Spiking Neural P Systems with Learning Functions

Spiking Neural P Systems with Learning Functions

Tao Song, *Senior Member, IEEE*, Linqiang Pan, *Member, IEEE*, Tingfang Wu, Pan Zheng, *Senior Member, IEEE*, M. L. Dennis Wong *Senior Member, IEEE*, Alfonso Rodríguez-Patón

***Abstract*—Spiking neural P systems, namely SN P systems, are a class of distributed and parallel neural-like computing models, inspired from the way neurons communicate by means of spikes. In this research, a new variant of the systems, called SN P systems with learning functions, is introduced. Such systems can dynamically strengthen and weaken connections among neurons during the computation. A class of specific SN P systems with simple Hebbian learning function is constructed to recognize English letters. The experimental results show that the SN P systems achieve average accuracy rate 98.76% in the test case without noise. In the test cases with low, medium and high noise, the SN P systems outperform Back Propagation (BP) neural networks and probabilistic neural networks. Moreover, comparing with spiking neural networks, SN P systems perform a little better in recognizing letters with noise. The result of this study is promising in terms of the fact that it is the first attempt to use SN P systems in pattern recognition after many theoretical advancements of SN P systems, SN P systems exhibit the feasibility for tackling pattern recognition problems.**

***Index Terms*—Bio-inspired computing, Membrane computing, Spiking neural P system, Learning, Letter classification**

I. Introduction

Bio-inspired computing, short for biologically inspired computing, is a major subfield of natural computation, whose aim is to abstract computing ideas from biological systems to construct efficiency computing models and algorithms. The abstract computing ideas include data structures, information encoding/decoding strategy, operations with data, ways to control operations, computing intelligence, etc. Membrane computing is a new branch of bio-inspired computing, which seeks to discover new computational models from the study of biological cells, particularly of the cellular membranes [1], [2]. The obtained models are distributed and parallel computing devices, usually called P systems.

In 2006, spiking neural P systems, namely SN P systems, were proposed by mimicking the way neurons communicate via electrical impulses (spikes) [3]. SN P systems are a class of neural-like models [2], which is a subject of investigation for developing powerful neural-like computing models.

Biological neurons' spiking is powerful in doing computation and communication with temporal information coding in spikes [4], [5]. A neural network can be achieved by connecting a number of neurons, among which information (encoded in form of spikes or real numbers) can be sent from one neuron to another [6], [7], [8], [9]. During the past decades, neural-like computing models have gained their popularity for their learning capability [10], [11], [12], [13], [14] and applications in solving realistic problems [15], [16], [17], [18], [19]. Spiking neural networks (SNNs) represent a class of the most bio-plausible candidate(s) in neural network models [20]. In addition to neuronal and synaptic state, SNNs incorporate the concept of time into their operating model. The neuron in the SNN cannot fire at each propagation cycle, but only when a membrane potential reaches a specific value. When a neuron fires, it generates a signal which travels to other neurons which, in turn, increase or decrease their potentials in accordance with this signal [5]. In order to increase the level of realism in a neural simulation, the spiking rule, denoted in form of production in grammar of formal languages, is used to describe the neuron's spiking behaviour in SN P systems, which determine the conditions of triggering spiking, the number of spikes consumed, and the number of spikes emitting to the neighboring neurons. The spikes from different neurons can be accumulated in the target neuron for further spiking [3].

It is formulated in [5], [20] that candidates in the third generation of neural neural networks have some common features.

- The concept of time is incorporated into neuron's spiking.
- Neurons do not fire at each propagation cycle, but rather fire only when a membrane potential or the stack of spikes reaches a specific value.
- Various coding methods exist for interpreting the outgoing spike train as a real-value number, either relying on the frequency of spikes, or the timing between spikes, to encode information.

In terms of features of models, SN P systems fall into the third generation of neural network models.

Researchers have made efforts leading to many significant contributions to SN P systems. Notably, SN P systems can generate and accept the sets of Turing computable natural numbers [3], generate the recursively enumerable languages

This work was supported by National Natural Science Foundation of China (61320106005, 61502535, 61772214 and 61873280), the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012), Key Research and Development Program of Shandong Province (2017GGX10147), and research project TIN2016-81079-R (AEI/FEDER, Spain-EU) and grant 2016-T2/TIC-2024 from Talento-Comunidad de Madrid, project TIN2016-81079-R (MINECO AEI/FEDER, Spain-EU) and and InGEMICS-CM project (B2017/BMD-3691, FSE/FEDER, Comunidad de Madrid-EU). *Asterisk indicates corresponding author.*

T. Song with the College of Computer and Communication Engineering, China University of Petroleum, Qingdao 266580, Shandong, China. L. Pan* and T. Wu are with the Key Laboratory of Image Processing and Intelligent Control of Education Ministry of China, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China; and School of Electric and Information Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, Henan, China (e-mail: lqpan@mail.hust.edu.cn). Pan Zheng is with Department of Accounting and Information Systems, University of Canterbury, Christchurch 8041, New Zealand. M.L.D. Wong is with School of Engineering & Physical Sciences, Heriot-Watt University (Malaysia), Kuala Lumpur, Malaysia. Alfonso Rodríguez-Patón is with Department of Artificial Intelligence, Faculty of Computer Science, Polytechnical University of Madrid, Campus de Montegancedo, Boadilla del Monte 28660, Madrid, Spain.

[21] and compute the sets of Turing computable functions [22]. Inspired by different biological phenomena and mathematical motivations, lots of variants of SN P systems have been proposed, such as SN P systems with anti-spikes [23], [24], asynchronous SN P systems [25], asynchronous SN P systems with local synchronization [26], SN P systems with weight [27], SN P systems with astrocyte [28], homogeneous SN P systems [29], [30], sequential SN P systems [31], SN P systems with rules on synapses [32], [33], [34]. For applications, SN P systems are used to design logic gates, logic circuits [35] and operating systems [36], perform basic arithmetic operations [37], solve combinatorial optimization problems [38], diagnose fault in electric power systems [39], and SN P system for image skeletonizing [40]. SN P systems with neuron budding and division and space-time trade-off strategy can theoretically solve computationally hard problems in a feasible (polynomial or linear) time [41], [42], [43].

The resource (here, in terms of the number of neurons) needed for constructing Turing universal computing neural-like computing has been heavily investigated. For conventional artificial neural networks, it was shown that 886 sigmoid function based processors are needed to achieve Turing universality [44], but 10 neurons (improved from 49 neurons [22]) are needed for SN P systems to achieve Turing universality [45]. The comparison result shows that SN P systems have a "desired" computational powerful while using less resource. Instead of using sigmoid function to imitate biological neuron's spiking in artificial neural networks, SN P systems use spiking rules, in form of formal grammar production, to describe the neuron's spiking behaviour. A neuron can contain multiple rules (describing the ability to select spiking conditions), and can send out different numbers of spikes by consuming different numbers of spikes. From the analysis above and known contributions to SN P systems, it is believed that SN P systems potentially offer better performance in doing computation.

During the past decades, neural-like computing models with learning strategies [4], [12], [17] have gained their popularity for their applications in pattern recognition [15], [16], [46], particularly for image classification and recognition [47], [48], [49]. Also, the heavily investigated deep learning methods have been applied to recognize specific objects [50], [51].

It is natural extension, but still remains an open research problem, to use SN P systems to do image recognition [52]. Recently, SNNs have been used to recognize English letters and symbols, and most of the characters were recognized uniquely in the sense that either a unique neuron fired or the firing times of the same neuron were different. But, Letters having the same number of pixels in there character representations may have non-unique representations [48]. It is of interests to investigate the power of recognizing letters by spiking neural network models. The number of spikes representing different letters may be non-unique, which sometimes bring extra difficulty in identifying the letters by spikes. So, the number of spikes is not sufficient to represent a letter, but the spiking frequency is considered to be useful. In SN P systems, besides spikes, spiking rules are also powerful, which can be used as a "selector" to control the spiking neuron and information emitted, which provides a natural way of dealing with the information encoded by spikes from different letters.

In this work, SN P systems, a new variant of SN P systems, namely SN P systems with learning functions, is introduced, and then a specific class of SN P systems with simple Hebbian learning functions are constructed and applied to recognize digital English letters. The main contribution of this research is that this is the first attempt to apply SN P systems in pattern recognition after many theoretical advancements of SN P systems. The results show that SN P systems are potentially powerful models for tackling pattern recognition problems.

II. Spiking Neural P Systems with Learning Function

In this section, we start by recalling some basic notions in formal language [2], which are helpful in understanding SN P systems, and then SN P system with learning functions is introduced. Given an alphabet V, V^* denotes the set of all finite strings of symbols from V. λ stands for empty string, and the set of all nonempty strings over V is denoted by V^+. When $V = \{a\}$, it is called a singleton, then $\{a\}^*$ and $\{a\}^+$ can be simply written as a^* and a^+, respectively.

A regular language is a formal language that can be expressed using a regular expression [55], [53], [54]. With each regular expression E, a regular language $L(E)$ is associated. The formal definition of regular language over an alphabet V is as follows, which can be referred to [55].

An *SN P system with learning functions* of degree $m \geq 1$ is a construct of the form

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn_0, f, I_{in}, I_{out}), \text{where}$$

- $O = \{a\}$ is a alphabet with single symbol, where a denotes a spike;
- $\sigma_i = (n_i, R_i)$, $1 \leq i \leq m$, are *neurons*, where
  1) n_i is the number of spikes initially contained in neuron σ_i when the system starts its computation;
  2) R_i is a finite set of rules of the following two forms, by which each neuron can process the information inside encoded in form of spikes:
     - Spiking rule: $E/a^c \rightarrow a; d$, where E is a regular expression over O, $c \geq 1$ and $d \geq 0$. (If $L(E) = a^c$, it can be simplified as $a^c \rightarrow a; d$; if $d = 0$, it can be simply as $E/a^c \rightarrow a$.) At certain moment t, if neuron σ_i contains k spikes and $a^k \in L(E)$, $k \geq c$, then rule $E/a^c \rightarrow a; d$ can be applied. The neuron fires consuming c spikes ($k - c$ spikes remaining in neuron σ_i) and sending one spike to each of its neighboring neurons after d time units, during which it cannot receive new spikes. This corresponds to the refractory period from neurobiology. In the step $t+d$, neuron σ_i becomes again open so that it can receive spikes and apply its rule in step $t + d + 1$.
     - Forgetting rule: $a^s \rightarrow \lambda$, for some $s \geq 1$, with restriction that $a^s \notin L(E)$ for any spiking rule $E/a^c \rightarrow a; d$ from R_i. The forgetting rule can

be applied only if the neuron contains exactly s spikes, by which s spikes are removed out of the neuron. In any neuron if a spiking rule is applicable, then no forgetting rule is enable to use, and vice versa.

