# EVOLUTIONARY NEURAL ARCHITECTURE DESIGN OF LIQUID STATE MACHINE FOR IMAGE CLASSIFICATION

*Cheng Tang*[a] *Junkai Ji*[b] *Qiuzhen Lin*[b] *Yan Zhou*[c]

[a] Faculty of Engineering, University of Toyama, Toyama-shi 930-8555, Japan
[b] College of Computer Science and Software Engineering,
Shenzhen University, Shenzhen 518060, China
[c] State Key Laboratory of Synthetical Automation for Process Industry,
Northeastern University, Shenyang 110167, China

## ABSTRACT

As a recurrent spiking neural network, liquid state machine (LSM) has attracted more and more attention in neuromorphic computing due to its biological plausibility, computation power, and hardware implementation. However, the neural architecture of LSM, such as hidden neuron number, synaptic density, percentage connectivity, and connection state, has significant impact on its model performance. Manually defining a neural architecture will be ineffective and laborious in most cases. Therefore, based on a state-of-the-art differential evolution algorithm, an evolutionary neural architecture design methodology is proposed to automatically build suitable model topologies for LSM in this study, without any prior knowledge. The effectiveness of the proposed method has been validated on commonly-used image classification tasks.

***Index Terms*—** spiking neural network, liquid state machine, neural architecture, evolutionary algorithm, classification

## 1. INTRODUCTION

The spiking neural networks (SNNs) are considered to be the third generation of artificial neural networks, and are promising to mimic the learning and computational functions of the human brain [1]. Liquid state machines LSMs are a recurrent variant of spiking neural networks (SNNs) that have gained attention due to their ability to handle spatiotemporal information, simplicity of structure and low complexity [2]. LSMs are sought after by researchers due to two properties. First, the architectures of LSMs are highly correlated with biological neurons. Compared to the energy consumed by simple pulse-based operations, computers exhibit high computational power but their high-energy design is undoubtedly extremely incompatible with the human brain [3]. Therefore, LSMs certainly provide a new perspective on the design of low-power neuromorphic computation and lay the foundation for realistic artificial brain modeling [4]. Additionally, the architecture of LSM is simpler than other recurrent neural networks (RNNs) and its training complexity is also lower. The complex liquids also have powerful computational capabilities for real-time calculations. While building circuits to achieve powerful computational capabilities is not considered necessary, the impact of architecture on the performance of LSMs is certainly deserving of consideration [5].

The neural architecture search has gained high importance and effectively improved many machine learning methods, such as convolutional neural networks [6] and RNNs [7]. In the improvements of LSM, Wang and Li enhanced LSM by training the connections between neurons in the reservoir [8]. Multiple reservoir layers were considered to form the deep LSM in [9]. Zhou et al. proposed a surrogate-assisted evolutionary search to optimize the neural architecture of LSM [10]. Tian et al. employed the simulated annealing algorithm to improve LSM [11]. Moreover, the genetic algorithms were also adopted to optimize the reservoir generation for LSM [12]. This artificial neural network model with brain-inspired mechanisms was studied to simulate the computational power of the mammalian brain at the cognitive level, which effectively improves the computational efficiency and flexibility. A large number of studies have focused on the learning of synaptic strength. However, both synaptic strength and neural structure play an important role in the nervous system, and neural structure is the hallmark of the mammalian brain. Therefore, the study of neural architecture design is urgent.

In this study, we propose an evolutionary neural architecture design methodology based on a state-of-the-art differential evolution (DE) algorithm, namely, success-history based parameter adaptation for DE (SHADE) algorithm, to automatically construct a suitable model topology for LSM (LSM-SHADE). The handwritten digits dataset is employed to evaluate the performance of the methods. The proposed LSM-SHADE is compared with the conventional LSM and several LSMs optimized by other evolutionary algorithms. The experimental results show that the performance of LSM-SHADE is more effective than other methods.

