p2}, \ldots, \sigma_{pm}\}$ is called *proposition neurons set.* Each neuron σ_{pi} (proposition neuron) with $(1 \leq i \leq m)$ is a tuple of the type $\sigma_{pi} = (\alpha_i, \vec{\omega}_i, \lambda_i, \mathbf{r}_i)$ where

  (a) The scalar $\alpha_i \in [0,1]$ is a fuzzy truth value to express the potential value in the σ_{pi}.

  (b) The vector $\vec{\omega}_i = (\omega_{i1}, \omega_{i2}, \ldots, \omega_{is_j})$ is the weight vector for the set of synapses within the σ_{pi}. Each element $\omega_{ij} \in [0,1]$ is called weighted value with $1 \leq j \leq s_j$, and s_j is the number of synapse.

  (c) $\mathbf{r}_i$ is a firing rule set and has the form $E/a^{\alpha} \rightarrow a^{\alpha}; d$, where $E = \{\alpha \geq \lambda_i\}$ is the firing condition, that is, the firing rules are activated when $\alpha \geq \lambda_i$. The values of $\lambda_i \in [0,1)$ denotes firing threshold. $d \in \mathbb{N}$ is the delay time to implement firing rules for proposition neurons. If the proposition neuron has no delay, the parameter d is omitted.

(iii) The set $N_r = \{\sigma_{r1}, \sigma_{r2}, \ldots, \sigma_{rn}\}$ is the *rule neuron set.* For each rule neuron, σ_{rj} $(1 \leq j \leq n)$ is in the form $\sigma_{rj} = (\alpha_j, \gamma_j, \vec{v}_j, \tau_j, \mathbf{r}_j)$ where

  (a) $\alpha_j \in [0,1]$ is the potential value in the σ_{rj}.

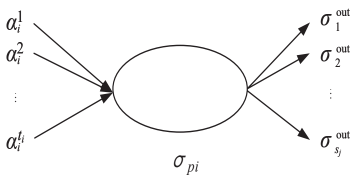
  (b) $\gamma_j \in [0,1]$ is the certainty factor, which represents the probability of firing rules in the σ_{rj}.

  (c) The vector $\vec{v}_j = (v_{j1}, v_{j2}, \ldots, v_{jt_i})$ is the weight vector for σ_{rj}. The elements $v_{ji} \in [0,1]$ is the weight value on the i^{th} synapse of σ_{rj}, where $1 \leq i \leq t_i$, and t_i is the number of synapse on neuron σ_{rj}.

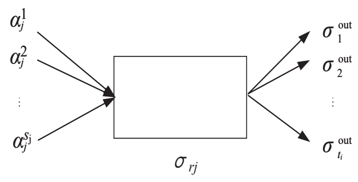
  (d) $\mathbf{r}_j$ is a firing rule set and has the form $E/a^{\alpha} \rightarrow a^{\beta}; d$ where $\alpha, \beta \in [0,1]$ and $d \in \mathbb{N}$. The expression $E = \{\alpha \geq \tau_j\}$ is called firing condition, it means that the firing rule is executed when the firing condition is satisfied, where $\tau_j \in [0,1)$ denotes the firing threshold. The parameter d is omitted when no delay is involved.

(iv) The set of synaptic connections $syn \subseteq (N_p \times N_r) \cup (N_r \times N_p)$ associated the proposition with rule neurons.
(v) IN, $OUT \subseteq N_p$ are special *proposition neurons*, that is, the *input* and *output* neurons.

The formula of potential value for proposition neurons and rule neurons is discussed below.



(a)proposition neurons



(b)rule neurons

Figure 1. Proposition and rule neurons in the WFSN P system.

In terms of proposition neurons, if multiple spikes are received, a Boolean OR operator is used to assess whether or not the firing condition is satisfied. The OR operator is here indicated as $\vee$. As shown in Fig.1a, if a series of potential spike values are sent to the proposition neuron, the potential value is $\alpha_i^{in} = \alpha_i^1 \vee \alpha_i^2 \vee \ldots \vee \alpha_i^{t_i}$. As such, if $\alpha_i^{in} \geq \lambda_i$, then the environment receives the potential $\alpha_i^{out} = \alpha_i^{in} \otimes \vec{\omega}_i$, where $\otimes$ is the multiplication operator, see[38] for formal details. However in this paper, the potential is consideblack as an interval number with an equal upper and lower limit. Therefore, the fuzzy operator is consistent with algebraic multiplication and $\alpha_i^{out} = \alpha_i^{in}\vec{\omega}_i$.

Similar for rule neurons, if multiple spikes have arrived, then the operator "$\oplus$" is used to determine the potential values and assess whether the firing condition is satisfied. If a rule neuron receives a series of potential spike values, the potential value is $\alpha_j^{in} = \alpha_j^1 \oplus \alpha_j^2 \oplus \ldots \oplus \alpha_j^{s_j}$, see Fig.1b. As such, if $\alpha_j^{in} \geq \tau_j$, then the environment receives the value $\alpha_j^{out} = (\alpha_j^{in} \oslash \vec{v}_j) \otimes \gamma_j$, where $\oplus$ and $\oslash$ are addition operator and division operator respectively for fuzzy truth values, see.[38] Also in this case, the addition and division operators are the same as algebraic addition and division, respectively, and hence $\alpha_j^{out} = (\alpha_j^{in}/\vec{v}_j)\gamma_j$.

Clearly, the WFSN P systems combine the advantages of fuzzy logic with SN P systems, that is, the model can not only process information within the SN P system framework but can also perform, by means of fuzzy logic, operations on real numbers.[41] These properties modify substantially the functioning of WFSN P systems and make them suitable to address classification problems.

3. Layeblack Spiking Neural P Systems

This section describes the proposed LSN P system. Section 3.1 describes the structure of the LSN P system, while section 3.2 describes the novel learning element and section 3.3 the novel learning process.

3.1. *LSN P systems and their structure*

The structure of a LSN P system is composed of two parts: a high-dimensional encoder and a weighted fuzzy SN P system classifier, as depicted in Fig.2. The input data are fed into the weighted fuzzy SN P system classifier by a high-dimensional encoder (that is a WFSN P system itself), the weighted fuzzy SN P system classifier consists of an example of a WFSN P system (see section 2) defined in terms of the following tuple Π:

$$\Pi = (\{a\}, \{\sigma_{p1}^1, \ldots, \sigma_{pk}^1, \sigma_{p1}^3, \sigma_{p1}^5\}, \{\sigma_{r1}^2, \ldots, \sigma_{rn}^2, \sigma_{r1}^4, \ldots, \sigma_{rn}^4\}, syn, IN, OUT)$$

where

(i) a denotes a spike.
(ii) The proposition neuron takes the form that $\sigma_{pi}^h = \{0, w_{ij}^h, \lambda_i^h, r_i^h\}$, where h is the layer label.
  (a) 0 denotes that no initial potential value is in proposition neurons.
  (b) The weights take the form of $w_{ij}^1 = rand(0,1)$ and $w_{1j}^3 = 1$, where $1 \leq i \leq k, 1 \leq j \leq n$.
  (c) The firing rules are contained in a set r_i^h where each rule is of the type $r_i^1 : E^1/a^{\alpha_i} \to a^{\alpha_i}$,

where $E^1 = \{\alpha_i \geq 0\}$ and $1 \leq i \leq k$. The last neuron is termed the bias neuron and $\alpha_k = 1$. $r_1^3 : E^3/a^o \to a^o$, where $E^3 = \{o \geq 0\}$, and $o = \theta_1 \vee \theta_2 \vee \ldots \vee \theta_n$. The θ_j is defined and presented in the next paragraph. $r^5 : a \to a$, the omission of firing condition means that the firing rule is always enabled when coming potential across 1.

