Research Article

Evolutionary Dendritic Neural Model for Classification Problems

Xiaoxiao Qian ¹, **Cheng Tang**,¹ **Yuki Todo** ², **Qiuzhen Lin** ³, and **Junkai Ji** ³

¹Faculty of Engineering, University of Toyama, Toyama 930-8555, Japan

²School of Electrical and Computer Engineering, Kanazawa University, Kanazawa-shi 920-1192, Japan

³College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China

Correspondence should be addressed to Junkai Ji; jjunkai@szu.edu.cn

Received 18 May 2020; Accepted 22 July 2020; Published 19 August 2020

Academic Editor: Qingling Wang

Copyright © 2020 Xiaoxiao Qian et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this paper, an evolutionary dendritic neuron model (EDNM) is proposed to solve classification problems. It utilizes synapses and dendritic branches to implement the nonlinear computation. Distinct from the classical dendritic neuron model (CDNM) trained by the backpropagation (BP) algorithm, the proposed EDNM is trained by a metaheuristic cuckoo search (CS) algorithm instead, which has been regarded as a global searching algorithm. CS algorithm enables EDNM to avoid several disadvantages, such as slow convergence, trapping into local minimum, and being sensitive to initial values. To evaluate the performance of EDNM, we compare it with a multilayer perceptron (MLP) and CDNM on two benchmark classification problems. The experimental results demonstrate that EDNM is superior to MLP and CDNM in terms of accuracy rate, receiver operator characteristic curve (ROC), and convergence speed. In addition, the neural structure of EDNM can be replaced by a logical circuit completely, which can be implemented in hardware easily. The corresponding experimental results also verify the effectiveness of the logical circuit classifier.

1. Introduction

Classification is machine learning techniques that allocate objects in a collective form to find the classes. Many problems of science, business, and medicine can be treated as classification problems, for example, medical diagnosis, quality control, bankruptcy prediction, credit scoring, handwritten character recognition, and speech recognition [1]. Various machine learning techniques have been proposed to solve classification problems, namely, k -nearest neighbor [2], decision tree [3], artificial neural networks (ANNs) [4], rule-based classifier [5], naive Bayes classifiers [6], linear discriminant analysis [7], and support vector machine [8].

Among them, ANNs are considered as one of the comprehensive classifiers [9], which are computational models inspired by the biological nervous system, to mimic the information processing way of neurons in the human brain [10]. The first mathematical model of ANNs which apply nonlinear threshold unit was proposed by McCulloch and Pitts in 1943 [11]. In McCulloch and

Pitts's model, the neuron receives input signals from other neurons and assigns the weight that represents the connection strength between nerve cells to each input. It determined whether it is activated or remains inactive based on the results in the model [12]. The research reveals that it is difficult for the model to solve the nonlinearly separated problems because of its oversimplified structure [13].

Although the learning ability of MLP which utilizes McCulloch and Pitts's model as a fundamental calculative unit makes it a powerful tool for various applications [14], further biological studies have inferred that a single neuron could own powerful computation capacity by taking into account the synaptic nonlinearities of a dendritic tree, which is significantly different from McCulloch and Pitts's model [15, 16]. Similar issues also occur in the spiking neural networks (SNNs), in which integrate-and-fire neurons communicate information via discrete spike events based on spike-timing-dependent plasticity (STDP) rule [17]. It is proven that although SNNs are biologically more plausible, they ignore the information computation of dendritic

structures and bear little resemblance with the biological neural model [18].

Different neurons own distinct dendritic structures in vivo; even a small variation in the dendritic morphology will produce a great change in neuron functions [19, 20]. By analyzing the interaction of the excitatory and inhibitory synaptic inputs in neural cells, Koch et al. proposed a ϵ -like neural model with a dendritic structure [21, 22]. While the dendritic structure of this model remains unchanged concerning all the given tasks, it cannot realize the plasticity property of the dendritic morphology. To be specific, the ϵ -like neural model lacks an effective mechanism to determine whether the synapse is excitatory or inhibitory, as well as which branches of dendritic trees are redundant and need to be eliminated, which means that Koch's model is unrealistic when compared with biological neuron models [23].