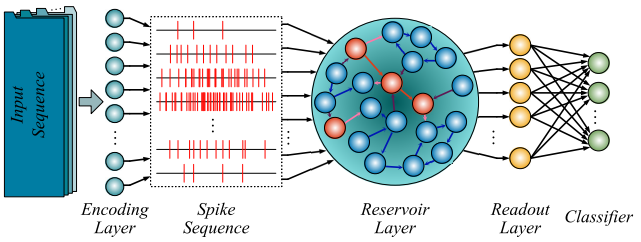


**Fig. 1**. Neural architecture of the liquid state machine.

## 2. MATERIALS

### 2.1. Liquid State Machine

As shown in Fig. 1, LSM mainly consists of four layers: the encoding layer, the reservoir layer, the readout layer, and the classifier.

#### 2.1.1. Encoding Scheme

There are many schemes to transform the real-value input sequences into spike sequences that can be supplied to LSM, namely, rate coding [13], temporal coding [14] and population coding [15]. The square cosine encoding method is employed as the encoding function in the encoding layer [16]. First, the real-value elements are normalized to the range of $[0,\ \pi]$. The normalized values $Nd$ are processed by the encoding layer, and the spike times $t_s$ can be expressed as follows:

$$t_s = T \times \cos\left(Nd + i \cdot \pi/n\right), \tag{1}$$

where $T$ denotes the maximum encoding time. $i \in [1, 2, ..., n]$, and $n$ represents the number of encoding neurons.

#### 2.1.2. Neural Model

The leaky integrate-and-fire model and the spike response model are utilized [17]. The impulse signals are injected into the neural model and the Dirac function is used to generate the pulses. Numerous pulses of $i^{th}$ neuron form the spike train $S_i(t)$, which is described as follows:

$$S_i(t) = \textstyle\sum_f \delta(t - t_{i,f}), \tag{2}$$

where $t_{i,f}$ represents the fire time. The postsynaptic membrane potential $u_i(t)$ at time $t$ is given as follows:

$$u_i(t) = \eta(t - \widehat{t_i}) + \textstyle\sum_j \sum_f \epsilon_{i,j}(t - t_{i,f}), \tag{3}$$

where $\epsilon_{i,j}(t - t_{i,f})$ is the postsynaptic potential, and the kernel function is used, which can be expressed as follows:

$$\kappa_{i,j}(t) = \textstyle\sum_f w_{i,j} \cdot e^{-\frac{t}{\tau_s}} \cdot \delta_{i,j}(t - t_{j,f}) + I_0, \tag{4}$$

where $\tau_s$ denotes the decay time constant. $w_{i,j}$ and $I_0$ are the weights and resting currents. These potentials are accumulated to the $i^{th}$ neurons, which can be described as follows:

$$\tau_n \frac{\mathrm{d}\, u_i(t)}{\mathrm{d}\, t} = -u_i(t) + u_{rest} + \textstyle\sum_j \kappa_{i,j}(t), \tag{5}$$

where $\tau_n$ represents the membrane time constant. The membrane time constant of the excitatory and inhibitory neurons are noted as $\tau_{n-E}$ and $\tau_{n-I}$, respectively. When $u_i$ accumulates to the threshold $u_{th}$, it is reset to $u_{rest}$ and continues for a refractory period $t_{ref}$.

#### 2.1.3. Neural Architecture

The encoding neurons are all excitatory, and the spike trains from the encoding neurons are fed into thirty percent of the neurons in the reservoir layer via the synapses ($S_{En-R}$). In the reservoir layer, eighty percent of the neurons are excitatory neurons and the rest are inhibitory neurons. Each neuron in the reservoir layer has a corresponding neuron in the readout layer that receives its output via the synapse ($S_{R-Read}$). The synapses between neurons are generated randomly, which implies the existence of four types of synapses, namely, $S_{EE}$, $S_{EI}$, $S_{IE}$, and $S_{II}$. The initial strength of the synapses are determined by the gamma distribution and the corresponding mean values are $GM_{En-R}$, $GM_{EE}$, $GM_{EI}$, $GM_{IE}$, $GM_{II}$ and $GM_{R-Read}$, respectively. It is noted that $GM_{R-Read}$ is set to 1 in this study. The time delays from the presynaptic neurons are $t_{delay-E}$ and $t_{delay-I}$, respectively. There is no time delay in both synapse $S_{En-R}$ and synapse $S_{R-Read}$. The probability $P_{i,j}$ of a connection between $i^{th}$ and $j^{th}$ neurons in the reservoir layer is determined by their Euclidean distance $D(i, j)$, which can be defined as follows:

$$P_{i,j} = C \cdot e^{-(D(i,j)/\lambda)^2}, \tag{6}$$

where $C$ represents a constant determined by the type of synapse, namely, $C_{EE}$, $C_{EI}$, $C_{IE}$ and $C_{II}$. $\lambda$ is a parameter that controls the strength of the connection. The probability of connection between the neurons in the encoding layer and the neurons in the reservoir layer is denoted as $P_{En-R}$, which is optimized by the evolutionary algorithms.

