Applications of Spiking Neural P Systems in Cybersecurity

Mihail-Iulian Pleșa*[1], Marian Gheoghe[2], Florentin Ipate[1], and Gexiang Zhang[3]

[1] Department of Computer Science, University of Bucharest, Bucharest, Romania
mihail-iulian.plesa@s.unibuc.ro
[2] School of Electrical Engineering and Computer Science, University of Bradford, Bradford, UK
[3] School of Automation, Chengdu University of Information Technology, Chengdu 610225, China

Abstract. Spiking Neural P systems are third-generation neural networks that are much more energy efficient than the current ones. In this paper, we investigate for the first time the possibility of using Spiking Neural P systems to solve cybersecurity-related problems. We proposed a new architecture called Cyber Spiking Neural P systems (Cyber-SN P systems for short), which is designed especially for cybersecurity data and problems. We trained multiple Cyber-SN P systems to detect malware on the Android platform, phishing websites, and spam e-mails. We show through experiments that these networks can efficiently classify cybersecurity-related data with much fewer training epochs than perceptron-based artificial neural networks.

Keywords: Cybersecurity · Spiking Neural P systems · Machine learning

1 Introduction

Cybersecurity is one field in which machine learning has a disruptive impact [14]. Currently, most threat detection systems are based on signatures. The disadvantage of this approach is that the attacker can easily modify the malware to avoid detection. Machine learning algorithms can automatically analyze large datasets to detect patterns that can help to classify new entries. The most common type of attack architecture consists of the deployment of malware through a phishing website or a spam e-mail [22]. Deep learning (DL) algorithms can be used successfully at different levels of this architecture:

1. DL can detect spam e-mails. In this way, the user is no longer exposed to mail containing malware [8].
2. DL can detect phishing websites. In this way, even if the user accesses a malicious URL by mistake, he is informed of the vicious nature of the accessed website [40].
3. DL can detect malware. In this way, even if the user downloads a malicious binary, the system can detect it and stop its execution [33].

Although they offer high performance, the DL algorithms have a major disadvantage in terms of large energy consumption [39]. One reason why this happens is that the current deep learning algorithms are not inspired directly by the way the human brain

works [27]. Spiking neural networks, in general, and Spiking Neural P systems (SN P systems for short) in particular, represent a possible solution to this problem [16]. SN P systems are neural networks inspired by how the neural cells work and interact. Unlike current neural networks, the neurons from an SN P system communicate using discrete spiking. In an SN P system, each neuron has a number of firing rules. When the firing condition is met, the neuron will emit one spike to all neurons with which it is connected. This raises the possibility of implementing neural networks that are energy-efficient on neuromorphic hardware [17]. In this paper, we analyze the possibility of using the SN P system to solve three cybersecurity problems: malware detection, website phishing detection, and e-mail spam detection.

2 Related work and our contribution

In general, there are two main categories of machine learning algorithms used in malware detection: supervised and unsupervised. Supervised learning requires a labeled dataset used by the algorithm to learn how to classify new samples. On the other hand, the purpose of unsupervised learning algorithms, which do not require any labeled data, is to compute an approximate distribution of the dataset. In [19], the authors propose an ML system that detects Android malware by analyzing permission usage. To detect the best classifier, they analyzed four different algorithms: AdaBoost, Naive Bayes, Decision Trees, and SVM [12]. In [18] it is presented a random forest classifier is presented to detect malware based on more than 377 features about Android APK. In [28] it is proposed a decision tree approach that classifies samples as malware or benign based on the operational codes in machine language (opcode for short) frequency. In [4], the authors presented DRACO, a machine-learning framework that detects Android malware based on static and dynamic features. Another idea of combining dimensionality reduction and a classifier for Android malware detection is presented in [35]. All of these ideas were based on classical machine learning algorithms. There are also other more recent methods for detecting Android malware using deep learning. In [29], the authors proposed an artificial neural network that detects Android malware based on dynamic features with more than 97% accuracy. The authors in [38] proposed MalNet, a deep learning architecture for malware detection with automatic feature selection. The idea is to treat the malware as a grayscale image and then use convolution neural networks for classification. In [15] the authors used deep belief networks to detect malware based on Application Programming Interface (API) calls of the APK.

SN P systems, and P systems in general, have varied characteristics, each being designed from different biological phenomena found in neuronal cells [23, 24, 30, 32, 36]. There were also previous applications of SN P systems in machine learning [5, 44, 41, 26, 6]. In [43], SN P systems were used to construct a spiking convolutional neural network. In [21], a specific SN P system called Dynamic Threshold Neural P system (DTN P system for short) was used in medical image processing [3]. In [42], the authors proposed a general classifier based on the SN P system and tested it on the handwritten digit images dataset (MNIST dataset for short) [9]. Two comprehensive surveys on image processing using P systems are presented in [10] and [37].

This paper studies, for the first time, solving cybersecurity problems with SN P systems. Starting from the Layered Spiking Neural P systems (LSN P systems for short) introduced in [42], we proposed a new spiking neural network architecture called the Cyber Spiking Neural P systems. We train multiple Cyber-SN P systems to detect malware on Android platforms, phishing websites, and spam e-mails. Our experiments show that Cyber-SN P systems require much less training epochs than perceptron-based artificial neural networks (ANNs for short) for comparable performances. The results obtained demonstrate the suitability of the SN P systems for a new range of problems, namely, in the area of cybersecurity.

