Journal of Network Intelligence
Taiwan Ubiquitous Information

Volume **9**, Number **1**, February **2024**

An Improved Deep Spiking Neural Network Model Based on Bionic Optimization Algorithm

Xiaodong Wang, Yaqin Wu*

School of Computer Information
Inner Mongolia Medical University
Hohhot 010110, China
wxdchg@126.com, wuyq_06@126.com

*Corresponding author: Yaqin Wu
Received February 21, 2023, revised March 24, 2023, accepted May 20, 2023.

Abstract. *With the rapid development of Internet technology, data has new characteristics (Volume, Velocity, Variety, Value) that make it difficult for traditional classification algorithms to achieve high classification accuracy. Inspired by Spiking-based communication in the brain, Deep Spiking Neural Network (DSNN) uses a Spiking neuron model that mimics biological neuronal mechanisms for computation, with the ability to handle complex time-series data, very low energy consumption, and high robustness. Therefore, to test the solution performance of DSNN, a DSNN model and a modified deep DSNN model were used for multi-label classification evaluation, respectively. First, the deep Spiking response was combined with a convolutional neural network (CNN) to construct a deep Spiking neural network model. Then, the Bacterial Foraging Optimization (BFO) individuals were constructed with the weights, dynamic thresholds, and forgetting parameters of the DSNN, and the multi-label classification accuracy was used as the fitness function of the BFO algorithm. Secondly, the optimal individuals were obtained by continuously updating the fitness values through repellent, reproduction, and migration operations, and the DSNN parameters corresponding to the optimal individuals were saved. The simulation results show that by reasonably setting the gravitational and repulsive coefficients and migration probability threshold of the BFO algorithm, the multi-label classification of the DSCNN model achieves a high classification accuracy, and the BFO-DSNN algorithm has a better classification performance when compared with the multi-label classification algorithms of commonly used deep neural networks.*
Keywords: Bionic intelligence algorithm, Bacterial foraging optimization, Spiking neural network, Multi-label classification, Artificial intelligence

1. **Introduction.** Neural network algorithms are an important class of classification algorithms [1,2,3]. The basic idea is to mimic the work of the human brain as closely as possible, and its development has gone through two important stages, from shallow neural networks with only three layers of network structure to deep neural networks with hundreds of layers of network structure, and its performance has also achieved a qualitative leap.

Deep convolutional neural networks, a feed-forward neural network with a multi-layer structure [4,5] and a unique convolutional and pooling structure, were the first successful deep structural models. The first steps in the study of convolutional neural networks were taken in 1962 when Hubel and Wiesel published a research paper on the visual cortex of monkeys and birds. in the 1980s Kunihiko introduced the convolutional process into

the field of CNNs, named neocognitron. currently, deep convolutional neural networks (DCNNs) are successful in many applications [6,7].

Inspired by Spiking-based communication in the brain [8,9], Deep Spiking Neural Network (DSNN) uses Spiking neuron models that fit biological neuronal mechanisms for computation, with the ability to handle complex temporal data, extremely low energy consumption and high robustness. SNNs have received a lot of attention in the field of neuromorphic engineering and brain-like computing, and have been hailed as the next generation of neural networks [10,11]. Action Potential) in biological neurons [12].

Theoretical analysis shows that SNNs are comparable to conventional neuron models in terms of computational performance. Due to its ability to handle complex time-series data, its extremely low energy consumption and its deep physiological foundation [13], SNNs have received a lot of attention from scholars and have made rapid progress in image classification, target recognition, speech recognition and other fields. Recent studies have shown that SNNs approach or achieve comparable performance to classical Artificial Neural Networks (ANNs) in many areas. SNNs also exhibit greater robustness than ANNs. Firstly, the randomness in the dynamics of Spiking neurons can improve the robustness of the network to external noise [14]. Second, recent studies have shown that the issuing mechanism of Spiking neurons makes SNNs inherently robust against adversarial attacks [15].

Although the current research on DCNN algorithms in the big data environment has achieved some success, SNNs have been applied less in this field. In addition, DSNN suffers from too many redundant parameters in the network and poor parameter finding ability [16]. The Bacterial Foraging Optimization (BFO) algorithm [17] is a bioheuristic-based bionic intelligence algorithm, which has certain advantages in model parameter finding. For example, the BFO algorithm simulates real bacterial behaviour through the life cycle and movement process of colonies, which can effectively avoid local optimal solutions and has global search capability. At the same time, the BFO algorithm is suitable for high-latitude search problems, is not limited by the number of dimensions, and can complete the optimisation in a shorter time. The BFO algorithm is simple to understand, easy to implement, does not require advanced mathematical foundation, and can quickly achieve the optimisation solution [18]. Therefore, this work uses the BFO algorithm to find the optimal parameters of DSNN, thus improving its performance in multi-label classification tasks.

1.1. **Related Work.** SNN achieves data transfer by encoding the Spiking to be trained samples and the resulting Spiking responses, which enables a much larger data transfer compared to traditional ANNs because their Spiking sequences contain not only sample data but also rich temporal data.

The results of Burrows et al. [19] showed that in the multilayer operation of DSNNs, only when a spiking response occurs, the neurons inside the network perform the conversion computation, thus making it more resource efficient. The more powerful complex computation of DSNNs facilitates the extension of this structural model to applications in several industry domains. For example, Karekal et al. [20] proposed a DSNN containing multiple synapses to solve the problem of learning multiple distal rewards simultaneously. Liu et al. [21] first applied ANN-SNN conversion to the field of reinforcement learning, avoiding the difficulty of training SNNs directly for reinforcement learning, and demonstrated that SNNs can improve the robustness of the model to occlusion. However, the determination of the DSNN model also requires more parameters, such as neuron connection weights, ignition thresholds and forgetting variables, so the training solution of DSNN is more complex than that of traditional neural networks. In the model solving of

DSNN, Gao et al. [22] proposed to determine a stable SNN model by taking the minimum value of the difference between the ignition expectation and the actual ignition time as the objective function, and determining the parameters corresponding to this minimum value when it is obtained.

Recently, researchers have attempted to optimally solve DSNN parameters with the help of bionic intelligence algorithms. Madhiarasan and Deepa [23] proposed an improved DSNN model based on the Grey Wolf Optimization (GWO) algorithm and achieved more accurate long-term wind speed predictions. The results of Xu et al. [24] showed that compared to the GWO algorithm, the BFO algorithm has more advantages in terms of functional form independence, dependence on initial values, no need to maintain parameters and better robustness. This is because the GWO algorithm requires maintenance of parameters, such as alpha, beta, etc., whereas the optimisation process of the BFO algorithm does not require maintenance of these parameters.