(iii) A rule neuron is a set of the type $\sigma_{rj}^k = \left\{0, 1, \vec{1}, \tau_j^k, r_j^k\right\}$, where k is the layer label.

(a) The initial potential value is 0 in rule neurons.
(b) The certain factor is 1 for all rule neurons.
(c) The weight synaptic from rule neurons are set as 1.
(d) The firing rules of $\sigma_{r1}^2, \ldots, \sigma_{rn}^2$ have the form of r_j^2: $E^2/a^{\theta_j} \to a^{\theta_j}$, where $E^2 = \{\theta_j \geq 0\}$, $\theta_j = (w_{1j} \otimes \alpha_1) \oplus (w_{2j} \otimes \alpha_2) \oplus \ldots \oplus (w_{kj} \otimes \alpha_k)$, and $1 \leq j \leq n$. Neurons $\sigma_{r1}^4, \sigma_{r2}^4, \ldots, \sigma_{rn}^4$ have the spiking rules r_1^4: $E_j^4/a^{\theta_j} \to a; d_j$, where $E_j^4 = \{\theta_j \geq o\}$. It should be noted that the fuzzy operation degenerates into algebraic operations as discussed in Section 2.

(iv) $syn = \{(\sigma_{p1}^1, \sigma_{r1}^2), (\sigma_{p1}^1, \sigma_{r2}^2), \ldots, (\sigma_{pk}^1, \sigma_{rn}^2), (\sigma_{r1}^2, \sigma_{p1}^3), (\sigma_{r2}^2, \sigma_{p1}^3), \ldots, (\sigma_{rn}^2, \sigma_{p1}^3), (\sigma_{p1}^3, \sigma_{r1}^4), (\sigma_{p1}^3, \sigma_{r2}^4), \ldots, (\sigma_{p1}^3, \sigma_{rn}^4), (\sigma_{r1}^4, \sigma_{p1}^5), (\sigma_{r2}^4, \sigma_{p1}^5), \ldots, (\sigma_{rn}^4, \sigma_{p1}^5)\}$
(v) $IN = \{\sigma_{p1}^1, \sigma_{p2}^1, \ldots, \sigma_{pk}^1\}$, $OUT = \{\sigma_{p1}^5\}$.

We may note that the proposed LSN P system is not just a generalization of WFSN P system. Two reasons are listed as below.

(i) LSN P systems and WFSN P systems have different syntactic structures. The LSN P system is a layeblack model, while the WFSN P system only has one layer. In the LSN P system, only neuron σ_3^{p1} has the same syntactic structure as a proposition neuron in the WFSN P system, while the other neurons in the LSN P system have completely different syntactic structure from the proposition neuron or rule neuron in the WFSN P system. It is noted that neurons both in the LSN P system and the WFSN P system are consideblack as variants or extensions in an SN P system.[20] It is also worth pointing out that weights in the LSN P system originated from Spiking Neural P Systems with Weights,[42] instead of WFSN P system.[38]
(ii) LSN P systems and WFSN P systems have different semantics. In the LSN P system, only the neuron σ_3^{p1} has the same semantics as the proposition neuron in a WFSN P system, that is, with the fuzzy operators. The neurons in the other five layers in a LSN P system perform arithmetic operations, while all the proposition neurons or rule neurons perform fuzzy operations in a WFSN P system.

The proposed LSN P system is composed of a pre-processing layer, located on the top of the LSN P system as a high-dimensional encoder, and five subsequent layers: an input layer, a hidden layer, a comparison layer, a selection layer and an output layer. The high-dimensional encoder is implemented by a series of rule neurons and the corresponding firing rules are achieved by Taylor polynomials. Meanwhile, each pair of consecutive layers contain neurons of different types, with the neurons generating a new potential value every time the incoming potential reaches the firing condition. The function of pre-processing layer is to feed the dataset into the LSN P system.

The transition from the pre-processing layer to the first layer entails the encoding of the input of its own length/dimension into a standard length vector to be processed by the following layers (see section 3.2 for details). The input layer is fully connected to the hidden layer (all the possible arrows are indicated) through the synapses. The weight on each synapse is a random value between 0 and 1, the aim of training process is to detect those weights corresponding to the lowest loss. The synapse with the weight 1 is present in all the following layers, which means that these weights do not participate in the training process. The output of the LSN P system is a spike that carries the information related to the result of the classification task. The following section focuses on the functioning of the proposed LSN P system and describes its learning element.

3.2. *The novel learning element*

The learning element of the LSN P system includes two parts: high-dimensional coding and the weight adjustment rule. The high-dimensional coding is achieved via a high-dimensional encoder located between the pre-processing layer and the input layer. The coding scheme is based on both biological phenomena and the research on pattern recognition.

In terms of biological phenomena, various scientists have found that decision-making involves