- syn_t is the set of synapses at a computation step t, whose element is of the form (i, j, w_{ij}). It means there is a synapse (i, j) connecting neurons σ_i and σ_j and the weight on synapse (i, j) is $w_{ij} \in \mathbb{Z}^+$. (The function of the weights is to amplify the spikes (signal) passing along the associated synapses. Suppose that neuron σ_i emits one spike to neuron σ_j along synapse $(i, j, w_{ij}) \in syn_t$, neuron σ_j will receive $1 \times w_{ij}$ spikes.) The initial set of synapses syn_0 indicates the initial topological structure of the system, including the connection among neurons and the weights on the synapses.
- $I_{in}, I_{out} \subset \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ with $I_{in} \cap I_{out} = \emptyset$ indicates the sets of input and output neurons. The system reads input information from the environment in the form of spike trains through input neurons. Suppose at a step t, an input neuron begins to read spike trains. If the ith bit of the spike train is 1, then the input neuron receives one spike at step $t + i$; otherwise, no spike enters into the input neuron. For all input neurons, they can read spike trains from the environment in a parallel manner. The system emits spikes into the environment from the output neurons.
- Learning function f is used to rebuild, strengthen or weaken the connections among neurons during the computation. Mathematically, $syn_t = f(syn_{t-1})$, which means at step t, the synapse set syn_t can be obtained from syn_{t-1} with learning function f.

Note that, in each time unit, if a neuron σ_i can use one of its rules, then a rule from R_i must be used on the tick of the clock, for all neurons at the same time. Hence, the work of each neuron is sequential: only one rule can be applied in each time unit, but different neurons can operate in parallel.

The "state" of the system at a computation step is described by the number of spikes present in each neuron, the open-close status represented by the time delayed for spiking of the neurons, and the set of synapses at that moment. At any moment t, the configuration of the system is $\langle (q_1, d_1), (q_2, d_2), \ldots, (q_m, d_m), syn_t \rangle$, where $q_i, d_i \geq 0$ indicating neuron σ_i contains q_i spikes and after d_i steps it becomes open again. With this notation, the initial configuration of the system is $\langle (n_1, 0), (n_2, 0), \ldots, (n_m, 0), syn_0 \rangle$. It is allowed that different neurons hold different number of spikes initially. If a neuron initially has some spikes inside, when the computation of the system starts, the neuron can become active (fire) immediately without receiving any spikes from its neighboring neurons. Using the rules and learning function described above, one can define transitions among configurations. Any series of transitions starting from the initial configuration is called a computation. A computation is successful if it reaches a configuration, where no rule can be applied in any neuron (i.e., the SN P system has halted). The result of a computation is a vector recording the number of spikes sent to the environment by the output neurons, when the system halts.

In what follows, SN P systems with learning functions will be given graphically, which may be easier to understand than in a symbolic way. Rounded rectangles are used with the initial number of spikes and the set of rules inside to represent a neuron, and a directed graph to represent the structure of the system: the neurons are placed in the nodes of the graph and the edges represent the synapses; the input/output neurons have inputing/outgoing arrows, suggesting their communication with the environment.

III. SN P SYSTEMS WITH SIMPLE HEBBIAN LEARNING FUNCTIONS

A class of SN P systems with simple Hebbian learning functions are constructed to recognize English letters (in form of spike trains). Each system has two modules, which are Input and Recognize modules. At a computation step t, the set of synapses of the system is denoted by syn_t=$syn^t_{Input} \cup syn^t_{Rec}$, where syn^t_{Input} is the set of synapses of the Input module and syn^t_{Rec} is the set of synapses of the Recognize module. The Hebbian learning function f is defined as Equation 1, by which the synapse between the two neurons in the Recognise module will be incremented by 1 whenever the neuron at the start of the synapse fires.

With the Hebbian learning function defined in Equation 1, the weights of synapses in the Input module will keep fixed, that is, they cannot change during the computation. When a neuron in the Recognize module fires at certain moment, the weights on the synapses starting from it will be increased by one.

The Input module shown in Figure 1

The Input module consists of 72 neurons, whose function is to read spike trains of English letters from the environment into the system. (Each letter is represented by a spike train of length 35, which is to be discussed in detail in Subsection IV-A.) Suppose a spike train $v = v_1 v_2 \ldots v_{35}$ with $v_i \in \{0, 1\}$ is to be read. The process of reading spike train v by the Input module is as follows. Initially, all the neurons in the Input module have no spike inside, with the exception that start neuron σ_{start} has one spike. At step 1, neuron σ_{start} fires by using spiking rule $a \to a$, sending one spike to neuron σ_{S_1}. At the same moment, input neuron σ_{Input} read the first bit of v (v_1) from the environment.

- If $v_1 = 1$, neuron σ_{Input} receives one spike from the environment at step 1. With this spike inside, neuron σ_{Input} fires immediately by using spiking rule $a \to a$, sending one spike to each of neurons $\sigma_{I_i}, i = 1, 2, \ldots, 35$. Meanwhile, presynaptic neuron σ_{S_1} fires sending one spike to each of neurons σ_{I_1} and σ_{S_2}. Having two spikes inside, neuron σ_{I_1} will fire at step 2, and one spike is sent to neuron σ_{R_1}. The spike in neurons $\sigma_{I_i}, i = 2, 3 \ldots, 35$ will be removed by using forgetting rule $a \to \lambda$. In this case, neuron σ_{R_1} receives one spike, representing $v_1 = 1$.
- If $v_1 = 0$, neuron σ_{Input} receives no spike from the environment at step 1, and keeps inactive. At step 2, the spike in presynaptic neuron σ_{I_1} (received from neuron

$$syn_{t+1} = f(syn_t) \begin{cases} (i,j,w_{ij}), & \text{if } (i,j,w_{ij}) \in syn^t_{Input}; \\ (i,j,w_{ij}), & \text{if } (i,j,w_{ij}) \in syn^t_{Rec} \text{ and } \sigma_i \text{ doesn't fire at step } t; \\ (i,j,w_{ij}+1), & \text{if } (i,j,w_{ij}) \in syn^t_{Rec} \text{ and } \sigma_i \text{ fires at step } t. \end{cases} \quad (1)$$

σ_{S_1}) will be removed. In this case, no spike is emitted to dendrite neuron σ_{R_1}. Neuron σ_{R_1} receives no spike, representing $v_1 = 0$.

Using the Input module, spike train v can read one by one bit and store one spike (or no spike) in neuron σ_{R_i} with $i = 1, 2, \ldots, 35$. Specifically, if $v_i = 1$, then neuron σ_{R_i} will store one spike inside; if $v_i = 0$, then neuron σ_{R_i} will receive no spike. When the Input module finishes reading a spike train, a spike goes back to the neuron σ_{start} by passing along the path of neurons $\sigma_{S_1}, \sigma_{S_2}, \ldots, \sigma_{S_{35}}$, which indicates Input module returns to the initial configuration and is ready to read the next spike train. In this way, the Input module can read multiple spike trains one by one, and store a specific number of spikes in neurons σ_{R_i} with $i \in \{1, 2, \ldots, 35\}$. The initial values of the weights on the synapse in the Input module are 1, and will not change during the process of reading spike trains (with learning function f defined in Equation 1.)

The Recognize module shown in Figure 2

The topological structure of the Recognize module is obtained with the following considerations.

- Since every English letter is represented by 7×5 pixels and each pixel is associated with a neuron, 35 neurons in total, labeled with $\sigma_{R_i}, i = 1, 2, \ldots, 35$are needed in Recognize module. Four output dendrite neurons labelled σ_{output_j}, $j = 1, 2, 3, 4$ collect the spikes emitting from the system of four directions, up, down, left and right. Thus, the Recognize module has 39 neurons.
- The spiking rule in each neuron is used to process the information inside the neuron encoded by the number of spikes. The spiking rule in any neuron is of the form $a^*/a \to a$. The rule has the function of slowing the passing speed of the spikes but will not lose any information. For example, if a neuron receives k spikes at some moment, then as described in basic notions of SN P systems, the spiking rule $a^*/a \to a$ will be applied for k times in k steps. In each time of spiking, the neuron emits one spike to its neighboring neurons, thus emitting in total k spikes to each of its neighboring neurons.
- The connections among the neurons are based on the idea of transmitting the input information from the innermost layer to the outermost layer. With this consideration, a three layer topological structure, including the innermost layer, middle layer and the outermost layer is obtained. Each layer needs to emit spikes to four directions, so the neurons in each layer are divided into four parts. The neurons in the same direction have a full connection to the neurons in its upper layer. Specifically, neurons in the middle layer have full connections to the neurons in the same direction of the outermost layer, and neurons in the innermost layer have full connections to each of the neurons in the middle layer. But, no connection exits among the neurons in the middle and the outermost layers in different directions. In this way, the topological structure shown in Figure 2 is obtained.

In general, the architecture of the SN P system follows the observation that biological spiking neural networks tend to operate from the innermost to the outermost neurons. The architecture of the SN P system is inspired from the biological motivation of liquid state machine (LSM), which is a particular kind of spiking neural network. An LSM consists of a large collection of units (called nodes, or neurons). Each node receives spikes from external sources (the inputs) as well as from neighboring nodes. The word "liquid" comes from the analogy of information transmitting in the network like dropping a stone into a still body of water or other liquid. The falling stone will generate ripples in the liquid. The input (motion of the falling stone) has been converted into a spatio-temporal pattern of liquid displacement (ripples).

In our SN P system, each neuron in Recognize module can receive spikes from the neuron in input module, as well as can receive spikes from its neighboring neurons. The neurons in Recognize module are arranged in three layers, which are the innermost, middle and outermost layers. Such architecture is the analogy of information transmitting in neural networks like dropping a stone into liquid. The spikes emitting from the four edges by four outputting neurons are collected as the computational result of the system.