The paper is organized as follows: in Section 3, we present the Cyber-SN P systems; in Section 4, we describe the experiments in which we evaluate the Cyber-SN P systems in the detection of malware, phishing websites, and spam e-mails. Section 5 concludes the paper and suggests some further research directions.

3 Cyber-SN P systems

In this section, we describe an efficient classifier for cybersecurity problems based on the LSN P systems described in [42].

Definition 1. *A Cyber Spiking Neural P system is defined as the following construct:*
$\Pi = (\{a\}, \sigma_{p_1}, \cdots \sigma_{p_k}, \sigma_{r_1}, \sigma_{r_2}, \sigma_o, syn, IN, OUT)$

where:

1. The symbol a denotes a spike.
2. $\sigma_{p_i} = \{\alpha_i, w_{ij}, r_i\}, 1 \leq i \leq k, 1 \leq j \leq 2$ denotes an input proposition neuron:
   (a) α_i represents the potential value of the neuron and is equal to the value of the i^{th} feature from the input vector.
   (b) w_{ij} denotes the weight on the synapse that connects the neuron σ_{p_i} to the neuron σ_{r_j}. Initially, the weights are chosen randomly from the range $[0, 1]$.
   (c) r_i denotes a set of firing rules of the form $r_i : E^1/a^{\alpha_i} \rightarrow a^{\alpha_i}$, where $E^1 = \{\alpha_i \geq 0\}$.
3. $\sigma_{r_j} = \{\theta_j, r_j\}, j \in \{1, 2\}$ denotes a rule neuron:
   (a) θ_j denotes the potential value of the neuron.
   (b) r_j denotes a set of firing rules of the form $r_j : E^2/a^{\theta_j} \rightarrow a^{\theta_j}$, where $E^2 = \{\theta_j \geq 0\}$ and $\theta_j = (w_{1j} \times \alpha_1) + (w_{2j} \times \alpha_2) + \cdots + (w_{kj} \times \alpha_k)$.
4. $\sigma_o = \{\alpha_o, r_o\}$ denotes an output proposition neuron:
   (a) α_o denotes the potential value of the neuron and is equal to the maximum of the potential values received from the neurons σ_{r_1} and σ_{r_2}.
   (b) r_o denotes a set of firing rules of the form $r_o : E^3/a^{\alpha_o} \rightarrow a^{\alpha_o}$, where $E^3 = \{\alpha_o \geq 0\}$.
5. $syn = \left\{ \left(\sigma_{p_i}, \sigma_{r_j}\right), \left(\sigma_{r_j}, \sigma_o\right), 1 \leq i \leq k,\ 1 \leq j \leq 2 \right\}$
6. $IN = \{\sigma_{p_i}, 1 \leq i \leq k\}$
7. $OUT = \{\sigma_o\}$

Figure 1 shows the Cyber-SN P system used in this paper. The running steps of the system are the following:

1. The input is encoded using the standard scaling algorithm. If $\mathbf{x}$ is the training dataset, the scaling mechanism transforms it into one that has a mean of 0 and a variance of 1:
$$\mathbf{x} \leftarrow \frac{\mathbf{x} - \boldsymbol{\mu_x}}{\boldsymbol{\sigma_x}}$$
where $\boldsymbol{\mu_x}$ and $\boldsymbol{\sigma_x}$ represent the mean and the standard deviation of the dataset, respectively.
2. The input layer will receive the encoded input. Each neuron will send its potential value to all of the neurons from the hidden layer. The potential value of a neuron from the hidden layer is the weighted sum of the received potential values from the input neurons.
3. The neuron σ_o will output the maximum potential value received from the hidden layer.

The learning in a Cyber-SN P system is based on a slightly modified version of the Widrow-Hoff learning rule [34]. The rule first applies the sigmoid function to the output of the Cyber-SN P system and then updates the weights. The reason for applying the sigmoid function is to transform the output of the system into a real value between 0 and 1, which can be interpreted as a probability. Let $\tilde{t}$ be the output of the system and by t the ground truth. Let $\mathbf{W}$ be the $k \times 2$ matrix of weights, where k represents the number of features of an input vector. Updating the weights is done accordingly to (1).

$$\mathbf{W} \Leftarrow \mathbf{W} + \eta \left(f\left(\tilde{t}\right) - t \right) \boldsymbol{x} \tag{1}$$

where η is the learning rate, $\boldsymbol{x}$ is the input and f is the sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

.

The learning rule will be applied for several training epochs until satisfactory accuracy is obtained over the training set.

There are certain differences specific to cybersecurity-related problems between this construction and the LSN P systems proposed in [42]:

1. Since cybersecurity-related problems require an input with a large number of features, Cyber-SN P systems do not require an encoding mechanism that increases the dimensions of the input vector. The LSN P systems use Taylor decomposition to transform an m-dimenisonal input vector into one of $\sum_{i=1}^{k} \binom{i}{m+i-1}$ dimensions [42].
2. The inputs of a Cyber-SN P system are not fuzzy values, like the ones of an LSN P system. Thus, they are not limited to real numbers from $[0, 1]$. This is important for cybersecurity-related data where sensitivity to outliers is important [11]. The inputs of an LSN P system are encoded using the min-max normalization. In contrast, Cyber-SN P systems use standard scaling, which is much more sensitive to outliers than min-max normalization [13].