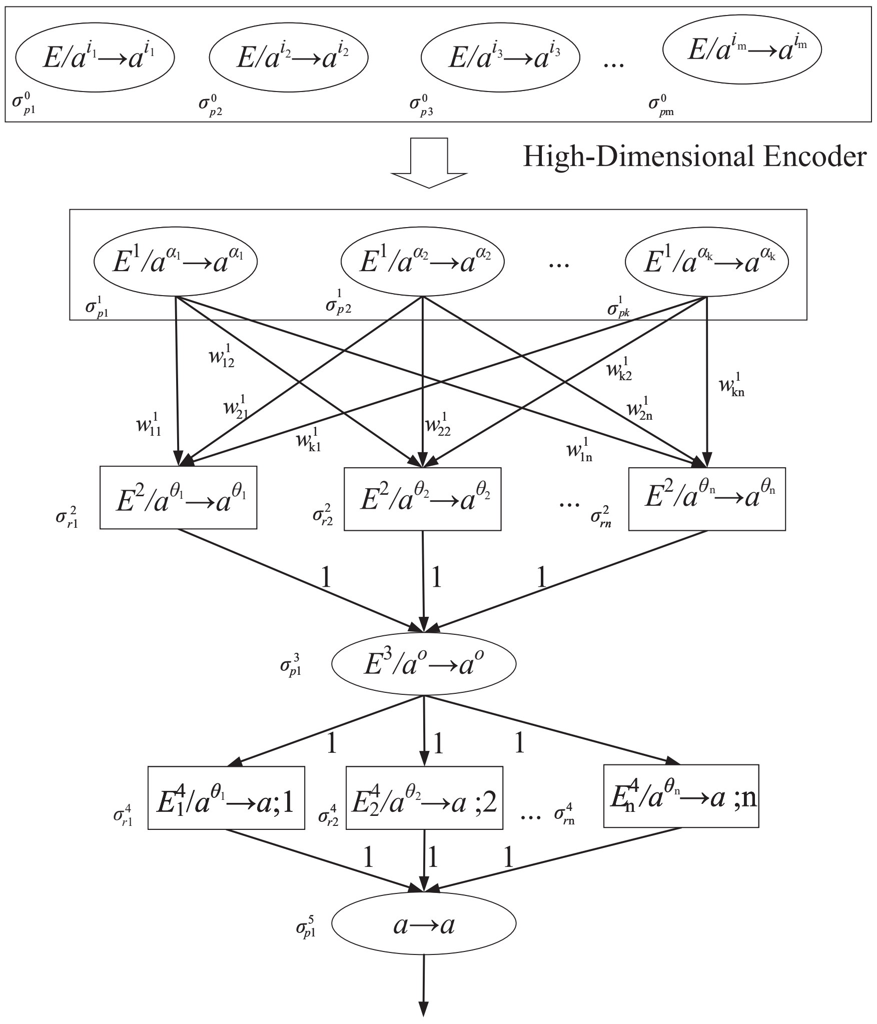


Figure 2. The structure of the proposed LSN P system

highly diverse nonlinear mixtures of task-relevant variables.[40] The nonlinear mixtures of variables carry more information which facilitates classification. However, the actual encoding scheme remains largely unclear.[40] The classification task is a problem related to finding the smooth decision boundary.

The decision boundary can be a line, surface, or higher-order surface, which depends on the dimensions of decision variables. With any smooth decision boundaries having the potential for being approximated by the Taylor decomposition. The high-dimensional encoder is represented by a nonlinear mixture of variables and implemented by Taylor polynomials.

$$\begin{aligned} f\left(x_1,...,x_n\right) = f(x_{1_0},...,x_{n_0}) + (h_1\frac{\partial}{\partial x_1} + ... \\ +h_n\frac{\partial}{\partial x_n})f(x_{1_0},...,x_{n_0}) + .. + \frac{1}{k!}(h_1\frac{\partial}{\partial x_1} + ... \\ +h_n\frac{\partial}{\partial x_n})^k f(x_{1_0},...,x_{n_0}) + R_{k+1} \end{aligned} \tag{1}$$

where:

(i) $\overrightarrow{x} = (x_{1_0}, \ldots, x_{n_0})$ is any point on the decision boundary $f(x_1, \ldots, x_n)$;
(ii) $h_i = x_i - x_{i_0}(1 \le i \le n)$;
(iii) $R_{k+1} = \frac{1}{(k+1)!}(h_1\frac{\partial}{\partial x_1} + \ldots + h_n\frac{\partial}{\partial x_n})^{K+1} f(x_{1_0} + \xi h_1, \ldots, x_{n_0} + \xi h_n)$, where $\xi \in (0, 1)$.

In Eq. (1), the constant term $f(x_{1_0},...,x_{n_0})$ is mimicked by the bias neuron σ_{pk}^1 in the LSN P system in view of fitting the decision boundary more accurately. Decision variables are transmitted into the Taylor polynomial to acquire the high-dimensional form. Finally, it is clear that the decision boundary is a linear combination of Taylor polynomial. It is noted that the Taylor polynomial is only one of the feasible ways to get high-dimensional representations. Other ascending dimension techniques, such as the kernel function, are also viable approaches which will deserve further investigation.

The other learning element is the Widrow–Hoff learning law,[39] which is for updating the weight on the synapse. The error signal is transformed into the hidden layer by the back-propagation, more detailed description is shown in Section 3.3. The Widrow–Hoff learning law is described as follows:

$$\boldsymbol{W} \Leftarrow \boldsymbol{W} + \eta(t - \tilde{t})\boldsymbol{\alpha} \tag{2}$$

where:

(i) $\boldsymbol{W}$ represents weight within the LSN P system;
(ii) η is the learning rate;
(iii) t is the expected output of neuron σ_{p1}^5;
(iv) $\tilde{t}$ is the real output of neuron σ_{p1}^5;
(v) $\boldsymbol{\alpha}$ is the potential value within the input layer.

3.3. *The learning process of the LSN P system*

The learning process of the LSN P system consists of three sub-processes: the initialisation, the training and the testing process of the system. First, the initialisation of the dataset and the LSN P system is carried out. The real-valued dataset is scaled into the interval between the maximum and minimum fuzzy truth values using the coding scheme,[43,44] as described in Eq. (3).

$$T(\boldsymbol{x}) = \frac{f_2 - f_1}{\boldsymbol{x}_{max} - \boldsymbol{x}_{min}} \cdot \boldsymbol{x} + \frac{f_1 \cdot \boldsymbol{x}_{max} - f_2 \cdot \boldsymbol{x}_{min}}{\boldsymbol{x}_{max} - \boldsymbol{x}_{min}} \tag{3}$$

where f_2 and f_1 are the maximum and minimum of the fuzzy truth value, respectively. $\boldsymbol{x}_{max}$ and $\boldsymbol{x}_{min}$ are the maximum and minimum of each feature in the dataset, respectively.

To employ a LSN P system, the structure and parameters need to be pre-determined. In terms of the structure, the number of neurons in the pre-processing layer is the length of the input vector (m) within the dataset. The number of neurons in the input layer is determined by the degree of Taylor polynomial (K), with the number $\sum_{i=1}^{K} C_{m+i-1}^{i}$ based on the number of Taylor expansion terms. Meanwhile, the number of neurons in the hidden layer and the selection layer is determined according to the output labels of the dataset, with the neuron number within the comparison layer and output layer fixed at one.

In terms of the weight, these parameters are initialised to be a randomly value at between 0 and 1 between the pre-processing layer and the input layer, with the maximum number of iterations set as a large enough number and the learning rate set to an appropriate size. The other parameters are set in terms of the description of the LSN P system.

Following this, the training process of the LSN P system is introduced. Here, the input potential of the neurons $\sigma_{p1}^0, \sigma_{p2}^0, ...\sigma_{pm}^0$ within the pre-processing layer is the normalised value of the dataset by using the Eq. (3). The information processing mechanism of the high-dimension encoder between the pre-processing layer and the input layer entails transmitting the length of the input vector into a higher length based on the Taylor expansion theory. The potential value within the neurons $\sigma_{p1}^1, \sigma_{p2}^1, ...\sigma_{pk}^1$ is the numerical value of Taylor polynomial term by using Eq. (1) in terms of the feed data. The output potential values are multiplied by the initialised weights and are deliveblack to the neurons $\sigma_{r1}^2, \sigma_{r2}^2, ...\sigma_{rn}^2$ within the hidden layer. Meanwhile, the comparison layer is composed of the neuron σ_{p1}^3, and the output spike is the maximum potential value of the neurons in the hidden layer, while in the selection layer, only one neuron is activated, which emits a spike to the output layer with a delay corresponding to the fire rule in the neurons $\sigma_{r1}^4, \sigma_{r2}^4, ...\sigma_{rn}^4$. The output layer is composed of the neuron σ_{p1}^5 and the spike emission time is the label of output.