For the readout layer, the membrane time constant is set to $\tau_{read}$. The membrane potential $u_{read}$ continues to increase with pulsed stimulation, which represents the states of the neurons in the reservoir layer. Finally, the state of the reservoir $x^{(i)}$ is transmitted to a classifier and the softmax regression model is employed in this study, which is determined by:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} 1\{y^{(i)} = j\} \cdot \log\left(p(y^{(i)} = j|x^{(i)}; \theta)\right) + \frac{1}{2}\alpha\,\|\theta\|^2, \tag{7}$$

where $k$ denotes the number of the features. $y^{(i)}$ represents the output of the softmax function, which is given as follows:

$$p(y^i = j|x^{(i)}; \theta) = e^{\theta_j^T x} / \textstyle\sum_{l=1}^{k} e^{\theta_j^T x}. \tag{8}$$

### 2.2. Success-History Based Parameter Adaptation for DE

In general, the DE algorithm is performed through three phases: mutation, crossover, and selection [18]. In the initial

phase, the population is generated by the following equation:

$$x_{i,j} = x_j^{min} + rand(0,1)(x_j^{max} - x_j^{min}), \quad (9)$$

where $i \in [1, 2, ..., N]$ and $N$ is the population size. $j \in [1, 2, ..., D]$ and $D$ indicates the number of dimensions. $x_j^{min}$ and $x_j^{min}$ are the minimum and maximum boundary values of the $j^{th}$ element, respectively. The solution of the $i^{th}$ individual can be described as $X_i = [x_{i,1}, x_{i,2}, x_{i,3}, ..., x_{i,D-1}, x_{i,D}]$.

The mutation operator is performed to generate a mutated offspring population. The "DE/current-to-best/1" mutation strategy is employed, which is described as follows:

$$V_{i,g} = X_{i,g} + F \cdot (X_{best,g} - X_{i,g}) + F \cdot (X_{r1,g} - X_{r2,g}), \quad (10)$$

where $g \in [1, 2, ..., G]$ and $G$ is the number of generation. $r1$, $r2$, $r3$, $r4$ and $r5$ are the indices of five other individuals. $best$ is the index of the individual with the best current fitness. $F$ denotes a scaling factor.

In the crossover phase, an exponential and binomial crossover operation is performed to generate the trial population $U_{i,g}$, which can be expressed as follows:

$$U_{i,g} = [u_{i,1,g}, u_{i,2,g}, u_{i,3,g}, ..., u_{i,D-1,g}, u_{i,D,g}], \quad (11)$$

where the $j^{th}$ element is defined as follows:

$$u_{i,j,g} = \begin{cases} v_{i,j,g}, & \text{if } rand(0,1) \le CR \text{ or } j = j_{rand} \\ x_{i,j,g}, & \text{otherwise} \end{cases}, \quad (12)$$

where $CR$ represents the crossover rate.

Finnally, the fitness of $X_{i,g}$ and $U_{i,g}$ is compared to confirm which individual will survive to the next generation. For the minimization problems, it can be described as follows:

$$X_{i,g+1} = \begin{cases} U_{i,g}, & \text{if } f(U_{i,g}) \le f(X_{i,g}) \\ X_{i,g}, & \text{otherwise} \end{cases}. \quad (13)$$

In the SHADE, a parameter adaptation scheme is employed, which utilizes historically successful parameters to guide the generation of offspring parameters.