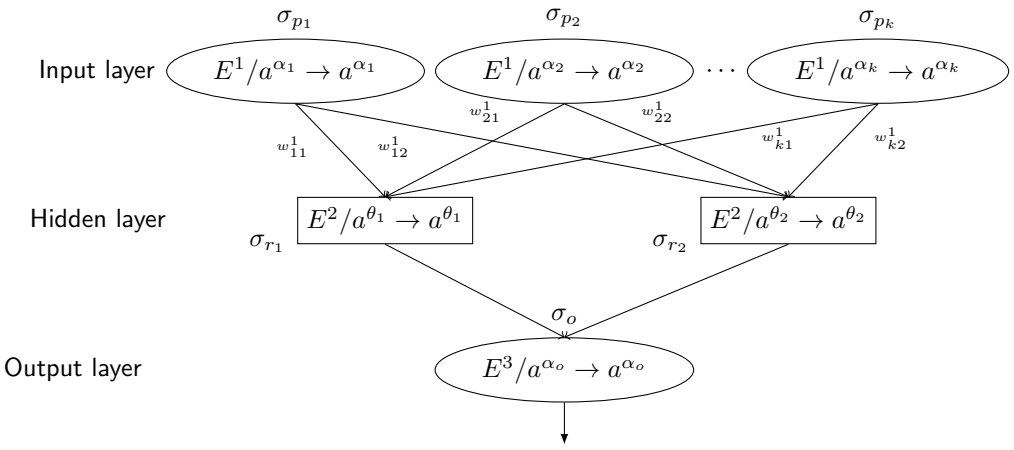

Fig. 1: Cyber-SN P system

3. Cybersecurity problems are usually solved using binary classification algorithms, which greatly simplify the model. Cyber-SN P systems have only three layers. The LSN P systems are constructed with five layers.
4. In the cybersecurity field, it is important for a classification algorithm to return not only the classification result but also the associated probabilities for each class. Cyber-SNP outputs a potential value that can be transformed into a probability by applying the sigmoid function. LSN P systems were defined to return only the class to which the input belongs.

It is important to note that during the training process, some of the weights might become negative. As a result, it is not guaranteed that a neuron's potential value will always be positive. This characteristic ensures that neurons' spiking behavior is dynamic

and dependent on the training data since a neuron from the input or hidden layer will fire only if its potential value is positive.

4 Experimental results

In this section, we experimentally assess the performances of Cyber-SN P systems on cybersecurity-related tasks and compare them with those of neural networks. For each dataset, we measure the following characteristics of the classifier:

1. *True positive (TP)*: Represents a sample that is malicious and is classified as malicious.
2. *False positive (FP)*: Represents a sample that is benign but is classified as malicious.
3. *True negative (TN)*: Represents a sample that is benign and is classified as benign.
4. *False negative (FN)*: Represents a sample that is malicious and is classified as benign.

We use four metrics to evaluate the classification algorithms [13]:

1. *Accuracy (Acc)*: Represents the ratio between the number of correctly classified samples and the total number of samples:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

2. *Precision (P)*: Represents the ratio between the number of samples correctly classified as malicious and the total number of samples classified as malicious:

$$P = \frac{TP}{TP + FP} \tag{3}$$

3. *Recall (R)*: Represents the ratio between the number of samples correctly classified as malicious and the total number of malicious samples:

$$R = \frac{TP}{TP + FN} \tag{4}$$

4. *F1 score (F1)*: Represents the harmonic mean between the precision and recall:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{5}$$

All the experiments were performed using the Python programming language on an Apple Mac M1 Max platform with 10 cores and 32 GB of RAM. All neural networks were trained using the Tensorflow library, version 2.16.1. For each dataset, 80% of the data is used to train the classifier and 20% for testing. The source code for the experiments is available at https://github.com/miiip/Cyber-LSNP.

4.1 Android malware detection

These experiments compare the Cyber-SN P systems with neural networks for Android malware detection. The dataset used is Drebin-215 [2]. Each sample in the dataset is characterized by 215 features from the following categories:

1. API call signatures, e.g., onServiceConnected, Android.os.Binder, attachInterface, etc.
2. Manifest permission e.g. SEND_SMS, READ_PHONE_STATE, RECEIVE_SMS, etc.
3. Intent, e.g., Android.intent.action.PACKAGE_REPLACED, Android.intent.action.TIME_SET, Android.intent.action.BATTERY_OKAY, etc.
4. Commands signature, e.g., mount, remount, chown, etc.

The dataset is composed of 15036 samples, of which 9476 are benign, and 5560 are malware.

The ANN has 215 input neurons and 2 output neurons with Softmax activation function [1]. The network is trained with the Adam optimizer using a learning rate of 0.001. In each trial of the experiment, the ANN and the Cyber-SN P system were trained until the accuracy over the training dataset was greater than 0.97. The performance metrics were averaged over all trials, and the results are shown in Table 1. Although the performances are similar, the main difference between the two models is that the Cyber-SN P system uses, on average, only 2.15 epochs for training, unlike the ANN, which needs 11.75 epochs to be trained.