Therefore, this work evaluates the classification performance of DSNN in terms of multi-label classification. The BFO algorithm is used to optimise the core parameters of the DSNN model when solving for them. The simulation experimental results show that the BFO algorithm effectively improves the classification performance of the deep Spiking neural network.

1.2. **Motivation and contribution.** . Since artificial neural networks have shown performance breakthroughs in the areas of target detection and scene classification, DSNNs in particular have excellent performance. Therefore, this work focuses on finding optimal parameters for DSNN models.

The main innovations and contributions of this work include:

(1) In order to reduce the probability of local convergence of the BFO algorithm, this work brings the idea of variation (mutation factor) from genetic algorithms into the BFO algorithm. The proposed adaptive BFO algorithm has a high convergence speed and accuracy, while reducing the probability of local convergence to a certain extent.

(2) Since the ANN-SNN conversion process requires scaling the weights of the neural network, the scaling ratio of each layer of weights can be regarded as an independent hyperparameter to be optimised. In order to reduce the errors arising from the conversion process and maximize the performance of the DSNN, this work uses the adaptive BFO algorithm to find the optimal scaling parameters for the DSNN, thereby improving the training efficiency and classification accuracy of the DSNN.

2. Spiking neural networks.

2.1. **Spiking neuron models.** Spiking neural networks, as third generation neural networks, use biologically sound Spiking neuron models and corresponding learning algorithms to model the neuronal structure and synaptic plasticity in the brain.

In living organisms, the typical structure of a neuron consists of three main parts: the dendrites, the cytosol and the axons [25]. The dendrites collect input signals from other neurons and transmit them to the cytosol; the cytosol acts as a central processor, generating Spiking (i.e. action potentials) when the accumulation of received afferent currents causes the neuron's membrane potential to exceed a certain threshold; the axon propagates Spiking and transmits the signal to the next neuron via the terminal synaptic structure. The basic neuron of the SNN is shown in Figure 1. In computer simulation, the continuous differential equations for each variable in the Spiking neuron model need to be discretized in time first, and then the corresponding variables are iteratively updated according to the discrete time steps. For the discrete time steps, when a smaller time interval is chosen, the model changes more closely to the original continuous differential

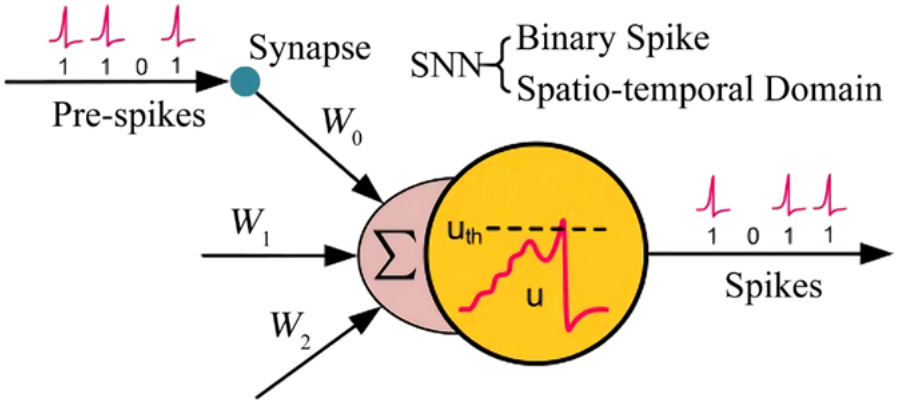

Figure 1. Basic neurons of the SNN

equations, with less loss of accuracy. However, this also increases the number of discrete time steps under the same conditions, thus increasing the computational complexity. Therefore, there is a trade-off between accuracy and computational complexity in discrete Spiking neuron models. In general, the discrete Spiking neuron model can be described as three discrete equations for charging, discharging and resetting.

$$H_t = f\left(V_{t-1}, X_t\right) \tag{1}$$

$$S_t = \Theta\left(H_t - V_{th}\right) \tag{2}$$

$$V_t = H_t\left(1 - S_t\right) + V_{\text{reset}}\, S_t \tag{3}$$

where H_t denote the membrane potential after charging at moment t and V_t denote the membrane potential after resetting. $f(\cdot)$ defines the specific neuron charging equation, which varies considerably between different Spiking neuron models. $\Theta(\cdot)$ is a step function, which is generally defined as

$$\Theta(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

During charging, the neuron receives exogenous input X_t and updates the membrane potential; during discharging, the neuron outputs a Spiking signal S_t when the membrane potential exceeds a threshold V_{th}; during resetting, the neuron updates the membrane potential in response to the Spiking signal. A generic discrete Spiking neuron model is shown in Figure 2.

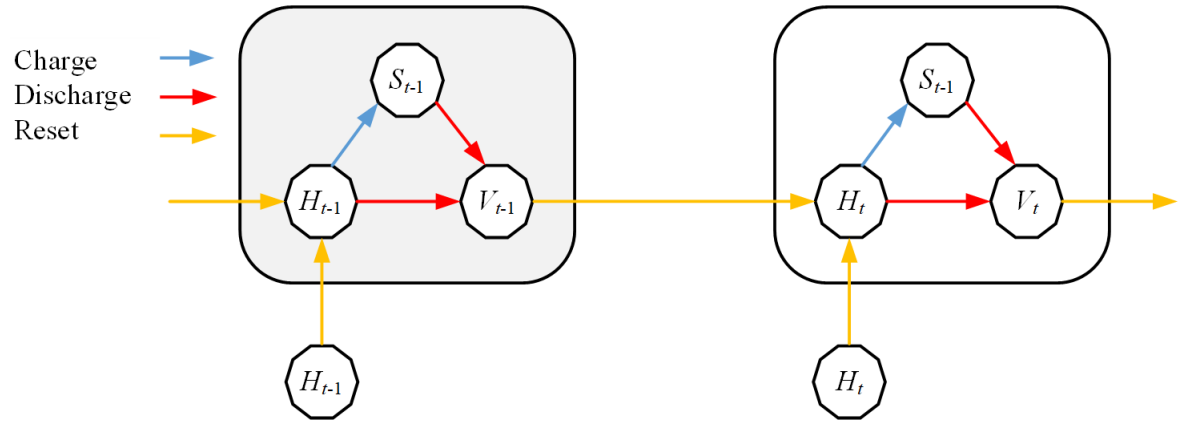

Figure 2. Generalized discrete Spiking neuron model

2.2. **Spiking encoding patterns.** The perception of the world around us, the regulation of endostasis and the behavioural response to sensory stimuli are all transmitted by neuronal Spiking sequences. The way in which neurons store information in Spiking sequences is one of the most important questions discussed in the field of neuroscience.