First, a variant of "DE/current-to-best/1" is adopted, which can be described as follows:

$$V_{i,g} = X_{i,g} + F_i \cdot (X_{pbest,g} - X_{i,g}) + F_i \cdot (X_{r1,g} - X_{r2,g}), \quad (14)$$

where greed can be scaled with the parameter $p_i$, and it is given as follows:

$$p_i = rand[p_{min},\ 0.2], \quad (15)$$

where $p_{min}$ is set to $2/N$. Second, the $CR_i$ and $F_i$ can be expressed as follows:

$$CR_i = randn_i(M_{CR,ri}, 0.1), \quad (16)$$

$$F_i = randc_i(M_{F,ri}, 0.1), \quad (17)$$

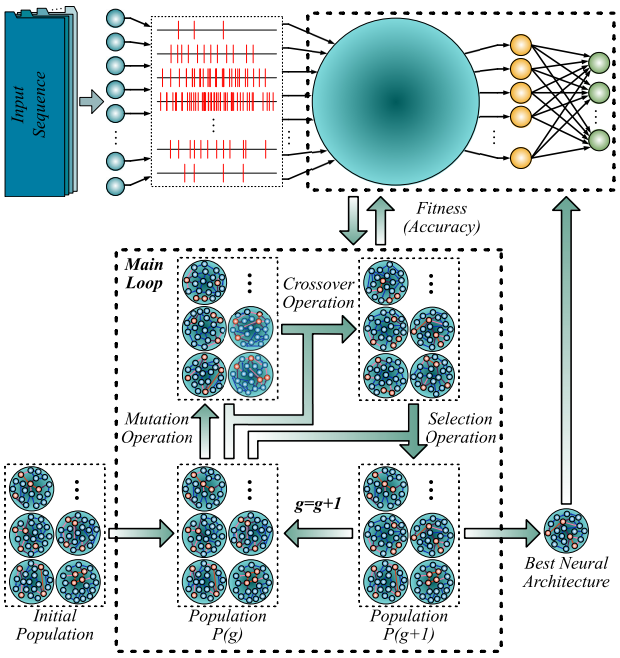


**Fig. 2**. Flowchart of the evolutionary neural architecture design methodology.

$$M_{CR,k,g+1} = \begin{cases} \sum_{k=1}^{|S_{CR}|} w_k \cdot S_{CR,k}, & if\ S_{CR} \neq \varnothing \\ M_{CR,k,g}, & otherwise \end{cases}, \quad (18)$$

$$M_{F,k,g+1} = \begin{cases} \frac{\sum_{k=1}^{|S_F|} w_k \cdot S^2_{F,k}}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F,k}}, & if\ S_F \neq \varnothing \\ M_{CR,k,g}, & otherwise \end{cases}, \quad (19)$$

$$w_k = \left| f(U_{k,g}) - f(X_{k,g}) \right| / \sum_{k=1}^{|S_{CR}|} \triangle f_k, \quad (20)$$

where $CR_i$ and $F_i$ are preserved in memory as $S_{CR}$ and $S_F$, respectively, when $U_{i,g}$ is superior to $X_{i,g}$. More detailed information can be referred to [19].

### 2.3. Evolutionary Neural Architecture Methodology

Nine hyperparameters determine the neural architecture of LSM, namely, $\tau_{n-E}$, $\tau_{n-I}$, $GM_{En-R}$, $GM_{EE}$, $GM_{EI}$, $GM_{IE}$, $GM_{II}$, $\lambda$ and $P_{En-R}$. These hyperparameters have a significant effect on the model performance of LSM. However, manually choosing the suitable hyperparameters of LSM is always time-consuming and inefficient. In this study, an evolutionary neural architecture design methodology based on the SHADE, is proposed to automatically build suitable model topologies for LSM in this study, without any prior knowledge. The flowchart of methodology has been illustrated in Fig. 2. The ranges of the hyperparameters are presented in Table 1. To ensure the fairness of the comparison, the population size of all evolutionary algorithms is set to 20, and the maximum number of function evaluations is set to 400.