Table 1: Android malware detection - comparison between LNS P system and ANN

Model	**Number of epochs**	**Acc**	**P**	**R**	**F1**
Cyber-SN P system	1.95	0.9690	0.9691	0.9690	0.9690
ANN	10.95	0.9761	0.9731	0.9621	0.9676

To illustrate the training behaviors of the Cyber-SN P system and the ANN, we trained both algorithms for 50 epochs. As can be seen from Figures 2 and 3, the training of the ANN is smoother than that of the Cyber-SN P system for the same learning rate.

4.2 Phishing detection

These experiments compare the Cyber-SN P systems with neural networks for website phishing detection. The dataset used is constructed from [31]. There are 10000 samples in the dataset: 5000 representing legitimate websites and 5000 representing phishing websites. Each sample is composed of 48 features signifying different characteristics of the webpage [7]. There are three types of features:

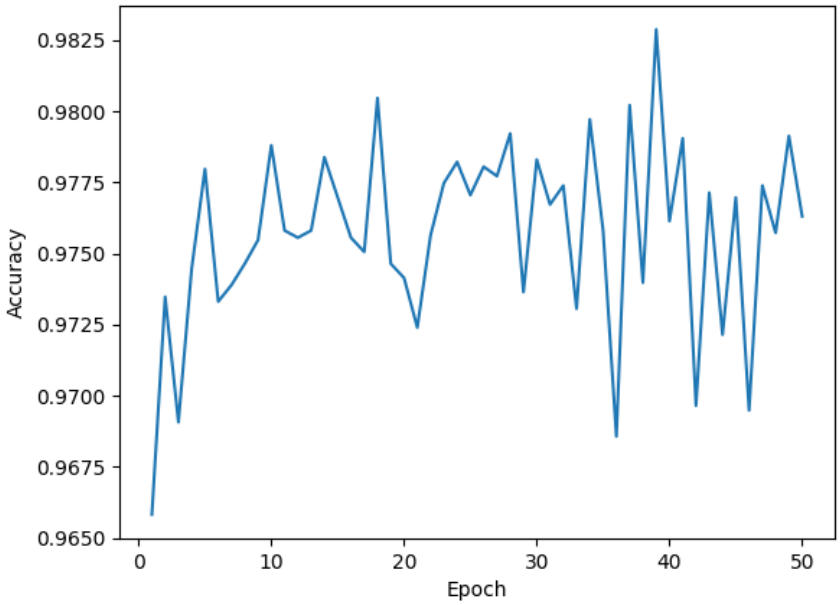

Fig. 2: Malware detection - Accuracy of the Cyber-SN P system with respect to the training epoch

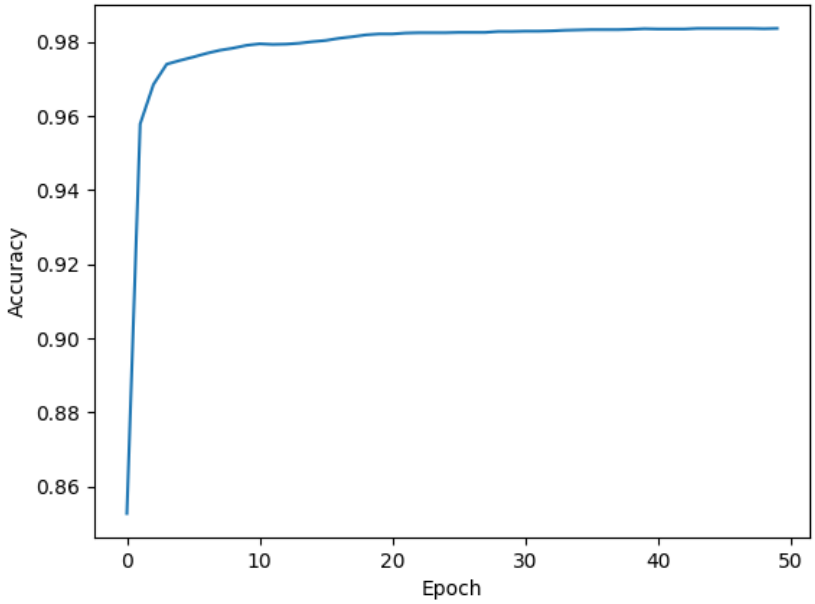

Fig. 3: Malware detection - Accuracy of the ANN with respect to the training epoch

1. Discrete features: the number of dots in the URL, the depth of the path in the URL, the number of special characters in the URL (e.g. "-", "&", "#", "_", "%", etc.), the number of numeric characters in the URL, the number of sensitive words in the URL (e.g. "banking", "safe", "secure", etc.), the total number of characters in the URL, etc.
2. Binary features: the existence of "@" symbol in the URL, the existence of "$\sim$" symbol in the URL, whether the webpage uses HTTPS, whether the IP address is used directly in the URL, the existence of the sequence "//" in the URL, whether the most frequent domain named in the source code of the webpage is corresponding to the domain name from the URL, the existence of JavaScript in the webpage, whether the function "mailto" is used in the webpage, etc..
3. Continuous features: the percentage of external hyperlinks to and from the webpage, the percentage of hyperlinks fields containing empty value or self-redirect value, etc.

The complete list of features, together with their descriptions, can be found in [7].