In the system, neurons $\sigma_{R_{13}}$, $\sigma_{R_{18}}$ and $\sigma_{R_{23}}$ in red are three central neurons in the innermost layer, having full connections with the neurons in the middle layer ($\sigma_{R_i}, i = 7, 8, 9, 12, 14, 17, 19, 22, 24, 27, 28, 29$). Neurons in the middle and outermost layers, are divided into four groups, colored in Gray, Black, Green and Yellow. Each neuron of the outer layer connects to 5 neurons in the outmost layer (marked in the same color). There are 20 neurons in the outmost layer, in which 5 neurons are labeled with Gray, Black, Green or Yellow, respectively. Four output neurons are used to read information of a letter of pixel array from 4-directions, left, right, up and down.

The neuron in the Recognize module works as follows. If neuron σ_{R_i} holds k spikes (these spikes come from neuron σ_{I_i} in the Input module), then spiking rule $a^*/a \to a$ can be used, consuming one spike and emitting one spike to its neighboring neurons. The number of spikes in neuron σ_{R_i} becomes $k-1$. Neuron σ_{R_i} fires for the second time in the next step. In this way, neuron σ_{R_i} will fire for k times in k steps, emitting k spikes in total to its neighboring neurons, one spike in each step.

The weights of the synapses in the Recognize module are initially associated with value 1. During the computation, the weights of the synapses in the Recognize module are updated by learning function f. Specifically, suppose at a step t, it has $(R_i, R_j, w_{R_i,R_j}) \in syn^t_{Rec}$. If neuron σ_{R_i} fires at step t sending one spike to neuron σ_{R_j}, the weight of the synapse

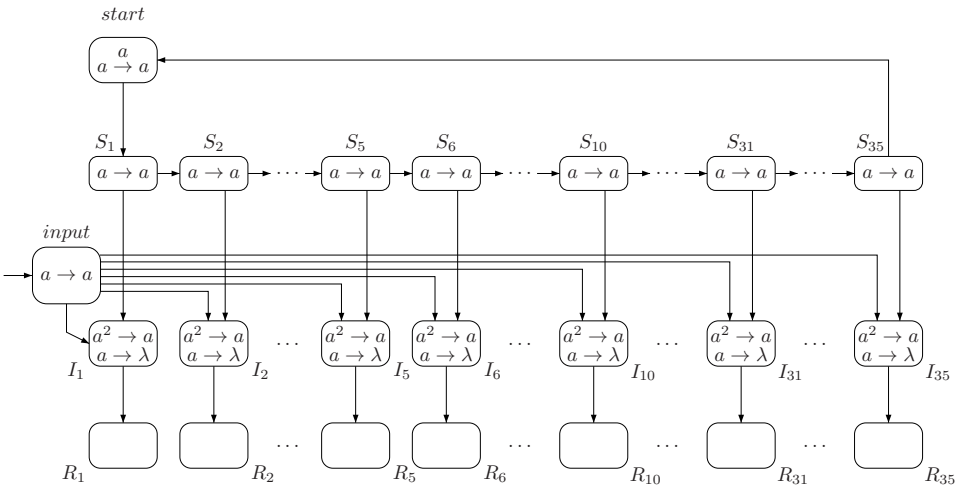

Fig. 1. The structure of the Input module

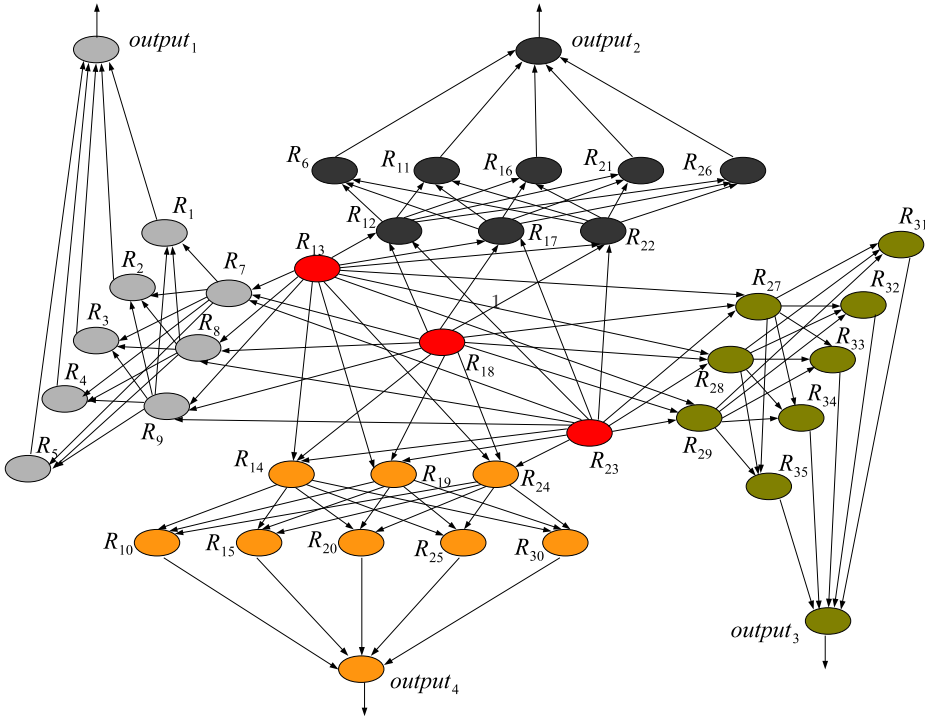

Fig. 2. The structure of the Recognize module

will be increased by one; otherwise, the weight of the synapses keeps unchanged. The output neurons $\sigma_{output_j}, j = 1, 2, 3, 4$ have spiking rule $a^*/a \to a$, and can emit its spikes to the environment. The number of spikes emitted by each output neuron is counted, and then a 4-dimensional vector can be obtained.

IV. Recognizing English Letters by SN P System with Hebbian Learning Function

This section starts by introducing the way to encode English letters by spike trains, particularly in three levels of noise, and then SN P systems with simple Hebbian learning functions are constructed to recognize the letters.

A. Representing English letters by spike trains

Each English letter is represented by a black-white image in the size of 7×5 pixels. The representation of letter "B" in Calibri, for example, is shown in Figure 3.

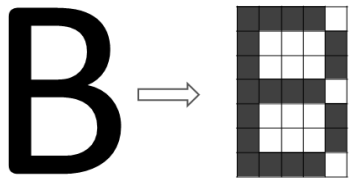

Fig. 3. Representing letter "B" in Calibri by a 7×5 pixel array

The representations of English letters "A" to "Z" by black-white images in the size of 7×5 pixels are shown in Figure 4.

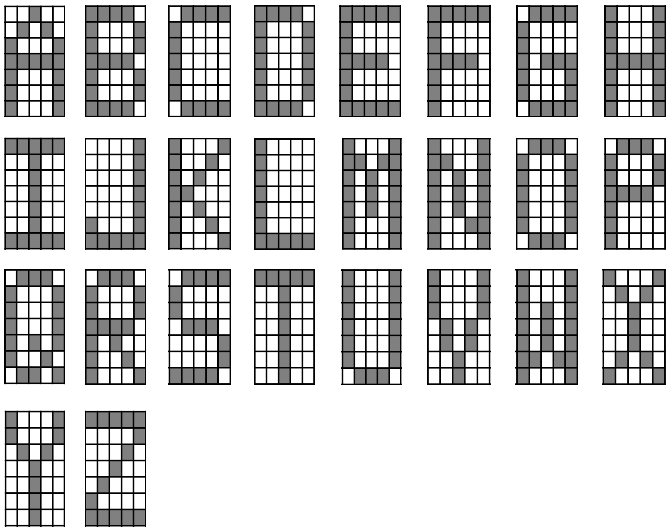

Fig. 4. Representing of all English letters by 7×5 pixel arrays

Each black and white image can be represented by a 7×5 binary matrix, where each pixels is associated with a corresponding element in the matrix. Specifically, the pixel at the ith row and jth column corresponds to the element at the ith row and the j column of the matrix; if the color of a pixel is black, its value is 1; otherwise, its value is 0. With this conversion, it is easy to obtain a binary string (spike train) by concatenating the rows of the matrix one by one from top to bottom. A letter can be represented by a spike train. Figure 5 shows an example of converting letter "B" to a spike train.

This type of spike train is considered as the standard form of a spike train of a letter that contains no noise. Spike trains of letters with noise are set by flipping some bits of standard letter spike trains, i.e., changing a given number of bits from 0 to 1 or from 1 to 0. Test cases with three different levels of noise are generated randomly. The three levels of noise are specified as follows.

- Low noise case: 0–3 bits are flipped in a spike train of the standard form.
- Medium noise case: 4–6 bits are flipped in a spike train of the standard form.
- High noise case: 7 bits are flipped in a spike train of the standard form.

B. The general process of recognizing English letters

For an English letter, an SN P system with Hebbian learning function is associated. The process of recognizing an English letter (represented by spike train) has four stages: Reading Stage, Training Stage, Generating Standard Output and Recognizing Unknown Letters.

Reading Stage

For an English letter, first, a set of spike trains associated with the letter at a certain level of noise is generated. The spike trains are generated by flipping randomly a certain number of bits from a spike train of the standard letter form. The number of bits to be flipped is determined by the level of noise. The generated spike trains are to be read into the system through the Input module (the input neuron) from the environment.

Training Stage

The input neuron σ_{input} can read spike trains one by one, sending spikes to neuron σ_{I_i}. When neuron σ_{I_i} receives spikes from neuron σ_{S_i}, it can fire by using spiking rule $a^*/a \to a$, emitting spikes to neuron σ_{R_i}. When neurons σ_{R_i} has spikes, it fires, emitting spikes to its neighbor neurons. During this process, the weights of the synapses between each pair of neurons will be updated by learning function f. Output neurons can emit spikes into the environment, but will be ignored. When the system halts, it forms a specific topological structure by processing the input information. For 26 English letters, 26 "trained" SN P systems will be obtained. The obtained "trained systems" have their topological structure fixed and will be used to recognize unknown letters.