Based on the physiological findings obtained from the brain, the research community has proposed a number of different Spiking coding styles. The main common Spiking coding methods include Rate Coding, Temporal Coding, Bursting Coding and Population Coding.

Rate Coding focuses on the Spiking delivery rate of a neuron, i.e. the mean number of Spiking deliveries issued by a neuron over the duration of the simulation. The Spiking rate of a neuron reflects the strength of the input stimulus, and temporal coding is more concerned with differences in the temporal structure of the Spiking sequence than frequency coding. In addition to the complete Spiking delivery time information, the time from receipt of the stimulus to delivery of the first Spiking and the temporal logic between Spikings are considered to encode In addition to these two common temporal coding schemes, temporal coding schemes include Latency Coding, Phase Coding, Synchrony Coding and Polychronisation Coding, among others [26].

Explosive coding considers the encoding of information using the behaviour of neurons firing explosively. During burst-type firing, neurons rapidly dispense large amounts of Spiking over a period of time, and then remain resting for a longer period of time. In contrast to frequency coding, temporal coding, etc., which focuses more on the activity of individual neurons [27], population coding considers the information generated by a stimulus to be characterised by the joint activity of multiple neurons.

Let the voltage of the posterior synaptic neuron j in the Spiking neural network be $u(t)$. When $u(t)$ is less than the threshold τ, an output $t_j^{(f)}$ is triggered with the output Spiking sequence of

$$F_j = \{t_j^{(f)}; 1 \leq f \leq n = t | u_j(t) = \tau\} \tag{5}$$

Assuming that all anterior synaptic neurons i are in the set Γ_j, ignition of Spiking will affect $u(t)$.

$$P_j(t) = \sum_{i \in \Gamma_j} \sum_{t_j^{(f)} \in F_i} w_{ij} \varepsilon_{ij}(t - \hat{t}_j, t - t_j^{(f)} - \Delta_{ij}^{ax}) + U_j^{ext} \tag{6}$$

where w_{ij} is the weight, U_j^{ext} is the voltage change, ε_{ij} is the synaptic potential, $\hat{t}_j$ is the Spiking output time and Δ_{ij}^{ax} is the delay term.

Let $s = t - t_j^{(f)} - \Delta_{ij}^{ax}$ and use u_{rest} for the resting potential, the change before and after ignition will be $\eta_j(t - \hat{t}_j)$.

In practice, the above equation is too complex to calculate and a simplified Spiking response model is often used for the calculation.

$$u_j(t) = \eta_j(t - \hat{t}_j) + \sum_{i \in \Gamma_j} \sum_{t_i^{(f)} \in F_i} w_{ij} \varepsilon_{ij}(t - t_j^{(f)}) + U_j^{ext} \tag{7}$$

The Spiking response of the posterior synaptic neuron j is represented by

$$u_j(t) = \eta_j(t - \hat{t}_j)) + P_j(t) \tag{8}$$

The synaptic potential $\varepsilon_{ij}(t)$ is calculated as

$$\varepsilon_{ij}(t) = \frac{t - t_i^{(f)} - \Delta_{ij}^{ax}}{\tau} \exp(1 - \frac{t - t_i^{(f)} - \Delta_{ij}^{ax}}{\sigma}) H(t - t_i^{(f)} - \Delta_{ij}^{ax}) \tag{9}$$

where $H(\cdot)$ is the step response and σ is a constant.

3. **BFO algorithm.**

3.1. **Basic principles.** Bacterial Foraging Optimization is a bionic intelligence algorithm that simulates the foraging behaviour of Escherichia coli.

According to biological studies, the foraging process of E. coli begins by searching for areas where food may be present. Then, the selected area is judged whether to enter or not, and if it decides to enter, it looks for food in that area, otherwise it continues to look for possible areas. After acquiring some food in the selected area, the area is judged whether to migrate to other areas. Each step of the bacterial population is based on the selection of its own conditions and environment, thus allowing the bacteria to obtain the most food per unit time. Assume that S individual bacteria perform movement operations in the interval [min, max]. Initialise the location of the bacterial swarm at P as follows:

$$P = \min + rand * (\max - \min) \tag{10}$$

Bacterial individuals perform 3 main types of operations: repellent, reproduction and migration. Let the position of bacteria i be $X_i = (x_1, x_2, \ldots, x_v)$, where v denotes the position dimension. The position from position P after the jth repellent operation is shown below:

$$\mathbf{X}_i^{j+1} = \mathbf{X}_i^j + C(i)\frac{\mathbf{\Delta(i)}}{\sqrt{\mathbf{\Delta^T(i)\Delta(i)}}} \tag{11}$$

where $C(i)$ and $\Delta(i)$ are the repellent step and direction vector for bacteria i, respectively.

The degree of adaptation after the Roundup operation is shown below:

$$J(\mathbf{X}_i) = \sum_{i=1}^{S} J_{attract}^i + \sum_{i=1}^{S} J_{repellant}^i \tag{12}$$

where J^i_{attract} and $J^i_{\text{repellant}}$ denote the gravitational and repulsive forces of the bacterial population, respectively.

$$\sum_{i=1}^{S} J_{attract}^i = \sum_{i=1}^{S} [-d_{attract} \exp(-w_{attract} \sum_{m=1}^{v} (x_m - x_m^i)^2)] \tag{13}$$

where d_{attract} and w_{attract} are the longitudinal and transverse components of the gravitational force, respectively.

$$\sum_{i=1}^{S} J_{repellant}^i = \sum_{i=1}^{S} [-h_{repellant} \exp(-w_{repellant} \sum_{m=1}^{v} (x_m - x_m^i)^2)] \tag{14}$$

where $h_{\text{repellant}}$ and $w_{\text{repellant}}$ are the longitudinal and lateral components of repulsion, respectively, x_m is the m-th dimensional component of all individuals, and x_m^i is the m-th dimensional component of bacteria i.