**Table 1**. The hyperparameters in LSM.

| Parameter | Initial value | Range |
|---|---|---|
| $\tau_{n-E}, \tau_{n-I}$ | 15 | (0, 30) |
| $GM_{En-R}, GM_{EE}, GM_{EI}, GM_{IE}, GM_{II}$ | 30 | (0, 60) |
| $\lambda$ | 0.5 | (0, 1) |
| $P_{En-R}$ | 0.05 | (0, 0.1) |
| **Parameter** | **Default value** | |
| $u_i, u_{th}, u_{rest}, u_{read}$ | 13.5, 15, 13.5, 0 | |
| $\kappa_{i,j}$ | 0 | |
| $I_0$ | 13.5 | |
| $\tau_{read}, \tau_{s-E}, \tau_{s-I}$ | 30, 3, 6 | |
| $t_{ref-E}, t_{ref-I}, t_{delay-E}, t_{delay-I}$ | 3, 2, 0.8, 1.5 | |
| $C_{EE}, C_{EI}, C_{IE}, C_{II}$ | 0.3, 0.2, 0.4, 0.1 | |

## 3. EXPERIMENTS AND ANALYSIS

### 3.1. Experimental Setup

In our experiments, the modified national institute of standards and technology (MNIST) dataset is adopted to evaluate the performance of LSM-DE. The MNIST is a huge dataset of handwritten digits, which consists of 60,000 training images and 10,000 test images [20]. In the MNIST dataset, the image size is $28 \times 28$ pixels, which is converted into $784 \times 1$ input sequence in this study. The batch size is set to 1,200, which implies that 1,200 images are randomly selected to train the models in one iteration. 400 images from the training data are utilized for validation, and 400 images from the test data are used to assess the performance.

### 3.2. Performance Comparisons

In this subsection, LSM optimized by the SHADE (LSM-SHADE) is compared with conventional LSM, LSM optimized by the genetic algorithm (LSM-GA), LSM optimized by the particle swarm optimization (LSM-PSO), LSM optimized by the differential evolution (LSM-DE). The results have been presented in Table 2. It is evident that LSM-SHADE performs better than other methods. LSM-SHADE achieves the accuracy of 94.50% on the MNIST dataset, which is higher than the conventional LSM. It suggests that the evolutionary neural architecture design methodology is effective in tuning the hyperparameters of LSM automatically. The results of LSM-SHADE are also better than LSM-GA, LSM-PSO and LSM-DE. It is because that the SHADE has a more powerful search ability than the GA, PSO and DE algorithms.

Furthermore, for a more intuitive comparison between LSM-SHADE and LSMs optimized by other evolutionary algorithms, the convergence plots of the accuracy for the four methods are depicted in Fig. 3. Since the population size is 20, the comparison step is also set to 20 evaluations. Observing the accuracy in each iteration, namely the 20 results, can provide a more comprehensive perspective for detailed comparisons. In each iteration, LSM-SHADE has led the way and ultimately achieved the best performance. Therefore, we can conclude that the SHADE-based evolutionary neural architecture design methodology can construct a more suitable model topology for LSM.

**Table 2**. Comparison of LSMs trained by different evolutionary algorithms.

| Model | LSM | LSM-GA | LSM-PSO | LSM-DE | LSM-SHADE |
|---|---|---|---|---|---|
| Accuracy | 80.75% | 91.75% | 92.25% | 93.25% | 94.50% |