Then, the mean square error between the real output time and the desiblack output time is calculated as the loss of the LSN P system, as shown in Eq. (4). The loss is backpropagation to all layers of LSN P system to minimize. For constant weights, since the partial derivative of the loss is 0, these weights do not change during the back-propagation. Because of only one layer of variable weights, the partial derivative of the loss with respect to the weight is the manifestation of the Widrow-Hoff learning law. So the The training process is performed until the maximum iterations (*max_iter*) are reached.

$$Loss = \frac{1}{N}\sum_{i=1}^{N}(t_i - \tilde{t}_i)^2 \qquad (4)$$

where N is the total number of samples.

The last part is the testing process of the LSN P system. Here, all the parameters of the trained LSN P system are saved and the LSN P system is assessed based on the testing set to determine the classification ability and the generalisation ability.

To further explain the structure and functioning of the proposed LSN P systems, Algorithm 1 presents the training algorithm of the LSN P system in terms of pseudocode.

Algorithm 1 Training algorithm

```
Input: N sets of labeled dataset.
Output: Weights value $\boldsymbol{W}$.
1: Initialisation parameters: $K$, $\eta$, max_iter, $\boldsymbol{W}$.
2: The dataset is normalised by using Eq. (3).
3: Feed normalised values into neurons $\sigma^0_{p1}, \sigma^0_{p2}, \ldots, \sigma^0_{pm}$.
4: Compute spike potential for neurons $\sigma^1_{p1}, \sigma^1_{p2}, \ldots, \sigma^1_{pk}$ by using the numerical value of the Taylor polynomial shown Eq. (1).
5: for iter = 1:max_iter do
6:     Compute output potential within neurons $\sigma^2_{r1}, \sigma^2_{r2}, \ldots, \sigma^2_{rn}$.
7:     Compute the output potential of neuron $\sigma^3_{p1}$.
8:     Activate rule neurons $\sigma^4_{r1}, \sigma^4_{r2}, \ldots, \sigma^4_{rn}$ via the firing rule.
9:     Get data class by the spiking time of $\sigma^5_{p1}$.
10:    Compute the loss and update $\boldsymbol{W}$ by using (4) and (2), respectively.
11: end for
```

4. Experimental Results

This section displays the potential of LSN P systems to resolve specific classification problems.

The performance evaluation was based on the overall training/testing accuracy, with the accuracy rate the ratio between the number of correctly classified samples and the total number of samples.

The comparison of the performances of the learning algorithms was implemented using the non-parametric Holm–Bonferroni procedure.[45] Meanwhile, the attendant rank is the average rank between the training set and the testing set, as described in Eq. (5).

$$R = 0.5 * R_1 + 0.5 * R_2 \qquad (5)$$

where R_1 and R_2 are the average rank within the training set and the testing set, respectively, and R is the overall rank of the learning algorithm.

The performance results of the LSN P system were generated using Python 3.7.0 in a Windows Operating system on a machine with 28 logical cores, 64 GB of RAM and a speed of 3.3 GHz. All the results reported in this section represent the average training/testing accuracy obtained over 20 random trials.

4.1. *Boolean Logic Problems*

In this subsection, we use the LSN P system to classify specific Boolean logic problems, including the OR-problem and the XOR-problem. To ensure that readers can gain a better understanding of how the LSN P system works, examples are used to elaborate on the solution for boolean logic problems.

Table 1. The iteration process of the OR-problem

Iterations	w^1_{11}	w^1_{21}	w^1_{31}	w^1_{12}	w^1_{22}	w^1_{32}	Loss
1^{st}	0.56	0.28	0.67	0.53	0.52	0.49	0.5
2^{nd}	0.66	0.38	0.67	0.43	0.42	0.49	0.25
3^{rd}	0.66	0.38	0.57	0.43	0.42	0.59	0.5
4^{th}	0.66	0.48	0.57	0.43	0.32	0.59	0

4.1.1. *OR-problem*

The so-called OR-problem consists of a linear classification with two input values and states. As such, the structure of the network can be confirmed and the weight between σ^1_{pi} and σ^2_{rj} can be randomly initialised. Let us suppose that $w^1_{11} = 0.46, w^1_{21} = 0.18, w^1_{31} = 0.67, w^1_{12} = 0.63, w^1_{22} = 0.62, w^1_{32} = 0.49$ and the learning rate is set as 0.1, where w^1_{31} and w^1_{32} are the weights from the bias neuron. The maximum and minimum fuzzy truth values are set as 0 and 1, respectively. Table 1 shows the functioning of the algorithm across multiple iterations.

The loss of the LSN P system is 0; then, the training process will stop and the set of weights will be saved.

4.1.2. *The XOR-problem*

Since the XOR-problem is non-linear, the high-dimensional encoder is crucially important. The increase in dimensions is given by the following formula:

$$(x_1, x_2) \to (x_1, x_2, x_1^2, x_2^2, x_1 x_2) \tag{6}$$

From the above formula, it is clear that the degree of Taylor polynomial is two. The main reason is that the XOR-problem presents an extremely simple non-linear model. In the case of non-linear data, a higher-degree Taylor polynomial must be used. The computational budget in terms of number of iterations is set as 20 and the learning rate is set as 0.1. Here, these higher-dimensional information were fed into the model, with the attendant loss (see Eq. (4)) shown in Fig.3.

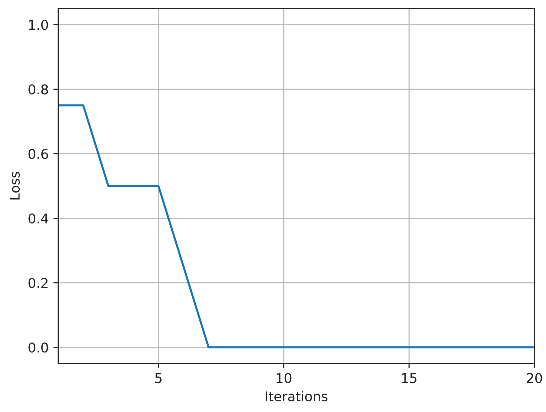


Figure 3. The XOR-problem loss.

4.2. *Simple classification problems*

4.2.1. *Breast cancer problem*

The breast cancer problem is a binary classification problem, one that is composed of nine labels, including age, menopause and the size of the tumour. The final output is the phenomenon of recurrence or no-recurrence. Given the 13 missing data, the total number of training sets and testing sets is 350 and 333, respectively. The proportion of target and non-target classes in the training set is 49.8% and 52.0%, respectively.

The fuzzy coding scheme is carried out as described in Eq. (3), with the maximum and minimum of the fuzzy truth value 1 and 0, respectively. Here, the expansion order of the Taylor series was chosen to be three in the coding scheme, while the learning rate and the maximum number of iterations were set to be 0.05 and 450 respectively via the trial and error method. The high-dimensional normalised data can then be fed into the network for training the LSN P system.