The fitness is updated after each repellent operation and the total fitness of individual i after several iterations $H(X_i)$ is shown as follow:

$$H(\mathbf{X}_i) = \sum_{iter} J(\mathbf{X}_i) \tag{15}$$

A new bacterial population location is generated by selecting the higher bacterial individuals for the reproduction operation according to the descending order of fitness. According to Equation (15), individuals with higher fitness are retained for reproduction. The three types of movement are alternated until the maximum number of iterations is reached.

3.2. **Adaptive BFO.** The step size $C(i)$ is a key parameter in the evolutionary process. If the step size is too large, the bacteria can move faster towards the target but tend to ignore the optimal value or fall into a local optimum. If the step size is too small, the search will be less efficient and will tend to fall into local convergence.

In the early stage of the search, a larger step size is needed to quickly find the region where the optimal solution is located and to improve the global search capability of the algorithm. And in the later stage of the search with smaller step size, it is bacteria with stronger local search ability. Therefore, Adaptive Bacterial Foraging Optimization (ABFO) is proposed in this work.

Step 1: Initialize the sensitivity V.

$$V = \frac{J_i}{J_{\max}} (X_{\max} - X_{\min}) \cdot rand \tag{16}$$

where J is the adaptation value.

Step 2: Perform updates of the bacterial locations and corresponding adaptation values by generating a unit random vector.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(j) \tag{17}$$

Step 3: If the adaptation values improve after the above flip, swim in the direction of the flip, otherwise leave it the same.

Step 4: The sensitivity is linearly decreasing according to the following rule.

$$V = \frac{step_{\max} - step_{\min}}{step_{\max}} \cdot V \tag{18}$$

However, there is still a significant possibility of local convergence in the way the step size is updated by linear decrement. To reduce this possibility, this work introduces the concept of variation in genetic algorithms into ABFO.

$$\eta = \frac{J_i}{\bar{\bar{J}}} \tag{19}$$

Where η is the mutation variance factor and $\bar{J}$ is the average fitness value of the colony.

After initiating the mutation factor, the step size of the bacterial population will mutate with probability P_m.

$$C(i) = \eta \left(C_{i\max} - C(i)\right) + C(i) \tag{20}$$

By introducing the colony step mutation as a measure, the ABFO algorithm is able to break out of the local convergence trap to some extent.

In order to verify the performance of the ABFO algorithm, three common standard non-linear test functions were selected for simulation and compared with PSO and BFO. The parameters for each experiment: population size of 50, dimensionality of 30 and maximum number of iterations of 300. The test functions are shown in Table 1:

Table 1. Three test functions.

Name	**Expressions**	**Initial range**	**BestT**	**Daily accuracy**
Sphere function	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$(50, 100)^{30}$	0	0.00001
Rastrigrin function	$f_2(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$(2.56, 5.12)^{30}$	0	100
Griewank functions	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$(300, 600)^{30}$	0	0.05

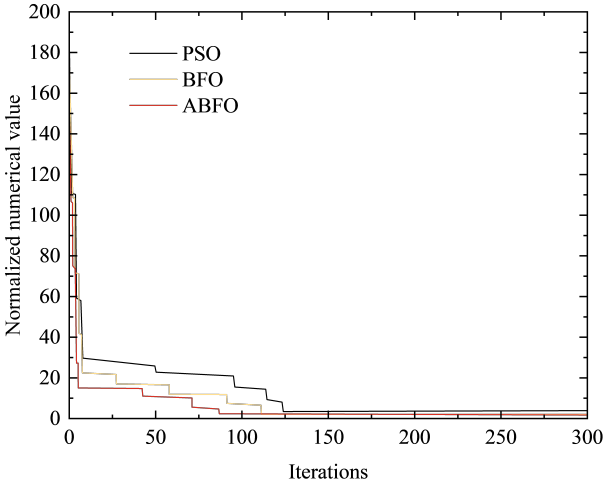

Figure 3. Sphere function

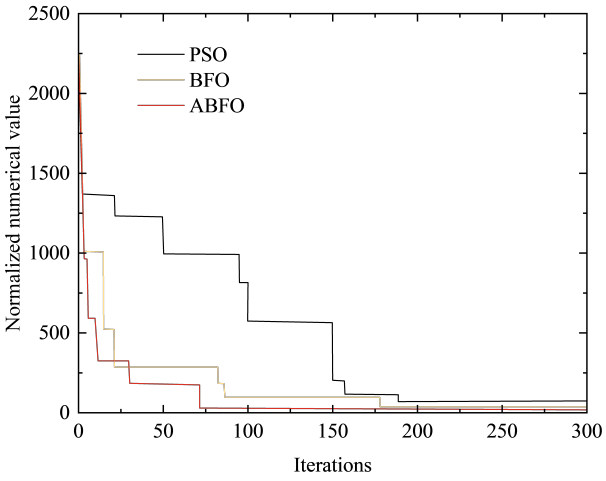

Figure 4. Rastrigrin function

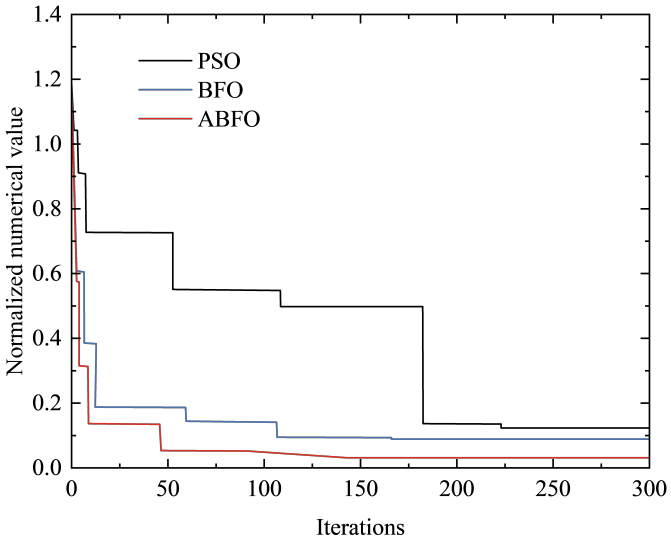

Figure 5. Griewan function

The convergence curves of the two bionic intelligence algorithms are shown in Figure 3, Figure 4 and Figure 5 respectively.