Note that, it is necessary to impose a bound to the weights on the synapses with a Hebbian learning strategy in neural network models. Since the size of the training set of each letter used in this work is not more than a finite number m, the weight on each synapse will not be bigger than m. It is considered as an inherent bound for the synapse from the working principle of SN P systems. In the worst case, if the weight on synapses are m, it will produce outputs in form of $O(m^3)$ spikes (when spikes pass from the core layer to the output neurons, they can be amplified for three times, each time amplified by m). Large output values will be simplified using the variance to generate the difference between known and unknown letters.

Generating Standard Output

For an English letter, a spike train from the set of spike trains used in Reading Stage is randomly chosen and read into the "trained system". With the spike train, each "trained system" starts its computation, and when the system halts, a 4-dimensional vector $(stan_1, stan_2, stan_3, stan_4)$ is generated to record the numbers of spikes emitted by the four output neurons $\sigma_{output_1}, \sigma_{output_2}, \sigma_{output_3}, \sigma_{output_4}$. The vector is called the standard outputting vector of the letter. Note that learning function f is only used to update the weights of synapses in the above training stage.

Recognizing Unknown Letters

The spike train of an unknown English letter is read into each of the "trained systems". With the spike train, each "trained system" starts it computation, and when it halts, a 4-dimensional vector $(out_1, out_2, out_3, out_4)$ is generated recording the numbers of spikes emitted by the four output neurons. It calculates the variance between the outputting vector of the unknown letter and standard outputting vectors. Specifically, the variance is calculated by

$$var = \sqrt{\sum_{i=1}^{4}(out_i - stan_i)^2},$$

where $(out_1, out_2, out_3, out_4)$ is the outputting vector of the unknown letter and $(stan_1, stan_2, stan_3, stan_4)$ is the standard outputting vector of a certain letter. The one with the

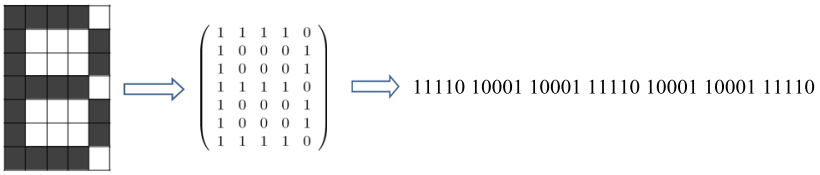

Fig. 5. Representing letter "B" by a spike train

lowest value of the variance is considered to represent the unknown letter.

V. Experimental Results

In this section, experimental results of recognizing letters by SN P systems with simple Hebbian learning function are shown. Furthermore, comparison with BP neural networks in four flavors (gradient descent with momentum, gradient descent with adaptive learning, Levenberg-Marquardt and scaled conjugate gradient), probabilistic neural networks (PNN) and spiking neural networks (SNN) are done for test cases with different levels of noise [56], [57].

The reason why BP neural networks, PNN, and SNN are chosen for comparison is that they are well known and classical artificial neural network models, which have been deeply investigated and widely applied in solving real-world problems. The purpose of comparing SN P systems with BP neural networks, PNN, and SNN is not to show SN P systems can can replace or improve upon the performance of existing significant neural-like models, but to indicate that SN P systems can be used in letter recognition with acceptable performance relative to these other neural-like computing models. These results may provide some potential applications or hints to solve real-life problems in some scenarios by SN P systems.

The The recognition accuracy rate is mathematically defined as $\sum_{i=1}^{m} T_i$, where if the letter is correctly recognized by its associated neural network and also by another n networks, then $T_i = \frac{1}{n+1}$; otherwise, $T_i = 0$, and m is the number trials.

A. Performances of SN P systems with Hebbian learning functions

For every English letter, the size of its training set is 100, that is, 100 spike trains of the letter are input to train the system. Spike trains in different levels of noise are generated randomly by flipping d bits of the standard ones, where $d = 0, 1, 2, 3, \ldots, 7$. The number of bit to be flipped is based on the assumption that less than 20% of the bits is incorrect in different level of noise. There are in total 5×7 bits, whose 20% are 7 bits at most for flipping. And it is find that there are some letters with less than 4 bit in difference, so if the number of flipping bits increased, it would be very hard to recognize the letters correctly. Given a letter and value of d, recognition experiments are carried out for 100 trials. The actual implementation of the SN P systems is done with C++.

Experimental results show that SN P systems with Hebbian learning functions generally perform well in recognizing English letters. In the test case with no noise, the average recognition accuracy rate is 98.76%. Notably, the recognition accuracy rate for 13 of the 26 English letters is 100% (corresponding to the case of $d = 0$ in Table I).

In case $d = 0$, most of the letters can be recognized by SN P system with accuracy rate above 97%, except for letters "H", "N" and "Q" having accuracy rates 96%. In case $d = 1$, five letters have recognition accuracy rate of 100%, which are "F", "R", "X", "Y" and "Z". In case $d = 4$, SN P system recognizes letters "D", "M", "N" and "O" with accuracy rate below 80%, but can recognize another 11 letters, "A", "B", "C", "E", "F", "K", "L", "Q", "S", "T" and "W", with accuracy rates no less than 90%. In the test cases with low level of noise, SN P system recognizes 21 letters, "A", "C", "D", "E", "F", "I", "J", "K", "L", "O", "P", "Q", "R", "S", "T", "V", "W", "X", "Y", "Z" and "W", with accuracy rates above 95%. In the high noise case, recognition accuracy rates for 13 letters, "A", "F", "H", "I", "J", "K", "L", "Q", "S", "T", "U", "W" and "X", are above 80%, performs extremely well in recognizing letters "A", "L", "J" and "S", which achieve accuracy rates above 85%.

Referring to the pixel arrays shown in Figure 4, the differences occur at only two pixels for letters "M" and "N", hence their spike trains have only two distinguishing bits. If the two different bits are exactly chosen to be flipped in noisy cases, then it is typically hard to recognize the letter correctly. This might be the reason why it is difficult to distinguish "M" from "N" when the value of d becomes large. Similar case happens to letters "D" and "O", whose accuracy rate are less than or equal to 65% with $d = 7$, reducing greatly with the increment of d.

B. Comparison results to BP neural networks

For comparison, BP neural network experiments to recognize English letters in form of spike trains with low, medium and high noise are implemented.

For each letter, a BP neural network is built using method from [56], [57] to recognize it. It is suggested in [57] that the number of input neurons should be related to the image to be recognized. We use a mapping of 1 : 1 pixel to neuron, so 35 input neurons are needed to read 35 binary bits of a letter. There are 20 neurons in hidden layer, which is obtained by experience from [56]. The number of neurons in the output layer is determined by the number of classes to be classified. For each letter, the associated BP neural network needs to

TABLE I
THE ACCURACY RATE OF RECOGNITION FOR ENGLISH LETTERS WITH 0–7 BIT(S) FLIPPING IN THE SPIKE TRAINS

Character \ d	0	1	2	3	4	5	6	7
A	100%	98%	96%	95%	97%	94%	93%	91%
B	100%	96%	96%	93%	91%	90%	88%	76%
C	100%	99%	97%	95%	91%	88%	87%	75%
D	100%	98%	97%	96%	79%	76%	69%	63%
E	100%	99%	98%	97%	92%	89%	86%	72%
F	100%	100%	99%	96%	90%	84%	82%	83%
G	98%	96%	95%	93%	84%	76%	69%	71%
H	96%	92%	92%	91%	89%	85%	79%	80%
I	100%	99%	98%	98%	89%	84%	84%	83%
J	98%	98%	98%	97%	89%	86%	85%	86%
K	100%	98%	99%	98%	93%	89%	86%	82%
L	99%	99%	99%	99%	95%	96%	91%	88%
M	97%	98%	95%	91%	79%	76%	68%	71%
N	96%	98%	94%	92%	78%	71%	69%	72%
O	97%	98%	95%	95%	78%	78%	76%	65%
P	98%	97%	97%	98%	89%	82%	79%	74%
Q	96%	97%	97%	96%	91%	86%	82%	82%
R	100%	100%	100%	98%	89%	86%	79%	76%
S	99%	98%	98%	98%	91%	89%	87%	85%
T	97%	99%	99%	98%	93%	86%	81%	81%
U	98%	98%	94%	92%	81%	76%	79%	82%
V	99%	98%	95%	95%	84%	80%	76%	73%
W	100%	99%	99%	98%	93%	85%	80%	82%
X	100%	100%	99%	99%	86%	86%	86%	82%
Y	100%	100%	99%	98%	85%	81%	74%	73%
Z	100%	100%	100%	99%	84%	80%	75%	74%

output the recognition result, whether positive or negative. The result is represented by a two-dimensional vector $[out1; out2]$ with $out1, out2 \in \{0, 1\}$, where the positive one is denoted by $[1; 0]$, and negative one is $[0; 1]$.

The training process starts by reading input samples, and then adjusts the weights and biases of the networks with purpose to make the output and expected output to be as close as possible. When the sum of the square error is less than the specified error, the training process is completed.

The specific steps are as follows:

Step 1. Initialization, the weights and thresholds are initialized randomly;
Step 2. The output of each unit in hidden layer and output layer is calculated by the given input and output mode;
Step 3. Calculate the new weights and threshold;
Step 4. Select the next input mode and return to Step 2, repeatedly training, until the output error of the network achieves the required value, then end the training.

The size of the training set for the BP neural network is 50, where 20 spike trains are used for learning and 30 spike trains are used to verify the model.

Four learning strategies, gradient descent with adaptive learning rate, gradient descent with momentum backpropagation, Levenberg-Marquardt and scaled conjugate gradient method are chosen to train BP neural networks in recognizing English letters. We briefly recall the basic notions of the four training strategies, and for details one can refer to the corresponding references.

– In gradient descent with adaptive learning rate, backpropagation is used to calculate derivatives of performance $perf$ with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent: $dX = lr * dperf/dX$, see [58].
– In gradient descent with momentum back-propagation, back-propagation is used to calculate derivatives of performance $perf$ with respect to the weight and bias variables X. Each variable is adjusted according to gradient descent with momentum, $dX = mc * dXprev + lr * (1 - mc) * dperf/dX$. More details can be found in [58].
– With the Levenberg-Marquardt training strategy, a network training function updates weight and bias values according to Levenberg-Marquardt optimization [58]. It is fast and usually used as a first-choice supervised algorithm.
– The scaled conjugate gradient method is used to train any network as long as its weight, net input, and transfer functions have derivative functions. Backpropagation is used to calculate derivatives of performance with respect to the weight and bias variables based on conjugate directions [59].