Here, 20 independent runs were performed to test the convergence of the systems. The mean loss (over the 20 runs) calculated as in Eq. (4) and 90% confidence interval (calculated by empirical distribution function[46]) are presented in Fig.4, with the testing accuracy = 100%. Here, it was clear that the LSN P system was convergent and exhibited a good learning ability.

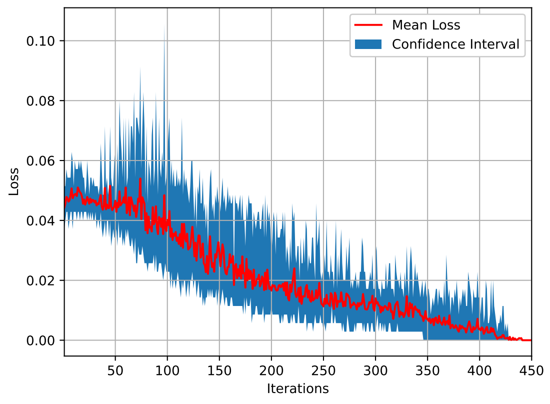


Figure 4. The breast cancer problem mean loss over 20 run (black line) and associated confidence interval (blue band) at 90%.

4.2.2. *Iris flower problem*

The iris flower problem consists of specific iris characteristics (four features), with the aim being to determine the species of the irises based on the provided sample information. Here, the total number of samples was 150, which were evenly divided into three classes (Setosa, Versicolour, Virginia), with 50% of the samples used for training and the remaining 50% of the dataset used as the testing set. The proportion of different classes in the training set is 52.0%, 52.0%, and 46.0%, respectively.

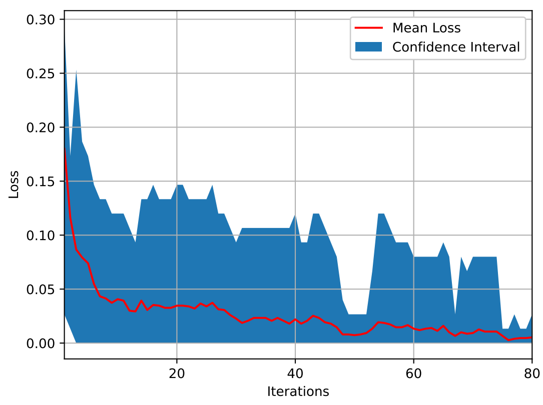


Figure 5. The iris problem mean loss over 20 run (black line) and associated confidence interval (blue band) at 90%.

The parameters used with the iris flower problem were the same as those used with the breast cancer problem, but the maximum number of iterations set as 80 is enough. Again, the convergence of the LSN P system was verified using 20 independent runs. The results indicated that the LSN P system has excellent potential in terms of learning ability and convergence, see the trends in Fig.5.

4.3. *Performance comparison using other learning algorithms*

In this subsection, we discuss the performance evaluation of the LSN P system in terms of five benchmark datasets from the UCI machine learning repository.[47] Here, the Breast Cancer, Ionosphere, and PIMA datasets are binary classification problems, while the Iris and Wine datasets are multiple classification problems. The number of input features and output classes are shown in Table 2.

Table 2. Description of data sets.

Data set	Features	Classes	Samples	
			Training	Testing
Breast Cancer	9	2	350	333
Ionosphere	34	2	175	176
PIMA	8	2	384	384
Iris	4	3	75	75
Wine	13	3	60	118

The performance results were compablack with some spiking learning algorithms for SNNs and non-spiking classifiers, namely, the SpikeProp,[48] SWAT,[49] SRESN,[50] OMLA,[51] MC-SEFRON,[52] TMM-SNN,[53] DC-SNN,[54] MeST,[55] finite element machine (FEMa),[56] deep belief network (DBN)[57] and ResNet[58]. The results of the spiking learning algorithms were obtained from the respective literature and the non-spiking classifiers were implemented on python library, with the results for all algorithms generated using the same spilt regarding the training/testing data. The degree of certainty for FEMa is set as 3. For the DBN model, the activation function is selected as Rectified Linear Unit (ReLu), the network layers of DBN are set as 5 and neurons in the hidden layers are chosen to be 10, 15 and 5. For the ResNet, a four-layer fully connected neural network and a residual block from the first layer to the last layer. The neurons in the hidden layer are 10 and 15 respectively and a ReLu activation function.

Table 3. Hyperparameters of the model.

Data set	η	K	max_iter
Breast Cancer	0.05	3	450
Ionosphere	0.01	3	150
PIMA	0.0005	3	1000
Iris	0.05	3	80
Wine	0.05	3	100

The results of the LSN P system are presented in terms of mean accuracy, which was obtained following 20 random trials. The maximum and minimum fuzzy truth values are set as 0 and 1 for all datasets while the other hyperparameters of the LSN P system are as shown in Table 3 via trial and error. For the PIMA datasets, due to the overlap of the high interclass, a larger learning rate may cause the LSN P system to oscillate and experience some difficulty in converging.

Table 4 shows the results of the performance evaluation and the number of configurable parameters for the LSN P system. Here, it was clear that in terms of classification problems with low input dimensions, such as the iris problem, both the training and the testing accuracy of the LSN P system were superior to that of the other algorithms. Meanwhile, with regard to classification problems with medium/high input dimensions, the proposed algo-

Table 4. Performance evaluation and the number of configurable parameters on UCI data sets.

	Breast Cancer (# parameters)	Ionosphere (# parameters)	PIMA (# parameters)	Iris (# parameters)	Wine (# parameters)
FEMa[56]	100/96.9(1)	100/63.8(1)	100/70.1(1)	100/96.9(1)	100/97.4(1)
DBN[57]	97.8/97.3(357)	96.0/90.2(607)	78.9/73.7(347)	98.6/97.4(313)	99.7/96.8(403)
ResNet[58]	95.0/97.3(317)	97.7/89.4(617)	77.5/72.1(305)	98.2/93.8(278)	99.6/96.5(395)
SpikeProp[48]	97.3/97.2(13680)	89.0/86.5(82800)	78.6/76.2(16640)	97.2/96.7(4480)	99.2/96.8(12960)
SWAT[49]	96.5/95.8(1404)	86.5/90.0(5304)	77.0/72.1(1404)	96.7/92.4(936)	98.6/92.3(2028)
SRESN[50]	97.7/97.2([432,648])	91.9/88.6([3264,4692])	70.5/69.9([486,756])	96.9/97.3([144,240])	96.9/91.0 ([390,780])
OMLA[51]	97.4/97.8(162)	94.0/**93.5**([4080,5304])	78.6/77.9(1134)	97.9/97.9([144,192])	98.5/97.9([312,546])
MC-SEFRON[52]	98.4/97.4(32508)	94.2/89.7(122808)	77.5/75.4(28896)	98.4/97.1(21672)	98.8/94.6(70434)
TMM-SNN[53]	97.4/97.2(48762)	98.7/92.4(184212)	79.7/**78.1**(43344)	97.5/97.2(28896)	**100**/97.5(93912)
DC-SNN[54]	97.4/97.8(448)	97.1/92.7(4020)	78.6/77.8(612)	96.1/97.7(297)	98.2/96.3(810)
MesT[55]	98.2/**98.0**(48762)	98.3/93.2(184212)	78.4/77.3(43344)	98.1/98.0(28896)	**100**/**98.0**(93912)
LSN P system	**100**/97.9(440)	**100**/92.6(15540)	**80.9**/75.6(330)	**99.5**/**98.8**(105)	**100**/97.5 (1680)

rithm exhibited a similar performance as the other algorithms, while its performance in terms of training set accuracy with the classification problem was better than that of the majority of the existing learning algorithms. Overall, nonlinear mixed selectivity is clearly a powerful tool for addressing a variety of classification problems.