The convergence results of the five bionic intelligence algorithms are shown in Table 2. From the simulation results, it can be seen that ABFO has greatly improved both in terms of convergence speed and convergence accuracy, while effectively reducing the probability of entering local convergence.

Table 2. Test function convergence point.

	PSO		BFO		ABFO	
	Iteration	**Value**	**Iteration**	**Value**	**Iteration**	**Value**
Sphere	143	3.812	126	1.204	113	1.102
Rarigrin	170	12.9343	155	8.9024	70	5.9124
Griewan	223	0.1232	166	0.0887	143	0.0314

4. **ABFO-based DSNN model.**

4.1. **Learning approaches.** Currently, there is no uniform learning algorithm in the field of impulsive neural networks. Learning algorithms for impulsive neural networks vary considerably due to the different emphasis on biological plausibility and task performance, as well as the model of impulsive neurons in the network and the way the impulses are encoded.

With the rise of deep learning, the need to make the layers of impulse neural networks deeper has become a major challenge due to the need to handle complex tasks as well as performance considerations. To solve the problem of network depth, the ANN-SNN transformation was proposed. In recent years, the training techniques of ANNs have become increasingly mature. Compared with ANN, training SNN currently requires not only a large amount of computational resources, but also has to overcome the difficulty that the pulse release function is not trivial. In order to avoid the difficulties of training SNNs and to take advantage of the data processing advantages of SNNs, the ANN-SNN transformation has become a feasible method to convert trained ANNs into SNNs by using the firing rate of each impulse neuron to approximate the corresponding ReLU activation values in artificial neurons. It is worth noting that the firing rate of pulse neurons here refers to the quotient of the total number of pulses fired in a discrete simulation period over the duration of the simulation. It was demonstrated that IF neurons in the Soft Reset setting are an unbiased estimate of the ReLU activation function in time [28].

In simulation, the firing rate of a pulse neuron is usually defined as the quotient between the total number of pulses fired by the pulse neuron and the duration of the simulation. Thus, this firing rate can be taken to be in the range of 0 to 1, whereas the continuous values involved in reinforcement learning (continuous actions, Q-values, etc.) are often not limited in their range and can be taken to any floating point value between plus and minus infinity. Even if there is a range restriction on these continuous values, it is often too difficult to know in advance to represent them in terms of the scaling results of the release rate. It is this difference in the range of values that greatly limits the scope of application of frequency encoding based impulse reinforcement learning algorithms. However, by combining the training benefits of ANN networks with the low energy inference of SNN networks, no training of SNNs is required. The ANN-SNN transformation is able to achieve performance comparable to that of the corresponding deep network. In Spiking's impulse model, the execution of its impulse response is often combined with different types of neural networks to enhance the computational power of the Spiking neural network, and this work brings together Spiking impulses with convolutional neural networks to form the SCNN model structure.

The feature sampling of Spiking is first performed, with the j-th node corresponding to the k-th layer sampled in the following manner:

$$X_j^k = f(\frac{1}{n}\sum_{i\in M_j} X_i^{k-1} + B^k) \tag{21}$$

where n is the sample size and B is the bias.

Sampling and entering the convolution layer operation can be expressed as

$$X_j^k = f(\sum_{i\in M_j} T(X_i^{k-1}) * Kernel_{ij}^k + B^k) \tag{22}$$

where T is the encoding function, Kernel is the convolution kernel, and M_j is the j-th feature map.

The conversion function for sampling and convolution f can be expressed as

$$f(z) = \frac{1}{1+\exp(-z)} \tag{23}$$

The output of DSNN $t_j^{(f)}$ can be expressed as

$$t_j^{(f)} : f(t_j^{(f)}) = \theta, \frac{df(t)}{dt}|_{t_j^{(f)}} > 0 \tag{24}$$

4.2. **The flow of the model.** As the ANN-SNN conversion process requires scaling the weights of the neural network, the scaling of each layer of weights can be considered as an independent hyperparameter to be optimised.

In order to reduce the errors that occur in the conversion process and maximise the performance of the DSNN, this work uses the ABFO algorithm to find the optimal scaling parameters for the DSNN, thereby improving the training efficiency and classification accuracy of the DSNN. The following are the steps for optimising the weight parameters of the DSNN model using the ABFO algorithm:

Step 1: Determine the fitness function. For the DSNN model, the fitness function can be a weighted sum of the accuracy and Spike Count. Assuming that the DSNN model has a total of n neurons and each neuron produces an output of $S(i)$ and the actual output is $T(i)$, the accuracy rate can be expressed as

$$accuracy = (1/n) \cdot \sum (\ T\,(i) == S\,(i) \ ) \tag{25}$$

The Spike Count can be expressed as

$$spike_{count} = \ \sum S\,(i) \tag{26}$$

The adaptability function can be expressed as

$$fitness \ = \ w_1 \cdot accuracy \ + \ w_2 \cdot spike_{count} \tag{27}$$

where w_1 and w_2 are the weights of the two targets respectively.

Step 2: Initialise the bacterial population. The number of bacterial populations is set to N, and the position and trophic state of each individual is initialised. The position is used to represent the value of each weight in the DSNN model, and the trophic state refers to the value of each individual's fitness function under the current model.

Step 3: Nutritional status update. For each individual, the nutritional status is calculated based on the fitness function under the current model. You can use the previously defined fitness as the nutritional status of the individual.

Step 4: Move mode selection. For each individual, select the movement mode. There are two types of movement patterns: single-step and multi-step. In principle, you should choose multi-step movement whenever possible to avoid duplication of searches in the search space.

Single step moves: for each weight value, a nearby random value is chosen for replacement. For the i-th weight, the new value is

$$X\,(i) = X\,(i) + rand\,(2 \cdot r) - r \tag{28}$$

where r is the range of movement.

Multi-step moves: a vector is randomly generated and the weight vector is updated using the vector values. For each weight vector $\mathbf{x}$ of an individual, a move vector $\mathbf{d}$ is generated via a Gaussian distribution.

$$\mathbf{d} \sim \mathcal{N}\left(0, s^2 \cdot \mathbf{I}\right) \tag{29}$$

where s is the step size and $\mathbf{I}$ is the unit matrix.