For every letter in the test cases with low, medium and high noise, its data experiment is repeated for 100 times. The simulation is implemented with Matlab BP-Neural Network Tool, where the values of involved parameters are shown in Table II.

The meaning of the parameters involved in the BP neural network for recognizing English letters in form of spike trains is as follows.

– Max iterations: the maximal number of training steps; if the training process reaches the maximal number of steps, the training will stop.
– Mean square error: the square of the difference between

the real output and target output; if this value is less the initially set one, then the training of BP neural network is stopped.
– Learning step: also known as learning speed, which means the unit of variation of weights in each time update.
– Step length: after how many steps it needs to show the variation on curve.

The value of parameters of Mean square error, Learning step, Step length are obtained with the empirical formulae from from [57].

TABLE II
VALUES OF PARAMETERS IN BP NEURAL NETWORKS

Max iterations	Mean square error	Learning step	Step length
10000	0.002	0.05	100

The comparisons of accuracy rates for BP neural networks and SN P systems with test cases in low, medium and high noise are shown in Figures 6, 7 and 8. In the figures, the BP neural network with gradient descent with adaptive learning rate is labelled by da, gradient descent with momentum backpropagation is labelled with dm, Levenberg-Marquardt method is labelled with lm, and scaled conjugate gradient method is labelled by scg, respectively.

Experimental results in the test case of low noise by BP neural networks are shown in Figure 6. It is found that the BP neural network with scaled conjugate gradient (scg) performs a little better than than the other three training strategies. The BP neural network with scaled conjugate gradient method can recognize 8 English letters with accuracy rates above 90%, 18 English letters with accuracy rates above 80%, and the average recognition accuracy rate is 84.78%.

In the test case of medium noise (shown in Figure 7), the BP neural network with scaled conjugate gradient method performs better than the other three training strategies, but the average accuracy rate of recognition reduces to 43.01%. The BP neural network with scaled conjugate gradient method recognize 2 letters with accuracy rates above 60%, 17 letters with accuracy rates above 40% and 7 letters with accuracy rates below 40%. The average accuracy rate of recognition of SN P system is 83.89%, which is typically better than BP neural networks. SN P system can recognize 2 letters with accuracy rate above 90% and 20 letters with accuracy rate of recognition above 80%. Moreover, the recognition accuracy rates for letters "M" and "N" are the lowest, which are 74.3% and 74.23%, respectively.

In the test case with high noise, the average recognition accuracy rate by BP neural networks is further reduced to 30.94%, while the average accuracy rate by SN P system is 77.92%. SN P system can recognize 11 letters with accuracy rates above 80%, and 22 letters with accuracy rates above 70%. The two letters with the lowest accuracy rates are "D" and "O", whose accuracy rates are 63% and 65%, respectively.

C. Comparison results to probabilistic neural networks

For the probabilistic neural network experiments, every letter, in the form of a spike train, is associated with a probabilistic neural network (PNN).

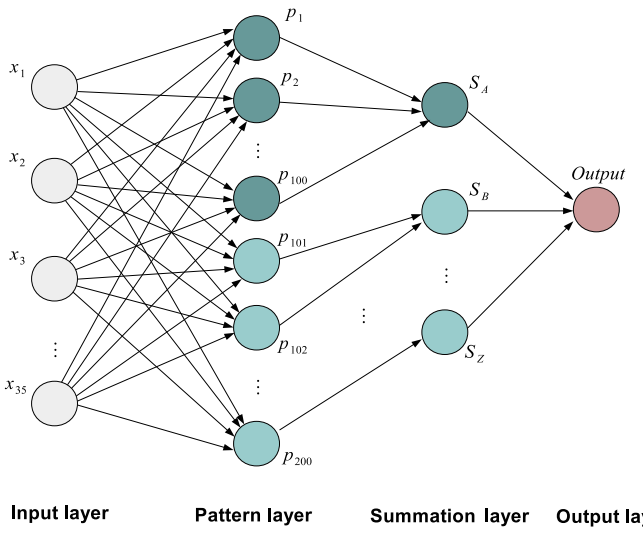

Fig. 9. The topological structure of the PNN used to recognize letter "A"

The topological structure of PNN is obtained by the principle for building PNN to solve pattern classification problems from [60]. There are four layers, which are Input layer, Pattern layer, Summation layer, and Output layer. Input layer has 35 neurons, each of which reads one bit of the spike train. Each neuron in Input layer has a connection to the neurons in Pattern layer. It is suggested in [60] that "A PNN consists of a node in layer one for each of the N training samples." We use here 200 training samples, so Pattern layer contains 200 neurons. There are 26 neurons labeled with $S_A, S_B, \ldots, S_Z$ in Summation layer, since in total 26 letters need to be classified. The top 100 neurons in Pattern layer have connections to each neuron S_α with $\alpha \in \{A, B, \ldots, Z\}$.

For the remaining 100 neurons in the Pattern layer, each four successive neurons are selected to connect each neuron $S_\beta, \beta \in \{A, B, \ldots, Z\} - \alpha$ in alphabetic order respectively. In the Output layer, there is an output neuron to record the result by 0 (negative) or 1 (positive). For example, the topological structure of PNN for recognizing letter "A" is shown in Figure 9.

The values of Max iterations, Mean square error, Learning step, Step length, and Spread are set by empirical formula introduced in [60]. We use function `newpnn` in Matlab probabilistic neural networks Tool-Box to train the probabilistic neural networks, which was developed in [61].

In the data experiments, the PNN is tested to recognize English letters in low, middle, and high noise cases. The size of the training set is 100. For any letter in the low, medium, or high noise cases, recognition experiments are repeated 100 times. The simulation is implemented by using Matlab probabilistic neural networks Tool, where the values of parameters are set by empirical formula from [60], which are shown in Table III.

Comparison results with the PNN in the low, medium, and high noise cases are shown in Figures 10, 11 and 12. The parameters in Table III have the same meaning as in Table II.

In the test case with low noise, the average recognition

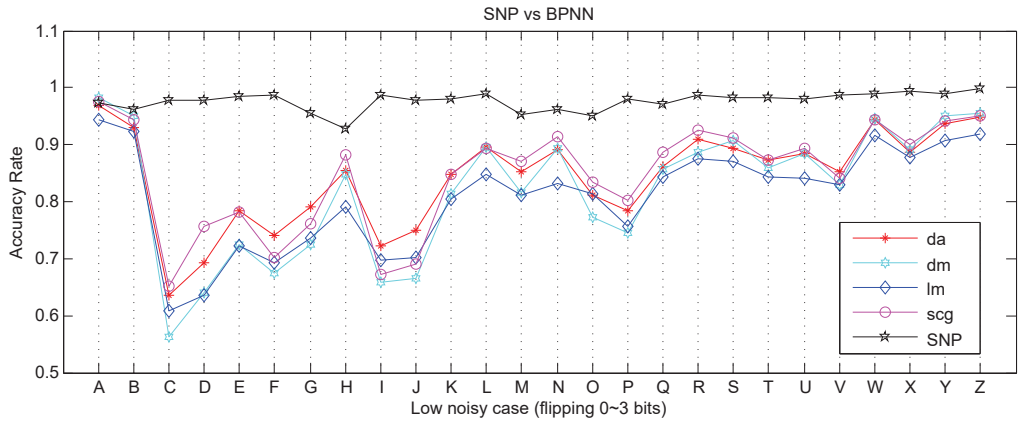

Fig. 6. Comparison results of BP neural networks in the test case with low noise

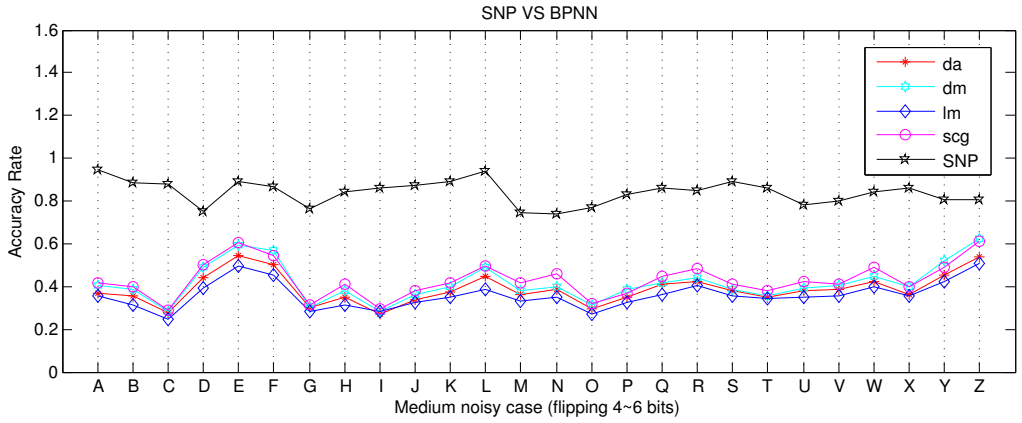

Fig. 7. Comparison results of BP neural networks in edium noise

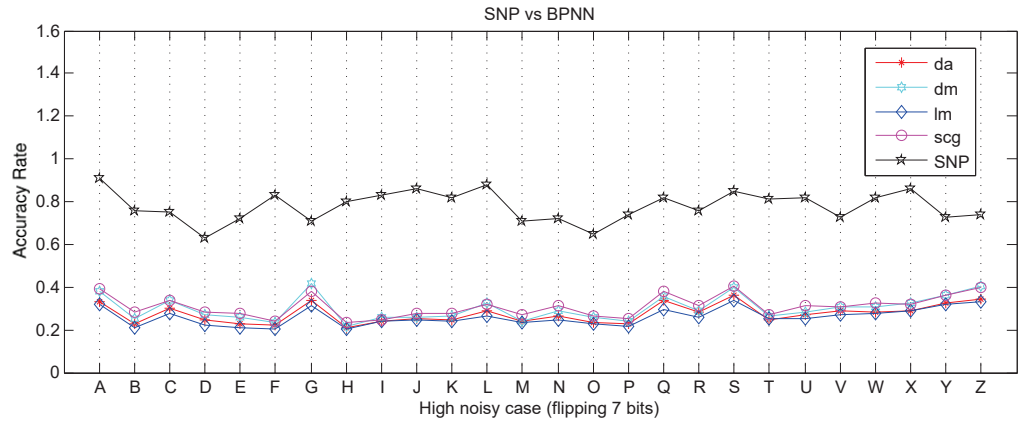

Fig. 8. Comparison results of BP neural networks in the test case with high noise

TABLE III
VALUES OF PARAMETERS IN PROBABILISTIC NEURAL NETWORKS

Max iterations	Mean square error	Learning step	Step length	Spread	Number of neurons
10000	0.002	0.05	100	1.5	106

accuracy rate of SN P systems is 97.64%, which is better than PNN with accuracy rate 86.62%. SN P system performs better in recognizing 21 letters, but PNN performs better than SN P systems in recognizing letters “A”, “K”, “Q”, “S”, and “Y”.