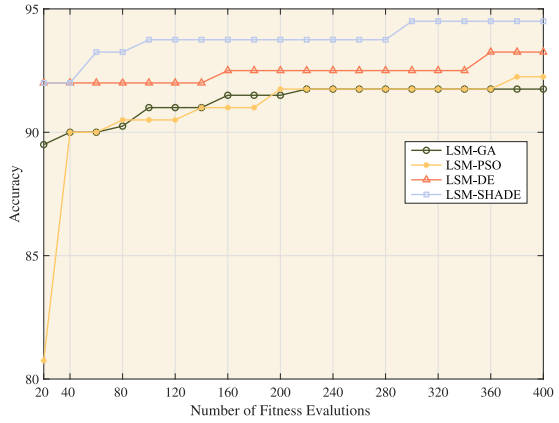


**Fig. 3**. Comparison of LSMs trained by different evolutionary algorithms.

## 4. CONCLUSION

Due to the neural dynamics instability of the spiking neurons and the cyclical character of the connections, training LSMs is challenging. It is implausible that conventional LSM create synapses and connections randomly in the reservoir. Therefore, an evolutionary neural architecture design methodology based on the SHADE algorithm is proposed for LSM, which can automatically optimize its neural architecture on diverse tasks. The proposed LSM-SHADE has been applied to the real-world MNIST dataset. The results compared to the conventional LSM give strong support for the validity of the neural architecture search. Compared with LSMs optimized by other evolutionary algorithms, benefiting from the powerful search capabilities of the SHADE, LSM-SHADE achieved the best performances. In the SHADE algorithm, adaptive mechanisms and historically successful populations are used to adjust the scale factor and crossover rate. The neural architecture design of LSM is optimized using SHADE, which can avoid local minima and significantly improve the accuracy of LSM. However, relevant experimental results show that the evolutionary algorithm cannot provide satisfactory optimization performance on higher dimensional benchmark problems. Our method is limited in the number of total optimization parameters, which requires more attention to address this issue in our future research. In addition, the neural architecture search for LSM models with multiple reservoir layers and deep LSM models is also worth investigating.

## 5. REFERENCES


[1] Catherine D Schuman, Thomas E Potok, Robert M Patton, J Douglas Birdwell, Mark E Dean, Garrett S Rose, and James S Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[2] Wolfgang Maass, Thomas Natschläger, and Henry Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[3] Jose M Cruz-Albrecht, Michael W Yung, and Narayan Srinivasa, "Energy-efficient neuron, synapse and stdp integrated circuits," *IEEE transactions on biomedical circuits and systems*, vol. 6, no. 3, pp. 246–256, 2012.

[4] Parami Wijesinghe, Aayush Ankit, Abhronil Sengupta, and Kaushik Roy, "An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 5, pp. 345–358, 2018.

[5] Parami Wijesinghe, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy, "Analysis of liquid ensembles for enhancing the performance and accuracy of liquid state machines," *Frontiers in neuroscience*, vol. 13, pp. 504, 2019.

[6] Wei Peng, Xiaopeng Hong, Haoyu Chen, and Guoying Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 2669–2676.

[7] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[8] Qian Wang, Yongtae Kim, and Peng Li, "Architectural design exploration for neuromorphic processors with memristive synapses," in *14th IEEE International Conference on Nanotechnology*. IEEE, 2014, pp. 962–966.

[9] Nicholas Soures and Dhireesha Kudithipudi, "Deep liquid state machines with neural plasticity for video activity recognition," *Frontiers in neuroscience*, vol. 13, pp. 686, 2019.

[10] Yan Zhou, Yaochu Jin, and Jinliang Ding, "Surrogate-assisted evolutionary search of spiking neural architectures in liquid state machines," *Neurocomputing*, vol. 406, pp. 12–23, 2020.

[11] Shuo Tian, Lianhua Qu, Lei Wang, Kai Hu, Nan Li, and Weixia Xu, "A neural architecture search based framework for liquid state machine design," *Neurocomputing*, vol. 443, pp. 174–182, 2021.

[12] John JM Reynolds, James S Plank, and Catherine D Schuman, "Intelligent reservoir generation for liquid state machines using evolutionary optimization," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[13] Michael D Forrest, "The sodium-potassium pump is an information processing element in brain computation," *Frontiers in physiology*, vol. 5, pp. 472, 2014.

[14] Nitin Gupta and Mark Stopfer, "A temporal channel for information in sparse sensory coding," *Current Biology*, vol. 24, no. 19, pp. 2247–2256, 2014.

[15] Si Wu, Shun-ichi Amari, and Hiroyuki Nakahara, "Population coding and decoding in a neural field: a computational study," *Neural Computation*, vol. 14, no. 5, pp. 999–1026, 2002.

[16] QX Wu, T Martin McGinnity, Liam P Maguire, B Glackin, and Ammar Belatreche, "Learning under weight constraints in networks of temporal encoding spiking neurons," *Neurocomputing*, vol. 69, no. 16-18, pp. 1912–1922, 2006.

[17] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski, *Neuronal dynamics: From single neurons to networks and models of cognition*, Cambridge University Press, 2014.

[18] Swagatam Das and Ponnuthurai Nagaratnam Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE transactions on evolutionary computation*, vol. 15, no. 1, pp. 4–31, 2010.

[19] Ryoji Tanabe and Alex Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE congress on evolutionary computation*. IEEE, 2013, pp. 71–78.

[20] Yann LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.