4.4. *Statistical analysis of the performance comparison*

The performance of the existing learning algorithms (OMLA, SWAT, SRESN, etc.) were compablack using the non-parametric Holm–Bonferroni procedure in relation to all the classification datasets presented in this paper, with the average ranks of both the training set and the testing set under consideration. The results of the ranking are presented in Table 5. which clearly indicated that the proposed LSN P system achieved the highest rank among the entire set of the eleven learning algorithms consideblack in this study. Specifically, while the results of the Holm–Bonferroni procedure indicated that the LSN P system did not significantly outperform the MeST ,TMM-SNN, OMLA, FMEa and DBN learning algorithm, it clearly did significantly outperform the other six competitors. The obtained results (Tables 4 and 5) demonstrated that the proposed LSN P system displays a good performance in classification problems.

4.5. *Image classification problem*

To assess the proposed algorithm on a more complex classification problem, we evaluate the LSN P system on the MNIST dataset.[59] The MNIST dataset consists of 70000 handwritten digits with 28×28 pixels. Compablack with the UCI dataset, MNIST dataset is a large-sized dataset. The ascending dimensional technology used directly will cause "the curse of dimension", therefore, we use multi-layers convolution and pool technique to mine the feature of an image.

For the convolution layer, three-layer small convolution kernels (5*5) and the number of channels are set as 8-16-32 for each convolution layer. The stride and padding parameter for each convolution layer is set as 1 and 2, respectively. The max-pooling layer is the successor of the convolution layer, the kernel size is 2*2, the stride and padding parameter for each convolution layer is set as 1 and 0, respectively. The implementation of convolution and max-pooling convert the original image into 32 feature maps with the size of 3*3. The ascending dimensional technology is applied for every feature map and the LSN P system is proposed to recognize the class information based on ascended dimensional data. The learning rate, the degree of Taylor polynomial and maximum number of iterations is set as 3, 0.001 and 3000, respectively. The error signal is propagated within the LSN P system, not backpropagate into the multi convolution-pooling layers. That is, the weights and biases in the multi convolution-pooling layers are fixed at random values. The test accuracy and configurable parameters of LSN P system is shown in Table 6 and compablack with the result reported in some state-of-art learning algorithms.[60–68] Furthermore, it should be noted that the number of parameters of the Spiking ConvNet[66] is inherited from that

Table 5. Holm–Bonferroni procedure in terms of accuracy rate; reference algorithm = LSN P (Rank = 10.4).

j	Learning Algorithm	Rank	z_j	p_j	δ/j	Hypothesis
1	MeST	9.55	-0.57	2.83e-01	5.00e-2	Accept
2	TMM-SNN	8.00	-1.62	5.28e-02	2.50e-2	Accept
3	OMLA	7.85	-1.72	4.28e-02	1.67e-2	Accept
4	FEMa	7.45	-1.99	2.34e-02	1.25e-2	Accept
5	DBN	7.40	-2.02	2.16e-02	1.00e-2	Accept
6	DC-SNN	6.45	-2.66	3.87e-03	8.33e-3	Rejected
7	MC-SEFRON	5.95	-3.00	1.35e-03	7.14e-3	Rejected
8	ResNet	4.85	-3.74	9.13e-05	6.25e-3	Rejected
9	SpikeProp	4.55	-3.94	4.01e-05	5.56e-3	Rejected
10	SRESN	3.00	-4.9	3.03e-07	5.00e-3	Rejected
11	SWAT	2.55	-5.29	6.03e-08	4.45e-3	Rejected

Table 6. Comparison with typical SNNs and DLNs

Learning algorithm	Training type	Accuracy (%)	# parameters
NDC[60]	Supervised	88.2	23560
Rectangular STDP[61]	Unsupervised	93.5	313600
Exponential STDP[62]	Unsupervised	95.0	5017600
AMAP-GABP STDP[63]	Unsupervised	95.1	5001605
BP-STDP[64]	Supervised	97.2	468500
Alpha Synaptic[65]	Supervised	97.96	269960
Spiking ConvNet[66]	Supervised	99.12	51140
MCDNN[67]	Supervised	99.77	1932895
DropConnect[68]	Supervised	99.79	3125047
LSN P system	Supervised	96.87	70090

of a convolutional neural network. Thus, the training of ConvNet is closer to a backpropagation-based algorithm than to a bio-inspired learning algorithm. Although the backpropagation-based algorithm[64–68] is effective and outperformance, however, some studies show that multilayer backpropagation appears biologically unrealistic.[69] The biologically plausible learning rule is also comparable to SNNs and DLNs.

4.6. *Algorithm Complexity analysis*

The time cost for spiking learning algorithms is rather large and usually executed on parallel computing units, such as FPGA and Memristor. Also develop chips or system-on-chip for spiking neural networks have been used, see SpiNNaker[70] and TrueNorth.[71] The complexity of spiking learning algorithms is much higher than that of other machine learning algorithms. The time cost is rather high when it meets big data. For the FEMa, the time complexity is $\mathcal{O}(n)$, where n is the number of training points. For the DBN algorithm, the worst time complexity is $\mathcal{O}(2N_1n + N_2n)$, where N_1 and N_2 is the maximum iteration of the contrasting divergence algorithm and back fine-tune learning algorithm respectively, n is the number of training points. The time complexity of ResNet and the proposed algorithm is $\mathcal{O}(N \cdot n)$, where N is the maximum iteration of the training process, n is the number of training points. This phenomenon shows that the proposed algorithm has a modest time cost in the case of rather high accuracy. In terms of space cost, Taylor polynomials generally produce more parameters when compablack with other algorithms such as NN and SVM. This drawback affects the entire family of spiking learning algorithms. On the other hand, even spiking learning algorithms that simulate structure and size of biological neural systems, e.g. about 10^{11} neurons with each neuron associated with $10^3 \sim 10^4$ synapses, the “space cost” is perfectly manageable by a modern computer [72].

5. Concluding Remarks

In this paper, we proposed a computational model belonging to the family of membrane computing systems for resolving classification problems. The proposed model, namely, the LSN P system, is a multilayer network composed of a number of WFSN P systems. The proposed model makes use of the flexibility of membrane computing to build a learning system that mimics the behaviour of a biological neural system more accurately than classical neural networks. This accuracy is achieved through the use of the multiple types of neurons and rules that P systems can accommodate. The motivation behind this study was based on the observation that certain classification tasks can generally be resolved by a biological brain.