It is found that the PNN performs slightly (about 2%) better on accuracy rate than the SN P systems in recognizing letters “Q”, “S”, “Y” and “Z”. Meanwhile, in recognizing letters “K”, “L” and “X”, SN P systems can achieve a little higher (on average 1.5% higher) accuracy rate than PNN.

In the test case with medium noise, SN P systems have

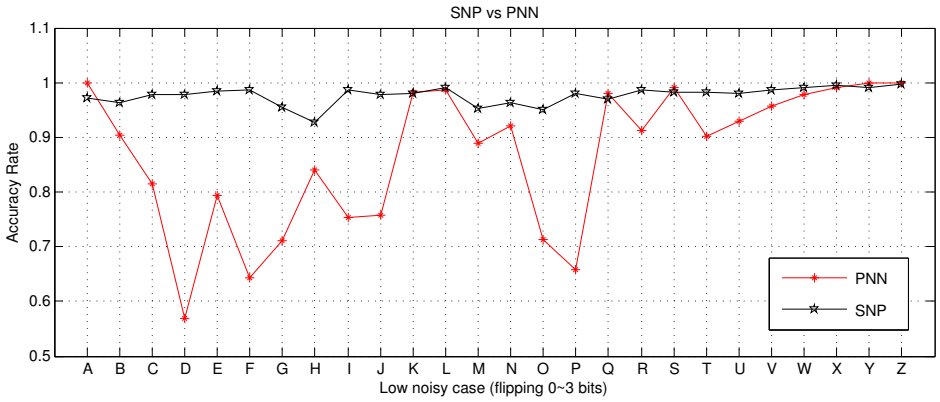

Fig. 10. Comparison results of probabilistic neural networks in the test case with low noise

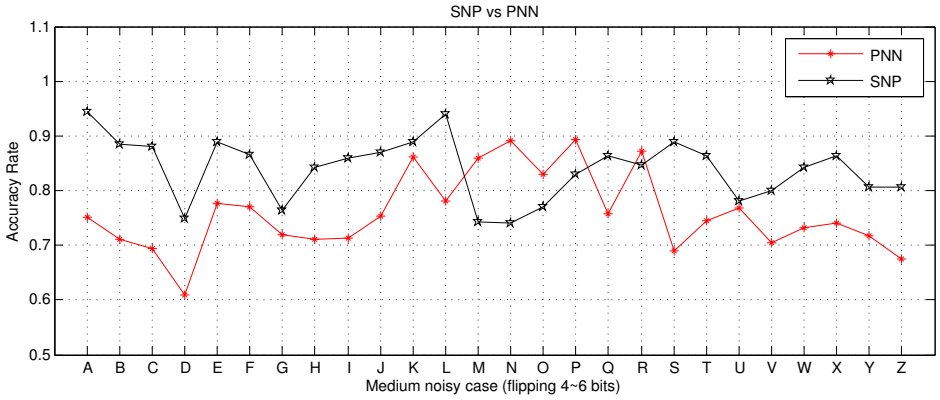

Fig. 11. Comparison results of probabilistic neural networks in the test case with medium noise

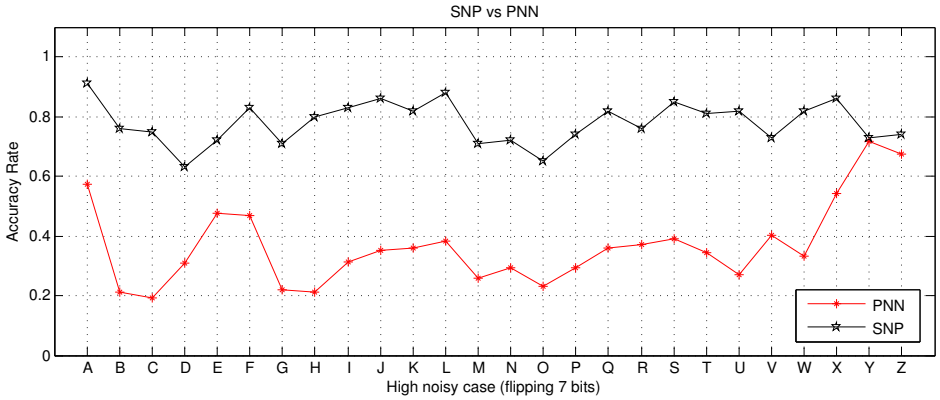

Fig. 12. Comparison results of probabilistic neural networks in the test case with high noise

average accuracy rate 83.89%, while average accuracy rate of PNN reduces to 75.86%. SN P system performs better than PNN in recognizing 21 letters, except for letters "M", "N", "O", "P" and "R". In the test case with high noise, SN P systems have average accuracy rate 77.92%, which is better than PNN, whose average accuracy rate reduces quickly to 36.72%.

D. Comparison results to spiking neural networks

In this subsection, comparison results with spiking neural network (SNN) are given. The SNN model is from [63] with spike time dependent plasticity (STDP). The main idea of STDP is that when the stimulus (in form of a constant current) is presented as input, the output should be in the form of a series of spikes, which represents a similar information processing strategy to spiking rule $a^*/a \to a$ in SN P system. This is the reason why such a model is selected to compare with SN P system for recognizing English letters.

To make the paper self-contained, the SNN model from [63] are briefly recalled, but for more details one can refer to [63], [65], [64].

Active-Dendrites-and-Dynamic-Synapses (ADDS) neuron proposed in [64] is used here, which is defined as follows. The neuron can receive input via spike(s) through a set of

its presynaptic neurons. For any synapse i with strength w^i (denoted by weight w^i) to an active dendrite, the total post-synaptic current I_d^i is defined by

$$\tau_d^i \frac{dI_d^i(t)}{dt} = -I_d^i(t) + R_d^i w^i \delta(t - t_f^i),$$

where t_f^i is the set of pre-synaptic spike times filtered as Dirac δ pulses. The time constant τ_d^i and resistance R_d^i define the active properties of the artificial dendrite as a function of the synaptic strength, and are defined as

$$\tau_d^i = \tau_{max} - |w^i|(T_{max} - T_{min}), |w^i| \leq 1.$$

It is not hard to see from the above equation that for high weights, τ_d^i is closer to T_{min}, while for low weights, τ_d^i is closer to T_{max}. Thus, as the time constant is low for stronger synapses, we have an earlier and steeper increase of the soma potential as compared to weaker synapses.

The resistance R_d^i is given by

$$R_d^i = \frac{\tau_d^i \theta}{R_m} (\frac{\tau_m}{\tau_d^i})^{\frac{\tau_m}{\tau_m - \tau_d^i}},$$

where θ is the neuron's firing threshold, R_m is the somatic resistance, and τ_m is the soma time constant. The above equation for R_d^i ensures that the maximum value of the membrane potential change is proportional to the neuron's firing threshold θ.

The other influence to an output neuron is from the somatic synapses feeding directly or close to the soma. These lateral or inhibitory connections enforce the winner-take-all mechanism as the synaptic activity of the neurons which are not the first to spike is inhibited. The equation governing the post-synaptic current is

$$\tau_i \frac{dI_i(t)}{dt} = -I_i(t) + \sum_i w^i \delta(t - t_f^i).$$

Combining the contributions from the dendritic connections and the synapses feeding directly to the soma, the equation obtained for the total soma membrane potential u_m is

$$\tau_m \frac{du_m(t)}{dt} = -u_m(t) + R_m(I_d(t) + I_m(t)),$$

where $I_d(t) = \sum_i I_d^i(t)$ is the total dendritic current, τ_m is the soma time constant, and R_m is the somatic resistance.

When the membrane potential reaches the threshold value θ, it produces a spike, and the membrane potential is reset to a value $u_{reset} = -1mV$. After this event, the membrane potential recovers to the resting potential value.

The general STDP learning rule is as follows.

$$\Delta w = \begin{cases} A + e^{\frac{\Delta t}{\tau^+}}, & \text{if } \Delta t < 0; \\ A - e^{\frac{\Delta t}{\tau^+}}, & \text{if } \Delta t > 0; \end{cases}$$

where $\Delta t = t_{pre} - t_{post}$. The weights can be updated by

$$w_{new} = \begin{cases} w_{old} + \eta \Delta w(w_{max} - w_{old}), & \text{if } \Delta w \geq 0; \\ w_{old} + \eta \Delta w(w_{old} - w_{min}), & \text{if } \Delta w < 0; \end{cases}$$

where η is the learning rate. For excitatory synapses, $w_{min} = 0$ and $w_{max} = 1$, whereas for inhibitory synapses, $w_{min} = -1$ and $w_{max} = 0$. If there is no pre-synaptic spike, the weight decays with a rate η_{decay}.

The SNN associated with each letter has two layers: input layer and output layer. Input layer has 35 neurons, which is equal to the number of pixels in the image. The number of output neurons is the number of characters to be trained, that is, 26 output neurons. Input layer has simple leaky integrate and fire neurons, which receives constant or zero input current, corresponding to "on" or "off" state to the input pixels. Output layer consists of active dendrite neurons, each of which is connected to all of the neurons in the previous layer. As well, every output neuron is connected to the other output neurons via inhibitory lateral connections, reflecting the competition among the output neurons.