Since each input of the Cyber-SN P system is a vector of 48 values and the output is binary, we compare the performances with those of an ANN with 48 input neurons and 2 output neurons. The activation function used in the ANN is Softmax, and the network is trained using the Adam optimizer with a learning rate of 0.001 [1]. Both systems were trained until the accuracy of the training dataset was greater than 0.97. The averaged results are described in Table 2. The Cyber-SN P system performed slightly better than the ANN in terms of accuracy, precision, recall, and F1-score, and it was trained on average for 1.1 epochs instead of 16.25.

Table 2: Phishing detection - comparison between LNS P system and ANN

Model	**Number of epochs**	**Acc**	**P**	**R**	**F1**
Cyber-SN P system	1.65	0.9905	0.9906	0.9905	0.9906
ANN	15.70	0.9889	0.9888	0.9891	0.9890

Figures 4 and 5 show the accuracies of the Cyber-SN P system and the ANN with respect to the training epoch. Both systems were trained for 50 epochs. As in the case of malware detection, the training of the ANN is smoother than that of the Cyber-SN P system for the same learning rate.

4.3 Spam detection

These experiments compare the Cyber-SN P systems with neural networks for spam detection. The dataset used is constructed from [20]. There are 5171 e-mails in the dataset. 3672 e-mails are ham while 1499 are spam. Unlike the previous tasks, the samples are not composed of manually crafted features but from raw text. To encode a text e-mail as a vector of real numbers, we used the following procedure:

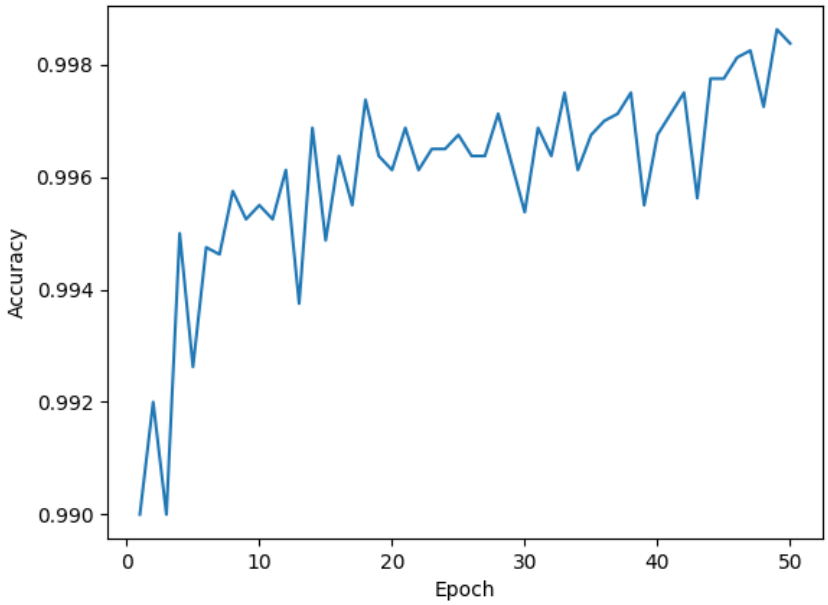

Fig. 4: Phishing detection - Accuracy of Cyber-SN P system with respect to the training epoch

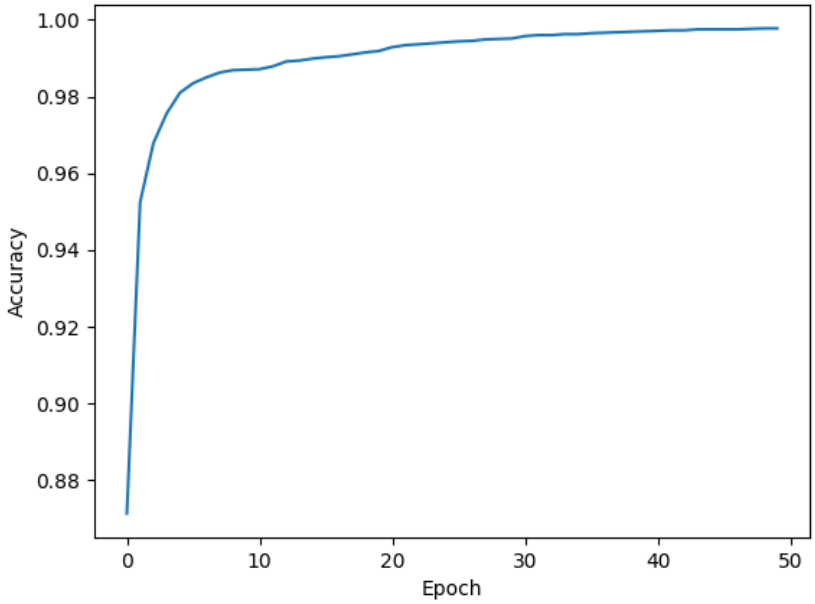

Fig. 5: Phishing detection - Accuracy of the ANN with respect to the training epoch

1. Split the text into words based on spaces.
2. For each word, use a pre-trained embedding map to transform the word into a vector of real numbers.
3. The vector that represents the text is computed as the average of the vectors assigned to each word from the text.