Update the weight vector according to the shift vector $\mathbf{d}$:

$$\mathbf{x}_{\text{new}} = \mathbf{x} + \mathbf{d} \tag{30}$$

Step 5: Random wandering. After a multi-step move, a random walk is performed with a certain probability to prevent a locally optimal solution.

$$X(i,j) = \text{rand}(-r, r) \ ; \ N(i,j) = \text{fitness}(X(i,j)) \tag{31}$$

Step 6: Blast event. A blast event is triggered when an individual's fitness value does not improve after several iterations. A blast event is a process that randomly resets the position and trophic status of the bacterial population.

Step 7: Loop execution. Repeat steps 3 to 6 a certain number of times until convergence or the maximum number of iterations is reached.

The pseudo-code for ABFO-DSNN is shown in Algorithm 1. The final optimization results in $\mathbf{X}_{\text{best}}$ and f_{best}, which correspond to the optimal individual nutrient concentrations and corresponding positions for all colony processes. These values will be returned as output parameters.

Algorithm 1 ABFO-DSNN

Input: Weight parameter $\mathbf{X}$ of the DSNN model (column vector); cost function costFunction; bacterial population size N; nutrient concentration step N_c; movement step N_s; transfer probability P_{move}; maximum number of iterations $\text{step}_{\max}$.

Output: The weight parameter $\mathbf{X}_{\text{best}}$ of the optimized DSNN model; the value of the cost function $\mathbf{f}_{\text{best}}$ of the optimized DSNN model.

1: Initialisation of the colony and random distribution of nutrient concentrations in the colony.
2: Repeat until $\text{step}_{\max}$.
3: Check the position of each bacterium and spot.
4: Calculation of the upward, downward, leftward and rightward gradients for each bacterium based on the nutrient concentration of the spot.
5: Choose to move up, down, left or right or to stay, depending on the best historical position of each individual bacterium in the colony and the best historical position of the group.
6: Calculate the position of each bacterium after it has moved.
7: Update the nutrient concentration of each bacterium.
8: Check that each bacterium in the colony has not colonised or spread.
9: The decision to colonise or spread is based on the probability of transfer.
10: Calculate and update the nutrient concentrations of all colonies.
11: Calculation of the optimal solution for the colony and the corresponding optimal nutrient concentration.

5. Experimental results and analysis.

5.1. Experimental design.

To validate the classification effectiveness and performance of the ABFO-DSNN model, this work conducts comparative experiments on the Wine, Gisette, Madelon and SECOM datasets.

All four datasets are commonly used machine learning datasets to test the performance of classification algorithms. Among them, the Wine dataset is mainly used to test the multi-classification algorithm, the Gisette dataset is mainly used to test the classification ability of sparse and high-dimensional data, the Madelon dataset is mainly used to test the algorithm's ability to distinguish between noisy and non-noisy attributes, and the SECOM dataset is mainly used to test the performance of the anomaly detection algorithm. Based on the accuracy of the classification results and their RMSE, this work compares the

ABFO-DSNN with DSNN, BFO-DSNN and ABFO-DSNN models for simulation. The experimental environment is shown in Table 3.

Table 3. Experimental environment

Experimental platform	**Specific parameters**
Memory/CPU	16 G/R7 5800X
Systems	Windows 11
Frames	Tensorflow, Keras
Programming Languages	Matlab

5.2. **Optimisation performance of ABFO on DSNN.** To verify the impact of the ABFO model on the classification performance of DSNN, the DSNN, BFO-DSNN and ABFO-DSNN models were used to simulate the classification performance of the four datasets respectively. A comparison of the classification accuracies of the three models is shown in Table 4. It can be seen that the classification accuracy of the DSNN model is

Table 4. Classification accuracy of three models

Sample	Models	Accuracy		
		Minimum value	Average	Maximum value
Wine	DSNN	0.8467	0.8534	0.8609
	BFO-DSNN	0.9265	0.9301	0.9325
	ABFO-DSNN	0.9413	0.9582	0.9596
Gisette	DSNN	0.8531	0.8629	0.8653
	BFO-DSNN	0.9167	0.9202	0.9224
	ABFO-DSNN	0.9407	0.9482	0.9489
Madelon	DSNN	0.8560	0.8605	0.8692
	BFO-DSNN	0.9196	0.9224	0.9261
	ABFO-DSNN	0.9398	0.9466	0.9485
SECOM	DSNN	0.8572	0.8628	0.8667
	BFO-DSNN	0.9187	0.9205	0.9223
	ABFO-DSNN	0.9403	0.9441	0.9462

significantly improved after the BFO optimisation. Meanwhile, the DSNN model accuracy was further enhanced with ABFO. In the Wine set, the ABFO-DSNN model improved the accuracy by 3.02% compared to the BFO-DSNN model and by 12.28% compared to the DSNN. In the Gisette set, the ABFO-DSNN model improved the accuracy by 3.04% compared to the BFO-DSNN model and by 9.88% compared to the DSNN. In the Madelon set, the ABFO-DSNN model improved the accuracy by 2.62% compared to the BFO-DSNN model and by 10.00% compared to the DSNN. In the SECOM set, the ABFO-DSNN model improved the accuracy by 2.56% compared to the BFO-DSNN model and by 9.42% compared to the DSNN.

The cross-sectional comparison revealed that the ABFO algorithm had the highest level of optimisation for the DSNN model in the Wine set and the lowest in the SECOM set. RMSE comparisons of the classification accuracy of the three models are shown in Table 5. It can be seen that the RMSE values of the ABFO-DSNN model are significantly lower than those of BFO-DSNN and DSNN in terms of the RMSE of the classification accuracy of the 4-class set, mainly because it is not easy to find better DSNN model parameters within a specified number of iterations, which means that the stability of the solved DSNN model is poor. However, with the BFO algorithm, the stability of DSNN is

Table 5. Classification accuracy rate RMSE of three kinds models

Sample	Models	RMSE		
		Minimum value	Average	Maximum value
Wine	DSNN	0.9311	0.9542	0.9881
	BFO-DSNN	0.8025	0.8242	0.8434
	ABFO-DSNN	0.7311	0.7372	0.7349
Gisette	DSNN	1.0183	1.0305	1.0532
	BFO-DSNN	0.8252	0.8468	0.8683
	ABFO-DSNN	0.7615	0.7637	0.7682
Madelon	DSNN	1.1215	1.1566	1.1793
	BFO-DSNN	0.9152	0.9271	0.9339
	ABFO-DSNN	0.8233	0.8289	0.8365
SECOM	DSNN	1.1304	1.1429	1.1553
	BFO-DSNN	0.9131	0.9253	0.9365
	ABFO-DSNN	0.8247	0.8296	0.8358

significantly improved. The comparison revealed that in the Wine set, the RMSE of the three models was better relative to the other three classes of sets, which may be due to the higher dimensionality of the other three classes of sets, resulting in more significant stability oscillations.