The values of parameters in SNN are shown in Table IV The size of the training set is 100 for each letter, and the simulation is performed by SNN simulator BRIAN [66], [67]. For any letter in the low, medium, or high noise cases, data experiments are repeated for 100 times. Comparison results of SN P systems with SNN in low, medium, and high noise cases are shown in Figures 13, 14 and 15, respectively.

From Figures 13, 14 and 15, it is found that the SN P system performs better than SNN in recognizing most of the letters. The results have some accordance with the result obtained in [63] that for letters having the same number of pixels in their character representations, SNN fails to recognize them. This is the reason why SNN performs poorly in recognizing English letters in noisy cases, particularly in the medium and high noise cases. The spikes representing different letters may be non-unique, which sometimes bring extra difficulty in identifying the letters by spikes. But for SN P systems, spiking rules can be used as a intelligent "selector" to choose a spiking pattern for a neuron when it reads the same number of spikes but at a different time, which can deal with the information encoded by indistinguishable spikes from different letters at different moments.

VI. Conclusion and Discussion

A new variant of SN P systems, called SN P system with learning function, is introduced in this work. With learning function, the weights on synapses can be modified during the computation, which represents the change of the strength of the connection between neurons. Furthermore, a class of SN P systems with simple Hebbian learning function are constructed to recognize English letters with three different levels of noise. The experiments show that the SN P systems achieve average accuracy rate 98.76% in the test cases without noise. In the test cases with low, medium and high noise, SN P systems outperform BP neural networks and PNN. Moreover, by comparison with SNN, SN P systems perform a little better in recognizing letters with noise. The results of this study are promising given that it is the first attempt to use SN P systems in pattern recognition after many theoretical advancements of SN P systems. SN P systems demonstrate a feasible approach for tackling pattern recognition problems.

SN P systems, BP neural network, PNN, and SNN perform well in recognizing letters in the low noise case (flipping 0–3

TABLE IV
VALUES OF RUNNING PARAMETERS IN SPIKING NEURAL NETWORKS FROM [63]

R_m	θ	τ_s, τ_{min}	τ_m, τ_{max}	η	η_{decay}	A^+	A^-	τ^-	τ^-	time step
80	10mv	2ms	30ms	0.1	0.05	0.1	-0.105	1ms	1ms	0.2ms

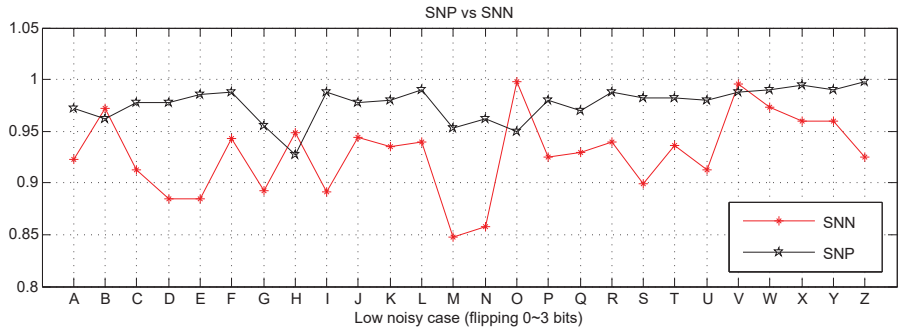

Fig. 13. Comparison results of SNN in the test case with low noise

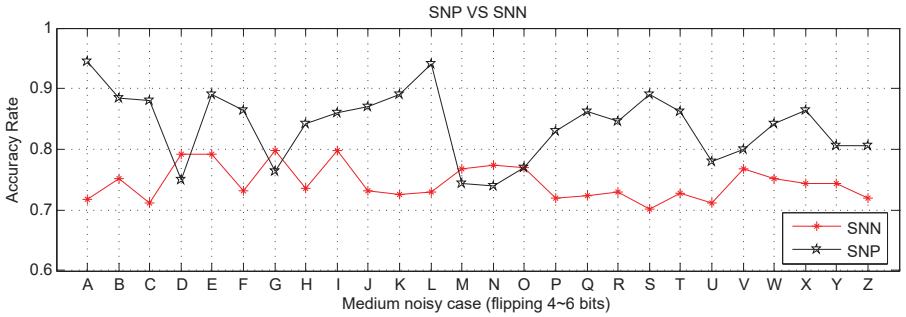

Fig. 14. Comparison results of SNN in the test case with medium noise

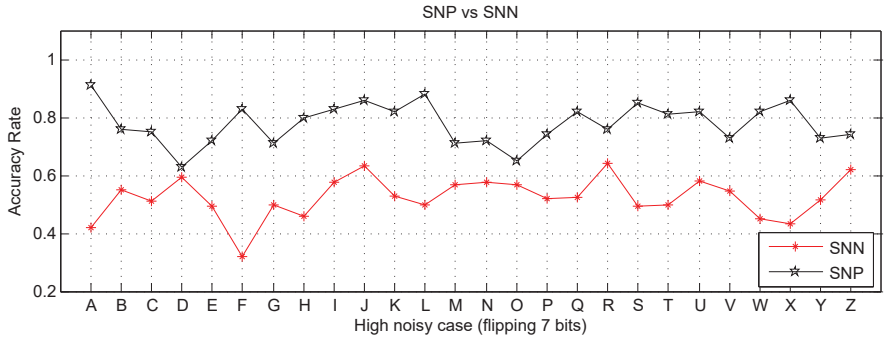

Fig. 15. Comparison results of SNN in the test case with high noise

bits randomly). This is due to the fact that most of the bits of the spike train associated with a letter remain unchanged and are input for recognition. In medium and high noise cases, more than 3 bits of the spike trains associated with letters are randomly flipped, imposing additional difficulty in classifying the letters having less than 3 distinct binary bits in their spike trains. For example, letters "M" and "N" have two distinct binary bits. It is possible that in medium and high noise cases, the two bits of "M" that distinguish it from "N" are selected and flipped, while the distinguishing two bits of letter "N" remain unchanged, making it hard to recognize them correctly. In the high noisy case, 28 spikes (after flipping 7 bits) of a letter can be used as correct input pixels, which helps SN P systems to to overcome BPNN, PNN, and SNN in recognizing letters in this case.

The number of neurons used in SN P systems for recognizing the letters is 76. The Input module has 37 neurons including 35 regular neurons, one start neuron and one input neuron. The Recognize module has 39 neurons, including 4 output neurons and 35 regular neurons. In BPNN, it has 35 input neurons for each pixel of the letter, 20 neurons in the hidden layer and 2 output neurons. It needs 57 neurons for

BPNN recognizing the letters. In PNN, we need 35 neurons as input neurons, 200 neurons in pattern layer, 26 neuron with each letter in summation layer and one output neuron, which is in total 262 neurons in PNN. The SNN with each letter has two layers, where input layer has 35 neurons and output layer is with 26 neurons. It has $(35 + 26) \times 26 = 1586$ neurons. The number of neurons used in SN P systems, BPNN, PNN and SNN is shown in Table V.

TABLE V
THE NUMBER OF NEURONS USED IN SN P SYSTEMS, BPNN, PNN AND SNN

	SN P systems	BPNN	PNN	SNN
Number of neurons	76	57	262	1586

There are some research topics in SN P system with learning function for future work. In our model, spike trains, indicating the spiking frequency are used to represent letters, and the noise is imposed by randomly mutating/flipping a predefined numbers of bits in the spike trains of the standard letter. The way of representing information by spikes could be a topic of interests, such as by spiking vectors or time related spiking behaviours. In addition, the training performance has a close relationship to the architecture of the SN P system, which makes it difficult to recognize the letters having fewer different bits in their standard spike trains. One possible way to solve the problem is to design new architectures of SN P systems (similar job has been done in neural network design theory) or develop a new strategy to represent letters by binary strings such that for any two letters, they have at least 3 distinguished bits. From the simulation results, it appears that in some sense SN P systems have a stochastic resonance-like effect. This should be studied for a theoretical point of view in the near future.

The relative distributions of possible distorted characters relative to the different levels of noise (low, medium, and high noise cases) deserves further research. The distributions will provide quantitative measures of how likely it would be that two different letters with relatively small differences in their undistorted spike trains would overlap after noise. Quantitative measures have potential applications in information communications involving transmitting text in different noisy environments, as well as in the design of decoding strategies when English letters are transmitted by spike trains in information exchange. It would be interesting to consider expanding the capability of SN P systems with learning function to recognize non-English alphabets.

It is an interesting research direction to consider decay or saturation mechanism in SN P systems to avoid the indefinite increase of weights, which will result in possibly very big weights indeed in long computations. Additionally, a learning function is a general notion first introduced into SN P system in this work. It is worthy to investigate the way of defining sophisticated learning functions. In our system, only one neuron is used to read spike trains. As a further research topic, SN P systems should be considered with multiple input neurons, by which Input module can read spike trains in a parallel manner. It would be interesting to construct SN P system which can learn to construct its structure by reading input information.

In the systems constructed in Subsection III, the weights on synapses can only be strengthened during the computation. It is desired to have a mechanism that is able to both strengthen and weaken weights of synapses during the computation, and evaluate the performance on some real life pattern recognition problems.

SN P systems with learning function/capabiliy may become a new direction. Learning strategies and feedback mechanism have been intensively studied and investigated in conventional artificial neural networks. It is worthy to look into these techniques and transplant these ideas into SN P systems so that more complex real life problems might be solved by SN P systems. How to design SN P systems for pattern recognition is also an open problem in this field. A possible way is to design SN P systems by learning some experienced methods from "Neural Network Design Theory" from artificial neural network.

In research of artificial neural networks to recognize digital English letters, database MNIST (Mixed National Institute of Standards and Technology database) is widely used for training various letter recognition systems [68], and for training and testing in the field of machine learning [69]. For further research, SN P systems can be trained with a data set of handwritten digits and the system used to recognize handwritten letters.

REFERENCES

[1] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, no. 1, pp. 108–143, 2000.

[2] G. Păun, G. Rozenberg, and A. Salomaa, *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.

[3] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2, pp. 279–308, 2006.

[4] S. M. Bohte, J. N. Kok, and H. La Poutre, "Error-backpropagation in temporally encoded networks of spiking neurons," *Neurocomputing*, vol. 48, no. 1, pp. 17–37, 2002.