The proposed LSN P system was tested in terms of four classification problems: a base classification problem, a binary classification problem, a multiple classification problem and the image classification problem. The obtained numerical results indicated that for all problems, the proposed LSN P system exhibited an excellent performance. We anticipate that in the future, the LSN P system will be applied to more sophisticated scenarios, including fault diagnosis, handwriting recognition, natural language processing, etc., where its powerful classification capabilities can be further exploited

Further improvements on this work will take two specific directions. The first will involve the spike-timing-dependent plasticity rule to adjust the weight on the synapse, while the second will involve investigating specific regularisation techniques to improve the generalization.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (61972324), the Sichuan Science and Technology Program (2021YFS0313, 2021YFG0133), the Artificial Intelligence Key Laboratory of Sichuan Province (2019RYJ06) and the Beijing Advanced Innovation Center for Intelligent Robots and Systems (2019IRS14).

Bibliography

1. F. Rosenblatt, *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*, *Neurocomputing: Foundations of Research* (MIT Press, Cambridge, MA, USA, 1988), Cambridge, MA, USA, p. 89–114.
2. J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences* **79**(8) (1982) 2554–2558.
3. D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning representations by back-propagating errors, *Nature* **323**(6088) (1986) 533–536.
4. M. Ahmadlou and H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integrated Computer-Aided Engineering* **17**(3) (2010) p. 197–210.
5. K. Alam, N. Siddique and H. Adeli, A dynamic ensemble learning algorithm for neural networks, *Neural Computing and Applications* **32**(12) (2020) 8675–8690.
6. S. Ghosh-Dastidar and H. Adeli, Spiking neural networks, *International Journal of Neural Systems* **19**(04) (2009) 295–308.
7. K. Roy, A. Jaiswal and P. Panda, Towards spike-based machine intelligence with neuromorphic computing, *Nature* **575**(7784) (2019) 607–617.
8. R. Khanna, D. Oh and Y. Kim, Through-wall remote human voice recognition using doppler radar with transfer learning, *IEEE Sensors Journal* **19**(12) (2019) 4571–4576.
9. J. Cui and B. Zhang, VeRA: A simplified security risk analysis method for autonomous vehicles, *IEEE Transactions on Vehicular Technology* **69**(10) (2020) 10494–10505.
10. K. Muhammad, S. Khan, J. D. Ser and V. H. C. d. Albuquerque, Deep learning for multigrade brain tumor classification in smart healthcare systems: A prospective survey, *IEEE Transactions on Neural Networks and Learning Systems* **32**(2) (2021) 507–522.
11. G. K. . D. C. William Lotter, A neural network trained for prediction mimics diverse features of biological neurons and perception, *Nature Machine Intelligence* **2** (2020) 210–219.
12. S. M. Bohte and J. N. Kok, Applications of spiking neural networks, *Information Processing Letters* **95** (September 2005) p. 519–520.
13. S. Ghosh-Dastidar and H. Adeli, Third generation neural networks: Spiking neural networks, *Advances in Computational Intelligence*, (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009), pp. 167–178.
14. S. Ghosh-Dastidar and H. Adeli, Improved spiking neural networks for eeg classification and epilepsy and seizure detection, *Integrated Computer-Aided Engineering* **14** (aug 2007) p. 187–212.
15. H. Adeli and S. Ghosh-Dastidar, *Automated EEG-based diagnosis of neurological disorders: Inventing the future of neurology* (CRC press, 2010).
16. S. Ghosh-Dastidar and H. Adeli, A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection, *Neural Networks* **22**(10) (2009) 1419–1431.
17. J. Ridet, A. Privat, S. Malhotra and F. Gage, Reac-

tive astrocytes: cellular and molecular cues to biological function, *Trends in Neurosciences* **20**(12) (1997) 570–577.

18. G. Păun, Computing with membranes, *Journal of Computer and System Sciences* **61**(1) (2000) 108–143.
19. C. Martín-Vide, G. Păun, J. Pazos and A. Rodríguez-Patón, Tissue P systems, *Theoretical Computer Science* **296**(2) (2003) 295–326.
20. M. Ionescu, G. Păun and T. Yokomori, Spiking neural P systems, *Fundamenta Informaticae* **71**(2, 3) (2006) 279–308.
21. T. Wu, L. Pan, Q. Yu and K. C. Tan, Numerical spiking neural p systems, *IEEE Transactions on Neural Networks and Learning Systems* **32**(6) (2021) 2443–2457.
22. L. Pan, J. Wang and H. J. Hoogeboom, Spiking neural P systems with astrocytes, *Neural Computation* **24**(3) (2012) 805–825.
23. Z. Lv, T. Bao, N. Zhou, H. Peng, X. Huang, A. Riscos-Núñez and M. J. Pérez-Jiménez, Spiking neural p systems with extended channel rules, *International Journal of Neural Systems* **31**(01) (2021) p. 2050049.
24. L. Pan, G. Păun, G. Zhang and F. Neri, Spiking neural P systems with communication on request, *International Journal of Neural Systems* **27**(08) (2017) p. 1750042.
25. T. Wu, F.-D. Bîlbîe, A. Păun, L. Pan and F. Neri, Simplified and yet turing universal spiking neural P systems with communication on request, *International Journal of Neural Systems* **28**(08) (2018) p. 1850013.
26. T. Wu, A. Păun, Z. Zhang and L. Pan, Spiking neural p systems with polarizations, *IEEE Transactions on Neural Networks and Learning Systems* **29**(8) (2018) 3349–3360.
27. H. Peng and J. Wang, Coupled neural p systems, *IEEE Transactions on Neural Networks and Learning Systems* **30**(6) (2019) 1672–1682.
28. R. T. A. de la Cruz, F. G. Cabarle and H. N. Adorna, Generating context-free languages using spiking neural p systems with structural plasticity, *Journal of Membrane Computing* **1**(3) (2019) 161–177.
29. S. Verlan, R. Freund, A. Alhazov, S. Ivanov and L. Pan, A formal framework for spiking neural p systems, *Journal of Membrane Computing* **2**(4) (2020) 355–368.
30. P. P. L. Lazo, F. G. C. Cabarle, H. N. Adorna and J. M. C. Yap, A return to stochasticity and probability in spiking neural p systems, *Journal of Membrane Computing* **3**(2) (2021) 149–161.
31. G. Zhang, H. Rong, F. Neri and M. J. Pérez-Jiménez, An optimization spiking neural P system for approximately solving combinatorial optimization problems, *International Journal of Neural Systems* **24**(05) (2014) p. 1440006.
32. M. Zhu, Q. Yang, J. Dong, G. Zhang, X. Gou, H. Rong, P. Paul and F. Neri, An adaptive optimization spiking neural p system for binary problems, *International Journal of Neural Systems* **31**(01) (2021) p. 2050054.
33. T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang and M. J. Pérez-Jiménez, Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, *IEEE Transactions on Power Systems* **30**(3) (2015) 1182–1194.
34. G. Zhang, H. Rong, P. Paul, Y. He, F. Neri and M. J. Pérez-Jiménez, A complete arithmetic calculator constructed from spiking neural p systems and its application to information fusion, *International Journal of Neural Systems* **31**(01) (2021) p. 2050055.
35. J. Wang and H. Peng, Adaptive fuzzy spiking neural P systems for fuzzy inference and learning, *International Journal of Computer Mathematics* **90**(4) (2013) 857–868.
36. Z. Chen, P. Zhang, X. Wang, X. Shi, T. Wu and P. Zheng, A computational approach for nuclear export signals identification using spiking neural P systems, *Neural Computing and Applications* **29**(3) (2018) 695–705.
37. T. Song, L. Pan, T. Wu, P. Zheng, M. L. D. Wong and A. Rodríguez-Patón, Spiking neural P systems with learning functions, *IEEE Transactions on NanoBioscience* **18**(2) (2019) 176–190.
38. J. Wang, P. Shi, H. Peng, M. J. Pérez-Jiménez and T. Wang, Weighted fuzzy spiking neural P systems, *IEEE Transactions on Fuzzy Systems* **21**(2) (2013) 209–220.
39. Zi-Qin Wang, M. T. Manry and J. L. Schiano, LMS learning algorithms: misconceptions and new results on converence, *IEEE Transactions on Neural Networks* **11**(1) (2000) 47–56.
40. M. Rigotti, O. Barak, M. R. Warden, X.-J. Wang, N. D. Daw, E. K. Miller and S. Fusi, The importance of mixed selectivity in complex cognitive tasks, *Nature* **497**(7451) (2013) 585–590.
41. G. Zhang, M. J. Pérez-Jiménez and M. Gheorghe, *Real-life applications with membrane computing* (Springer, 2017).
42. J. Wang, H. J. Hoogeboom, L. Pan, G. Păun and M. J. Pérez-Jiménez, Spiking neural p systems with weights, *Neural Computation* **22**(10) (2010) 2615–2646.
43. T. Gollisch and M. Meister, Rapid neural coding in the retina with relative spike latencies, *Science* **319**(5866) (2008) 1108–1111.
44. L. Kostal, P. Lansky and J. P. Rospars, Neuronal coding and spiking randomness., *European Journal of Neuroscience* **26**(10) (2007) 2693–2701.
45. F. Caraffini, F. Neri and M. Epitropakis, Hyper-SPAM: A study on hyper-heuristic coordination strategies in the continuous domain, *Information Sciences* **477** (2019) 186–202.
46. F. Dekking, C. Kraaikamp, H. Lopuhaä and L. Meester, *A Modern Introduction to Probabil-*