5.3. **Classification accuracy of different algorithms.** To further validate the classification performance of the ABFO-DSNN model, the CNN, LSTM, RNN and AWWO-SCNN algorithms were simulated for four types of sample sets. the classification accuracies of the four deep neural network models are shown in Figure 6. It can be seen that among

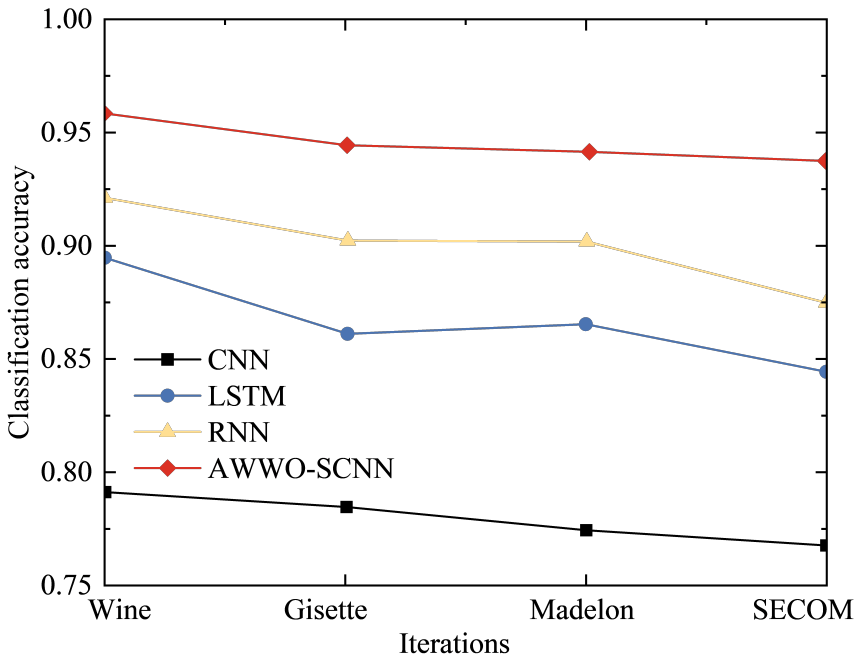

Figure 6. Classification accuracy of four models

these four deep neural network models, the ABFO-DSNN model has the highest classification accuracy with values above 0.9, the LSTM and RNN are both between [0.8, 0.9], while the CNN is basically around 0.7. This is mainly because ABFO reduces the errors that occur in the CNN-SNN conversion process in the DSNN model and maximizes the

performance of the SNN. In addition, ABFO updates the weight parameters by simulating biological evolution, which is more in line with the actual natural laws, and is therefore better able to find the appropriate weight parameters. the weight parameters of DSNN models are usually high-dimensional, while the ABFO algorithm is good at dealing with high-dimensional problems and can search for the optimal solution in the high-dimensional space.

6. **Conclusion.** This work evaluates the classification performance of DSNN from the perspective of multi-label classification. The BFO algorithm is used to optimise the core parameters of the DSNN model when solving for them. To reduce the probability of local convergence of the BFO algorithm, this work brings the idea of variation (mutation factor) from genetic algorithms into the BFO algorithm. The proposed adaptive BFO algorithm has a high convergence speed and accuracy, while reducing the probability of local convergence to a certain extent. As the ANN-SNN conversion process requires scaling the weights of the neural network, the scaling ratio of each layer of weights can be considered as an independent hyperparameter to be optimised. In order to reduce the errors arising from the conversion process and maximise the performance of the DSNN, this work uses the adaptive BFO algorithm to find the optimal scaling parameters for the DSNN, thereby improving the training efficiency and classification accuracy of the DSNN. However, organisms learn from their interactions with their environment throughout their lives, and artificial intelligence systems need to be equally capable of continuous learning if they are to act and adapt in the real world, hence the need to combine SNNs with reinforcement learning in order to improve generalisability and reduce energy consumption.

Acknowledgements. This work was supported by Natural Science Foundation of Inner Mongolia Autonomous Region of china(No.2023LHMS06024), and the Association of Fundamental Computing Education in Chinese Universities foundation project(No.2022-AFCEC-428).

REFERENCES

[1] Y. Ma, Y. Peng, and T.-Y. Wu, "Transfer learning model for false positive reduction in lymph node detection via sparse coding and deep learning," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2121-2133, 2022.

[2] F. Zhang, T.-Y. Wu, Y. Wang, R. Xiong, G. Ding, P. Mei, and L. Liu, "Application of Quantum Genetic Optimization of LVQ Neural Network in Smart City Traffic Network Prediction," *IEEE Access*, vol. 8, pp. 104555-104564, 2020.

[3] T.-Y. Wu, A. Shao, and J.-S. Pan, "CTOA: Toward a Chaotic-Based Tumbleweed Optimization Algorithm," *Mathematics*, vol. 11, no. 10, 2339, 2023.

[4] T.-Y. Wu, H. Li, and S.-C. Chu, "CPPE: An Improved Phasmatodea Population Evolution Algorithm with Chaotic Maps," *Mathematics*, vol. 11, no. 9, 1977, 2023.

[5] M. Osman, J. He, F. Mokbal, and N. Zhu, "Artificial Neural Network Model for Decreased Rank Attack Detection in RPL Based on IoT Networks," *International Journal of Network Security*, vol. 23, no. 3, pp. 497-504, 2021.

[6] N. A. Kamal, H. H. Ali, and G. Basuony, "Maximum power production operation of doubly fed induction generator wind turbine using adaptive neural network and conventional controllers," *International Journal of Computer Applications in Technology*, vol. 65, no. 2, pp. 173-186, 2021.

[7] A. L. H. P. Shaik, M. K. Manoharan, A. K. Pani, R. R. Avala, and C.-M. Chen, "Gaussian Mutation–Spider Monkey Optimization (GM-SMO) Model for Remote Sensing Scene Classification," *Remote Sensing*, vol. 14, no. 24, 6279, 2022.