[5] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[6] G. Deco and B. Schürmann, "The coding of information by spiking neurons: an analytical study," *Network: Computation in Neural Systems*, vol. 9, no. 3, pp. 303–317, 1998.

[7] C. W. Eurich and S. D. Wilke, "Multidimensional encoding strategy of spiking neurons," *Neural Computation*, vol. 12, no. 7, pp. 1519–1529, 2000.

[8] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3263–3277, 2015.

[9] Y.T. Liu, Y.Y. Lin, S.L. Wu, C.H. Chuang, and C.T. Lin, "Brain dynamics in predicting driving fatigue using a recurrent self-evolving fuzzy neural network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 347–360, 2016.

[10] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[11] Z. Ni, H. He, and J. Wen, "Adaptive learning in tracking control based on the dual critic network design," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 913–928, 2013.

[12] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014.

[13] H. Zhang, Z. Wang, and D. Liu, "A comprehensive review of stability analysis of continuous-time recurrent neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 7, pp. 1229–1262, 2014.

[14] X. Zhong, H. He, H. Zhang, and Z. Wang, "A neural network based online learning and control approach for markov jump systems," *Neurocomputing*, vol. 149, pp. 116–123, 2015.

[15] G. A. Carpenter, "Neural network models for pattern recognition and associative memory," *Neural Networks*, vol. 2, no. 4, pp. 243–257, 1989.

[16] D. V. Buonomano and M. Merzenich, "A neural network model of temporal code generation and position-invariant pattern recognition," *Neural Computation*, vol. 11, no. 1, pp. 103–116, 1999.

[17] F. Ponulak and A. Kasinski, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural Computation*, vol. 22, no. 2, pp. 467–510, 2010.

[18] S. Roy, P. P. San, S. Hussain, L. W. Wei, and A. Basu, "Learning spike time codes through morphological learning with binary synapses," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1572–1577, 2016.

[19] A. Dundar, J. Jin, B. Martini, and E. Culurciello, "Embedded streaming deep neural networks accelerator with applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 7, pp. 1572–1577, 2016.

[20] W. Maass and C. M. Bishop, *Pulsed neural networks*. MIT press, 2001.

[21] T. Song, X. Liu, Y. Zhao, X. Zhang, "Spiking Neural P Systems with White Hole Neurons," *IEEE Trans on Nanobioscience*, vol. 15, no. 7, pp. 666–673, 2016.

[22] A. Păun and G. Păun, "Small universal spiking neural P systems," *BioSystems*, vol. 90, no. 1, pp. 48–60, 2007.

[23] L. Pan and G. Paun, "Spiking neural P systems with anti-spikes," *International Journal of Computers, Communications & Control, IV (3)*, pp. 273–282, 2009.

[24] T. Song, L. Pan, K. Jiang, B. Song, and W. Chen, "Normal forms for some classes of sequential spiking neural P systems," *IEEE Transactions on NanoBioscience*, vol. 12, no. 3, pp. 255–264, 2013.

[25] M. Cavaliere, O. H. Ibarra, G. Păun, O. Egecioglu, M. Ionescu, and S. Woodworth, "Asynchronous spiking neural P systems," *Theoretical Computer Science*, vol. 410, no. 24, pp. 2352–2364, 2009.

[26] T. Song, L. Pan, and G. Păun, "Asynchronous spiking neural P systems with local synchronization," *Information Sciences*, vol. 219, pp. 197–207, 2012.

[27] J. Wang, H. J. Hoogeboom, L. Pan, G. Paun, and M. J. Pérez-Jiménez, "Spiking neural p systems with weights," *Neural Computation*, vol. 22, no. 10, pp. 2615–2646, 2010.

[28] G. Păun, "Spiking neural p systems with astrocyte-like control." *Journal of Universal Computer Science*, vol. 13, no. 11, pp. 1707–1721, 2007.

[29] X. Zeng, X. Zhang, and L. Pan, "Homogeneous spiking neural P systems," *Fundamenta Informaticae*, vol. 97, no. 1, pp. 275–294, 2009.

[30] T. Song, P. Zheng D.M. Wong, X. Wang, "Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control," *Information Sciences*, vol. 372, pp. 380-391, 2016.

[31] O. H. Ibarra, A. Păun, and A. Rodríguez-Patón, "Sequential SNP systems based on min/max spike number," *Theoretical Computer Science*, vol. 410, no. 30, pp. 2982–2991, 2009.

[32] T. Song, L. Pan, and G. Păun, "Spiking neural P systems with rules on synapses," *Theoretical Computer Science*, vol. 529, pp. 82–95, 2014.

[33] T. Song and L. Pan, "Spiking neural P systems with rules on synapses working in maximum spikes consumption strategy," *IEEE Transactions on NanoBioscience*, vol. 1, pp. 38–44, 2015.

[34] T. Song, L. Pan "Spiking neural P systems with rules on synapses working in maximum spiking strategy," *IEEE Transactions on NanoBioscience*, vol. 4, pp. 465–477, 2015.

[35] M. Ionescu and D. Sburlan, "Several applications of spiking neural P systems," *Fifth Brainstorming Week on Membrane Computing, Sevilla*, 2007.

[36] A. Adl, A. Badr, and I. Farag, "Towards a spiking neural P systems OS," *arXiv preprint arXiv:1012.0326*, 2010.

[37] X. Zeng, T. Song, X. Zhang, and L. Pan, "Performing four basic arithmetic operations with spiking neural P systems," *IEEE Transactions on NanoBioscience*, vol. 11, no. 4, pp. 366–374, 2012.

[38] 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*, vol. 24, no. 05, 2014.

[39] 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*, vol. 30, no. 3, pp. 1182–1194, 2014.

[40] T. Song, S. Pang, S. Hao, et al. "A parallel image skeletonizing method using spiking neural P systems with weights," Neural Processing Letters, pp. 1-18, 2018.

[41] T.-O. Ishdorj, A. Leporati, L. Pan, X. Zeng, and X. Zhang, "Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources," *Theoretical Computer Science*, vol. 411, no. 25, pp. 2345–2358, 2010.

[42] A. Leporati, G. Mauri, C. Zandron, G. Păun, and M. J. Pérez-Jiménez, "Uniform solutions to SAT and Subset Sum by spiking neural P systems," *Natural Computing*, vol. 8, no. 4, pp. 681–702, 2009.

[43] L. Pan, G. Păun, and M. J. Pérez-Jiménez, "Spiking neural P systems with neuron division and budding," *Science China Information Sciences*, vol. 54, no. 8, pp. 1596–1607, 2011.

[44] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[45] L. Pan and X. Zeng, "A note on small universal spiking neural P systems," *Lecture Notes in Computer Science*, vol. 5957, pp. 436–447, 2010.

[46] G. A. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*. The MIT Press, 1991.

[47] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for eeg classification and epilepsy and seizure detection," *Integrated Computer-Aided Engineering*, vol. 14, no. 3, pp. 187–212, 2007.

[48] A. Gupta and L. N. Long, "Character recognition using spiking neural networks," in *Proceedings of the International Joint Conference on Neural Networks*. IEEE, 2007, pp. 53–58.

[49] M. Kang and D. Palmer-Brown, "A modal learning adaptive function neural network applied to handwritten digit recognition," *Information Sciences*, vol. 178, no. 20, pp. 3802–3812, 2008.

[50] Y. Lecun, Y. Bengio, G. Hinton. "Deep learning," Nature, vol. 521, no. 7553, pp.436, 2015.

[51] B. Zhou, A. Lapedriza, J. Xiao J, et al. "Learning deep features for scene recognition using places database," Advances in Neural Information Processing Systems, vol. 12, no. 1, pp 487-495, 2014.

[52] M. Gheorghe, G. Păun, M. J. Pérez-Jiménez, and G. Rozenberg, "Chapter 13: Spiking neural P systems, research frontiers of membrane computing: Open problems and research topics," *International Journal of Foundations of Computer Science*, vol. 24, no. 05, pp. 547–623, 2013.

[53] M. Sipser, *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[54] J. E. Hopcroft, *Introduction to Automata Theory, Languages, and Computation*. Pearson Education India, 1979.

[55] R. Mitkov, *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 2005.

[56] J. S. Camacho, H. Li, P. Li, W. Zhang, "Machine printed character recognition system using backpropagation neural network," *Computer Knowledge and Technology*, vol. 5, no. 19, pp. 5238–5241, 2009.

[57] F. Li, S. Gao, "Character recognition system based on back-propagation neural network," *In Proceedings of IEEE 2010 International Conference on the Machine Vision and Human-Machine Interface*, IEEE, 2010, pp. 394–396, 2010.

[58] Hagan, M.T., H.B. Demuth, and M.H. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[59] Moller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, pp. 525–533, 1993.

[60] M. M. Ibrahiem, E. Emary, S. Ramakrishnan, "On the application of various probabilistic neural networks in solving different pattern classification problems," *World Applied Sciences Journal*, vol. 4, no. 6, pp. 772–780, 2008.

[61] P. D. Wasserman, *Advanced Methods in Neural Computing*. John Wiley & Sons, Inc., 1993.

[62] J.-H. Li, A. N. Michel, and W. Porod, "Analysis and synthesis of a class of neural networks: linear systems operating on a closed hypercube," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 11, pp. 1405–1422, 1989.

[63] A. Gupta and L. N. Long, "Character recognition using spiking neural networks," *In Proceedings of the International Joint Conference on Neural Networks*, IEEE, 2007, pp. 53–58.

[64] P. Christo, "Temporal processing in a spiking model of the visual system," *ecture Notes in Computer Science*, vol. 4131, pp. 750–759, 2006.

[65] A. Delorme, S. J. Thorpe, "Face identification using one spike per neuron: resistance to image degradations," *Neural Networks*, vol. 14, pp. 795–804, 2001.

[66] D. Goodman, R. Brette, "The Brian simulator", *Frontiers in Neuroscience*, vol 26, no. 3, 2009.

[67] M. Stimberg, D. Goodman, V. Benichoux, R. Brette, "Equation-oriented specification of neural models for simulations," *Frontiers in Neuroinformatics*, vol. 8, no. 6, 2014.

[68] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[69] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, no. 10, pp. 2271–2285, 2003.