For this experiment, we use the Glove map [25]. The embedding map is pre-trained on 6 billion tokens and outputs for each input word a vector of 300 real numbers. The map preserves the semantic meaning of the word, i.e., semantically close words are mapped to close vectors in terms of Euclidean distance.

Both the Cyber-SN P system and the ANN have 300 input neurons and 2 output neurons. Similar to the other experiments, the systems were trained until the accuracy over the training dataset was greater than 0.93. The averaged results are described in Table 3. The performances are similar, but the number of epochs needed to train the ANN is almost ten times higher compared to the number of epochs needed to train the Cyber-SN P system.

Table 3: Spam detection - comparison between LNS P system and ANN

Model	**Number of epochs**	**Acc**	**P**	**R**	**F1**
Cyber-SN P system	1.00	0.9443	0.9448	0.9443	0.9443
ANN	10.90	0.9603	0.9493	0.9723	0.9606

Figures 6 and 7 show the accuracies of the Cyber-SN P system and the ANN with respect to the training epoch. As in previous experiments, the systems were trained for 50 epochs. The results indicate that the training of the ANN is smoother than that of the Cyber-SN P system.

5 Conclusions and further directions of research

We demonstrated the performances of the Cyber-SN P systems in three different types of attacks: malware infection, website phishing, and e-mail spam spreading. Table 4 presents a summary of our results. We experimentally showed that Cyber-SN P systems can solve cybersecurity-related problems much more efficiently in terms of training epochs than perceptron-based neural networks. For the detection of phishing websites, Cyber-SN P systems performed better from the perspective of all analyzed metrics. For all experiments, the training process of the ANN was smoother than that of the Cyber-SN P system.

The first direction of research is to compare SN P systems with other deep learning architectures, e.g., Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), etc. The second one is to implement the Cyber-SN P system on dedicated neuromorphic hardware and to asses their energy consumption efficiency.

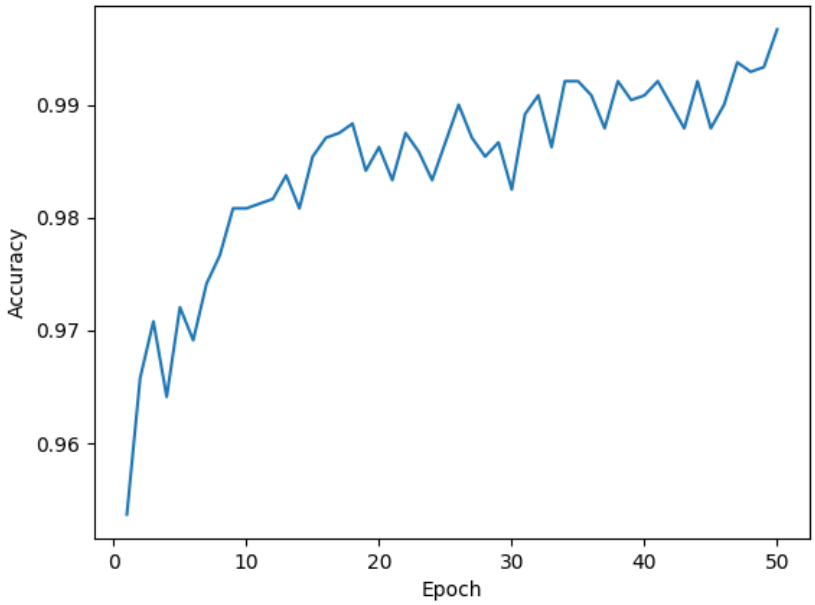

Fig. 6: Spam detection - Accuracy of Cyber-SN P system with respect to the training epoch

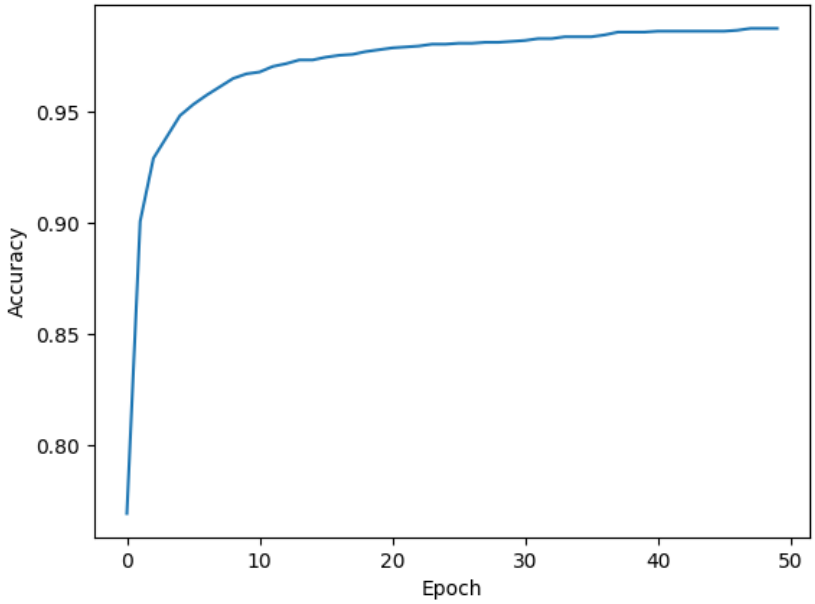

Fig. 7: Spam detection - Accuracy of the ANN with respect to the training epoch

Attack	Model	Number of epochs	Acc	P	R	F1
Malware	Cyber-SN P system	1.95	0.9690	0.9691	0.9690	0.9690
	ANN	10.95	0.9761	0.9731	0.9621	0.9676
Phishing	Cyber-SN P system	1.65	0.9905	0.9906	0.9905	0.9906
	ANN	15.70	0.9889	0.9888	0.9891	0.9890
Spam	Cyber-SN P system	1.00	0.9443	0.9448	0.9443	0.9443
	ANN	10.90	0.9603	0.9493	0.9723	0.9606