[8] C.-M. Chen, S. Lv, J. Ning, and J. M.-T. Wu, "A Genetic Algorithm for the Waitable Time-Varying Multi-Depot Green Vehicle Routing Problem," *Symmetry*, vol. 15, no. 1, 124, 2023.

[9] L. Kang, R.-S. Chen, N. Xiong, Y.-C. Chen, Y.-X. Hu, and C.-M. Chen, "Selecting Hyper-Parameters of Gaussian Process Regression Based on Non-Inertial Particle Swarm Optimization in Internet of Things," *IEEE Access*, vol. 7, pp. 59504-59513, 2019.

[10] T. Zhang, and J. Huang, "Detection of chicken infected with avian influenza based on audio features and fuzzy neural network," *Transactions of the Chinese Society of Agricultural Engineering*, vol. 35, no. 2, pp. 168-174, 2019.

[11] C. Xie, and A. Kumar, "Finger vein identification using Convolutional Neural Network and supervised discrete hashing," *Pattern Recognition Letters*, vol. 119, pp. 148-156, 2019.

[12] A. Vijayan, and S. Diwakar, "A cerebellum inspired spiking neural network as a multi-model for pattern classification and robotic trajectory prediction," *Frontiers in Neuroscience*, vol. 16, 909146, 2022.

[13] C. Michaelis, A. B. Lehr, W. Oed, and C. Tetzlaff, "Brian2Loihi: An emulator for the neuromorphic chip Loihi using the spiking neural network simulator Brian," *Frontiers in Neuroscience*, vol. 16, 1015624, 2022.

[14] A. Umakantha, B. A. Purcell, and T. J. Palmeri, "Relating a Spiking Neural Network Model and the Diffusion Model of Decision-Making," *Computational brain & behavior*, vol. 5, no. 3, pp. 279-301, 2022.

[15] S. Y. Kim, and W. Lim, "Disynaptic effect of hilar cells on pattern separation in a spiking neural network of hippocampal dentate gyrus," *Cognitive Neurodynamics*, vol. 16, no. 6, pp. 1427-1447, 2022.

[16] S. Gao, S. Y. Xiang, Z. W. Song, Y. N. Han, Y. N. Zhang, and Y. Hao, "Motion detection and direction recognition in a photonic spiking neural network consisting of VCSELs-SA," *Optics Express*, vol. 30, no. 18, pp. 31701-31713, 2022.

[17] Y. Liu, L. Tian, and L. Fan, "The hybrid bacterial foraging algorithm based on many-objective optimizer," *Saudi Journal of Biological Sciences*, vol. 27, no. 12, pp. 3743-3752, 2020.

[18] G. C. Sowparnika, M. Thirumarimurugan, V. M. Sivakumar, and N. Vinoth, "Controlled infusion of intravenous cardiac drugs using global optimization," *Indian Journal of Pharmacology*, vol. 51, no. 1, pp. 61-71, 2019.

[19] D. R. W. Burrows, G. Diana, B. Pimpel, F. Moeller, M. P. Richardson, D. S. Bassett, M. P. Meyer, and R. E. Rosch, "Microscale Neuronal Activity Collectively Drives Chaotic and Inflexible Dynamics at the Macroscale in Seizures," *Journal of Neuroscience*, vol. 43, no. 18, pp. 3259-3283, 2023.

[20] A. Karekal, S. Stuart, M. Mancini, and N. C. Swann, "Elevated Gaussian-modeled beta power in the cortex characterizes aging, but not Parkinson's disease," *Journal of Neuroscience*, vol. 129, no. 5, pp. 1086-1093, 2023.

[21] S. Liu, J. J. Wang, J. T. Zhou, S. G. Hu, Q. Yu, T. P. Chen, and Y. Liu, "An Area- and Energy-Efficient Spiking Neural Network with Spike-Time-Dependent Plasticity Realized with SRAM Processing-in-Memory Macro and On-Chip Unsupervised Learning," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 17, no. 1, pp. 92-104, 2023.

[22] H. Gao, J. He, H. Wang, T. Wang, Z. Zhong, J. Yu, Y. Wang, M. Tian, and C. Shi, "High-accuracy deep ANN-to-SNN conversion using quantization-aware training framework and calcium-gated bipolar leaky integrate and fire neuron," *Front Neurosci*, vol. 17, 1141701, 2023.

[23] M. Madhiarasan, and S. N. Deepa, "Long-Term Wind Speed Forecasting using Spiking Neural Network Optimized by Improved Modified Grey Wolf Optimization Algorithm," *International Journal of Advanced Research*, vol. 4, no. 7, pp. 356-368, 2016.

[24] Z. Xu, L. Zhuang, S. Tian, M. He, S. Yang, Y. Song, and L. Ma, "Energy-driven virtual network embedding algorithm based on enhanced bacterial foraging optimization," *IEEE Access*, vol. 8, pp. 76069-76081, 2020.

[25] R. C. Ivans, S. G. Dahl, and K. D. Cantley, "A Model for R(t) Elements and R(t) -Based Spike-Timing-Dependent Plasticity with Basic Circuit Examples," *IEEE Trans Neural Netw Learn Syst*, vol. 31, no. 10, pp. 4206-4216, 2020.

[26] M. Carlu, O. Chehab, L. Dalla Porta, D. Depannemaecker, C. Herice, M. Jedynak, E. Koksal Ersoz, P. Muratore, S. Souihel, C. Capone, Y. Zerlaut, A. Destexhe, and M. di Volo, "A mean-field approach to the dynamics of networks of complex neurons, from nonlinear Integrate-and-Fire to Hodgkin-Huxley models," *Journal of neurophysiology*, vol. 123, no. 3, pp. 1042-1051, 2020.

[27] S. Elzoheiry, A. Lewen, J. Schneider, M. Both, D. Hefter, J. C. Boffi, J. O. Hollnagel, and O. Kann, "Mild metabolic stress is sufficient to disturb the formation of pyramidal cell ensembles during gamma oscillations," *Journal of Cerebral Blood Flow & Metabolism*, vol. 40, no. 12, pp. 2401-2415. 2020.

[28] D. Depannemaecker, L. E. Canton Santos, A. M. Rodrigues, C. A. Scorza, F. A. Scorza, and A. G. Almeida, "Realistic spiking neural network: Non-synaptic mechanisms improve convergence in cell assembly," *Neural Networks*, vol. 122, pp. 420-433, 2020.