ity and Statistics Understanding - Why and How (Springer, 2005).

47. M. Lichman, UC Irvine Machine Learning Repository http://archive.ics.uci.edu/ml/index.php.
48. S. M. Bohte, J. N. Kok and H. La Poutré, Error-backpropagation in temporally encoded networks of spiking neurons, *Neurocomputing* **48**(1) (2002) 17–37.
49. J. J. Wade, L. J. McDaid, J. A. Santos and H. M. Sayers, SWAT: A spiking neural network training algorithm for classification problems, *IEEE Transactions on Neural Networks* **21**(11) (2010) 1817–1830.
50. S. Dora, K. Subramanian, S. Suresh and N. Sundararajan, Development of a self-regulating evolving spiking neural network for classification problem, *Neurocomputing* **171** (2016) 1216–1229.
51. S. Dora, S. Suresh and N. Sundararajan, Online meta-neuron based learning algorithm for a spiking neural classifier, *Information Sciences* **414** (2017) 19–32.
52. A. Jeyasothy, S. Sundaram, S. Ramasamy and N. Sundararajan, A novel method for extracting interpretable knowledge from a spiking neural classifier with time-varying synaptic weights, *arXiv preprint arXiv:1904.11367* (2019).
53. S. Dora, S. Sundaram and N. Sundararajan, An interclass margin maximization learning algorithm for evolving spiking neural network, *IEEE Transactions on Cybernetics* **49**(3) (2019) 989–999.
54. P. Machingal, M. Thousif, S. Dora and S. Sundaram, Self-regulated learning algorithm for distributed coding based spiking neural classifier, *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–7.
55. A. Jeyasothy, S. Ramasamy and S. Sundaram, Meta-neuron learning based spiking neural classifier with time-varying weight model for credit scoring problem, *Expert Systems with Applications* **178** (2021) p. 114985.
56. D. R. Pereira, M. A. Piteri, A. N. Souza, J. P. Papa and H. Adeli, FEMa: a finite element machine for fast learning, *Neural Computing and Applications* **32**(10) (2020) 6393–6404.
57. R. Salakhutdinov and G. Hinton, An efficient learning procedure for deep boltzmann machines, *Neural Computation* **24**(8) (2012) 1967–2006.
58. K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
59. Y. LeCun, C. Cortes and C. J. Burges, The mnist database of handwritten digits http://yann.lecun.com/exdb/mnist/.
60. M. H. Rafiei and H. Adeli, A new neural dynamic classification algorithm, *IEEE Transactions on Neural Networks and Learning Systems* **28**(12) (2017) 3074–3083.
61. D. Querlioz, O. Bichler, P. Dollfus and C. Gamrat, Immunity to device variations in a spiking neural network with memristive nanodevices, *IEEE Transactions on Nanotechnology* **12**(3) (2013) 288–295.
62. P. Diehl and M. Cook, Unsupervised learning of digit recognition using spike-timing-dependent plasticity, *Frontiers in Computational Neuroscience* **9** (2015).
63. S. Nazari and K. faez, Spiking pattern recognition using informative signal of image and unsupervised biologically plausible learning, *Neurocomputing* **330** (2019) 196–211.
64. A. Tavanaei and A. Maida, Bp-stdp: Approximating backpropagation using spike timing dependent plasticity, *Neurocomputing* **330** (2019) 39–47.
65. I.-M. Comşa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo and J. Alakuijala, Temporal coding in spiking neural networks with alpha synaptic function: Learning with backpropagation, *IEEE Transactions on Neural Networks and Learning Systems* (2021) 1–14.
66. P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu and M. Pfeiffer, Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing, *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.
67. D. Ciregan, U. Meier and J. Schmidhuber, Multi-column deep neural networks for image classification, *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.
68. L. Wan, M. Zeiler, S. Zhang, Y. Le Cun and R. Fergus, Regularization of neural networks using dropconnect, *International conference on machine learning*, 2013, pp. 1058–1066.
69. J. C. Whittington and R. Bogacz, Theories of error back-propagation in the brain, *Trends in Cognitive Sciences* **23**(3) (2019) 235–250.
70. L. A. Plana, D. Clark, S. Davidson, S. Furber, J. Garside, E. Painkras, J. Pepper, S. Temple and J. Bainbridge, Spinnaker: Design and implementation of a gals multicore system-on-chip, *ACM Journal on Emerging Technologies in Computing Systems* **7**(4) (2011).
71. F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson and D. S. Modha, Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **34**(10) (2015) 1537–1557.
72. R. M. Wang, C. S. Thakur and A. van Schaik, An FPGA-based massively parallel neuromorphic cortex simulator, *Frontiers in Neuroscience* **12** (2018).