Table 4: Comparison between LNS P systems and ANN

Declarations

Funding

This work was supported by the National Natural Science Foundation of China (61972324), the Sichuan Science and Technology Program (2023NSFSC1985) and Research Fund of Chengdu University of Information Technology (KYTD202212).

References

1. Agarap, Abien Fred: Deep learning using rectified linear units (ReLU). arXiv preprint arXiv:1803.08375 (2018)
2. Arp, Daniel and Spreitzenbarth, Michael and Hubner, Malte and Gascon, Hugo and Rieck, Konrad and Siemens, CERT: Drebin: Effective and explainable detection of Android malware in your pocket. In: NDSS. vol. 14, pp. 23–26 (2014)
3. Bao, T. and Zhou, N. and Lv, Z. and Peng, H. and Wang, J.: Sequential dynamic threshold neural P systems. Journal of Membrane Computing **2**(4), 255–268 (2020)
4. Bhandari, Shweta and Gupta, Rishabh and Laxmi, Vijay and Gaur, Manoj Singh and Zemmari, Akka and Anikeev, Maxim: DRACO: DRoid analyst combo an Android malware analysis framework. In: Proceedings of the 8th International Conference on Security of Information and Networks. pp. 283–289 (2015)
5. Ceon, Y.P., Anandharaj, H.C., Jebasingh, S., Chandy, D.A.: Generation of chain code pictures using cell-like spiking neural P system with several types of spikes. Journal of Membrane Computing **4**(3), 243–250 (2022)
6. Chen, Y., Chen, Y., Zhang, G., Paul, P., Wu, T., Zhang, X., Rong, H., Ma, X.: A survey of learning spiking neural p systems and a novel instance. International Journal of Unconventional Computing **16**, 173–200 (06 2021)
7. Chiew, Kang Leng and Tan, Choon Lin and Wong, KokSheik and Yong, Kelvin SC and Tiong, Wei King: A new hybrid ensemble feature selection framework for machine learning-based phishing detection system. Information Sciences **484**, 153–166 (2019)
8. Dada, Emmanuel Gbenga and Bassi, Joseph Stephen and Chiroma, Haruna and Adetunmbi, Adebayo Olusola and Ajibuwa, Opeyemi Emmanuel and others: Machine learning for email spam filtering: review, approaches and open research problems. Heliyon **5**(6) (2019)
9. Deng, Li: The MNIST database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)

10. Díaz-Pernil, Daniel and Gutiérrez-Naranjo, Miguel A and Peng, Hong: Membrane computing and image processing: a short survey. Journal of Membrane Computing **1**(1), 58–73 (2019)
11. Evangelou, Marina and Adams, Niall M: An anomaly detection framework for cyber-security data. Computers & Security **97**, 101941 (2020)
12. Geron, A.: Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Sebastopol, CA (2017)
13. Goodfellow, Ian and Bengio, Yoshua and Courville, Aaron: Deep learning. MIT press (2016)
14. Halbouni, Asmaa and Gunawan, Teddy Surya and Habaebi, Mohamed Hadi and Halbouni, Murad and Kartiwi, Mira and Ahmad, Robiah: Machine learning and deep learning approaches for cybersecurity: A review. IEEE Access **10**, 19572–19585 (2022)
15. Hou, Shifu and Saas, Aaron and Ye, Yanfang and Chen, Lifei: Droiddelver: An Android malware detection system using deep belief network based on API call blocks. In: Web-Age Information Management: WAIM 2016 International Workshops, MWDA, SDMMW, and SemiBDMA, Nanchang, China, June 3-5, 2016, Revised Selected Papers 17. pp. 54–66. Springer (2016)
16. Ionescu, Mihai and Păun, Gheorghe and Yokomori, Takashi: Spiking neural P systems. Fundamenta Informaticae **71**(2-3), 279–308 (2006)
17. Kumar, Shobhit and Das, Shirshendu and Badone, Gourav and Kumar, Amit: A Survey on Efficient Interconnects for Neuromorphic Systems. In: Soft Computing: Theories and Applications: Proceedings of SoCTA 2021, pp. 709–718. Springer (2022)
18. Li, Jian and Wang, Zheng and Wang, Tao and Tang, Jinghao and Yang, Yuguang and Zhou, Yihua: An Android malware detection system based on feature fusion. Chinese Journal of Electronics **27**(6), 1206–1213 (2018)
19. Li, Jin and Sun, Lichao and Yan, Qiben and Li, Zhiqiang and Srisa-An, Witawas and Ye, Heng: Significant permission identification for machine-learning-based Android malware detection. IEEE Transactions on Industrial Informatics **14**(7), 3216–3225 (2018)
20. Metsis, Vangelis and Androutsopoulos, Ion and Paliouras, Georgios: Spam filtering with naive bayes-which naive bayes? In: CEAS. vol. 17, pp. 28–69. Mountain View, CA (2006)
21. Mi, Siheng and Zhang, Li and Peng, Hong and Wang, Jun: Medical image fusion based on DTNP systems and Laplacian pyramid. Journal of Membrane Computing **3**(4), 284–295 (2021)
22. Mijwil, Maad and Unogwu, Omega John and Filali, Youssef and Bala, Indu and Al-Shahwani, Humam: Exploring the top five evolving threats in cybersecurity: an in-depth overview. Mesopotamian Journal of Cybersecurity **2023**, 57–63 (2023)
23. Pan, L., Song, B., Zandron, C.: On the computational efficiency of tissue P systems with evolutional symport/antiport rules. Knowledge-Based Systems **262**, 110266 (2023)
24. Pan, Ya and Ge, Xiuting and Fang, Chunrong and Fan, Yong: A systematic literature review of Android malware detection using static analysis. IEEE Access **8**, 116363–116379 (2020)
25. Pennington, Jeffrey and Socher, Richard and Manning, Christopher D: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)
26. Plesa, M.I., Gheoghe, M., Ipate, F., Zhang, G.: A key agreement protocol based on spiking neural P systems with anti-spikes. Journal of Membrane Computing **4**(4), 341–351 (2022)
27. Raichle, Marcus E and Gusnard, Debra A: Appraising the brain's energy budget. Proceedings of the National Academy of Sciences **99**(16), 10237–10239 (2002)
28. Santos, Igor and Brezo, Felix and Ugarte-Pedrero, Xabier and Bringas, Pablo G: Opcode sequences as representation of executables for data-mining-based unknown malware detection. Information Sciences **231**, 64–82 (2013)
29. Singh, Latika and Hofmann, Markus: Dynamic behavior analysis of Android applications for malware detection. In: 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT). pp. 1–7. IEEE (2017)

30. Song, B., Pan, L.: Rule synchronization for tissue P systems. Information and Computation **281**, 104685 (2021)
31. Tan, Choon Lin: Phishing dataset for machine learning: Feature evaluation. Mendeley Data **1**, 2018 (2018)
32. Verlan, S., Freund, R., Alhazov, A., Ivanov, S., Pan, L.: A formal framework for spiking neural P systems. Journal of Membrane Computing **2**(4), 355–368 (2020)
33. Vinayakumar, R and Alazab, Mamoun and Soman, KP and Poornachandran, Prabaharan and Venkatraman, Sitalakshmi: Robust intelligent malware detection using deep learning. IEEE access **7**, 46717–46738 (2019)
34. Wang, Zi-Qin and Manry, Michael T and Schiano, Jeffrey L: LMS learning algorithms: misconceptions and new results on convergence. IEEE Transactions on Neural Networks **11**(1), 47–56 (2000)
35. Wei, Linfeng and Luo, Weiqi and Weng, Jian and Zhong, Yanjun and Zhang, Xiaoqian and Yan, Zheng: Machine learning-based malicious application detection of Android. IEEE Access **5**, 25591–25601 (2017)
36. Wu, T., Lyu, Q., Pan, L.: Evolution-communication spiking neural P systems. International Journal of Neural Systems **31**(02), 2050064 (2021)
37. Yahya, Rafaa I and Shamsuddin, Siti Mariyam and Yahya, Salah I and Hasan, Shafatnnur and Al-Salibi, Bisan and Al-Khafaji, Ghada: Image segmentation using membrane computing: a literature survey. In: International Conference on Bio-Inspired Computing: Theories and Applications. pp. 314–335. Springer (2016)
38. Yan, J., Qi, Y., Rao, Q.: Detecting malware with an ensemble method based on deep neural network. Security and Communication Networks **2018**, 1–16 (03 2018). https://doi.org/10.1155/2018/7247095
39. Yang, Tien-Ju and Chen, Yu-Hsin and Emer, Joel and Sze, Vivienne: A method to estimate the energy consumption of deep neural networks. In: 2017 51st asilomar conference on signals, systems, and computers. pp. 1916–1920. IEEE (2017)
40. Yi, Ping and Guan, Yuxiang and Zou, Futai and Yao, Yao and Wang, Wei and Zhu, Ting and others: Web phishing detection using a deep learning framework. Wireless Communications and Mobile Computing **2018** (2018)
41. Zhang, H., Liu, X., Shao, Y.: Chinese dialect tone's recognition using gated spiking neural P systems. Journal of Membrane Computing **4**(4), 284–292 (2022)
42. Zhang, Gexiang and Zhang, Xihai and Rong, Haina and Paul, Prithwineel and Zhu, Ming and Neri, Ferrante and Ong, Yew-Soon: A layered spiking neural system for classification problems. International Journal of Neural Systems **32**(08), 2250023 (2022)
43. Zhang, Xiaoling and Liu, Xiyu: Multiview Clustering of Adaptive Sparse Representation Based on Coupled P Systems. Entropy **24**(4), 568 (2022)
44. Zhao, S., Zhang, L., Liu, Z., Peng, H., Wang, J.: ConvSNP: A deep learning model embedded with SNP-like neurons. Journal of Membrane Computing **4**(1), 87–95 (2022)