Recently, Legenstein and Maass proposed a single neuron model with dynamic dendritic structure based on STDP and branch-strength potentiation (BSP) [24]. The neuron model is used to solve a simple feature-binding problem in a self-organized manner. However, it has been proven that the neuron model is not capable of solving the nonlinear separable problem, such as the EXOR benchmark problem [25]. In addition, Ritter et al. proposed a mathematical model named lattice neural networks (LNNs) in a morphological neuron based on lattice algebra [26]. LNNs make use of lattice operators $\inf$ and $\sup$ for the construction of the computational algorithms and replace the multiplication algebra operator of the real numbers by the addition operator. They are closely related to Lattice Computing which is regarded as the collection of computational intelligence tools [27]. Although LNNs incorporate the dendrite computation in the neural model, they do not further interpret and realize the implement of the plasticity mechanisms in the dendritic structure.

In our previous work [28, 29], we proposed a single neuron model with nonlinear interaction among synapses in the dendrites, named CDN. Experimental studies prove that CDN can effectively settle practical tasks including cancer diagnoses [30, 31], financial time series forecasting [32], and credit risk assessment [33]. Besides, an unsupervised learnable neuron model based on CDN was proposed, and it has been proved to be able to solve the two-dimensional multidirectional selectivity problem [34]. Moreover, studies prove that plasticity mechanisms of CDN can be implemented via a neuron-pruning mechanism, which consists of synaptic pruning and dendritic pruning. Neuron pruning occurs along with the training process of the model. The pruned neural model can be replaced with the logical circuits that merely contain the comparators, logical AND, OR, and NOT [35, 36].

Although CDN has been used in various applications effectively, the original BP algorithm largely limits the computation capability of CDN. BP is the gradient-based training algorithm; it requires that the neuron transfer function must be differentiable. The gradient information is highly sensitive to the initial conditions, which makes BP suffer from trapping into local minima easily [37]. In

addition, BP and its variations have several drawbacks, such as slow convergence speed and overfitting [38]. Therefore, to avoid these disadvantages caused by BP algorithm, the proposed EDNM employs a nature-inspired CS algorithm [39] as the learning algorithm in this paper, which is acknowledged as a global search algorithm. The CS algorithm combines a global random walk with a local random walk, which mimics brood parasitic behavior of cuckoo species and the Lévy flight [40] behavior of some birds and fruit flies. The powerful optimization ability enables CS to become an effective training algorithm, and EDNM can avoid trapping into local minima due to the update of the solution being independent of explicit gradient information. The performances of EDNM are evaluated and compared on two benchmark classification datasets in our experiments. In addition, we also verify the effectiveness of neural pruning and logical circuit replacement.

The rest of this paper is organized as follows: Section 2 describes the details of the proposed model. Section 3 introduces CS algorithm. Simulations related to the descriptions of two benchmark datasets, evaluation metrics, experimental setup, and performance comparison are provided and discussed in Section 4. Finally, concluding remarks are presented in Section 5.

2. Model Description

The proposed EDNM mimics the mechanism of signal interactions in the biological neural model. The signal processing of EDNM is shown as follows: First, the synaptic layer receives the input signals and processes them through one of defined connection cases. Then, the results of the synapses are transferred to the dendritic branches. The membrane layer sums the dendritic activation and transfers the results to the cell body. The structural morphology of EDNM has been presented in Figure 1.

2.1. Synaptic Layer. In the synaptic layer, each synapse connects one feature attribute to receive the input signals of training samples. A sigmoid function is adopted to describe the process; it can be expressed by

$$Y_{i,m} = \frac{1}{1 + e^{-k(w_{i,m}x_i - q_{i,m})}}, \quad (1)$$

where x_i represents the i th ($i = 1, 2, 3, \dots, I$) input signal and $Y_{i,m}$ represents the output of the i th synapse on m th ($m = 1, 2, 3, \dots, M$) dendritic branches. k is a user-defined parameter and remains constant in the calculation process. The parameters $w_{i,m}$ and $q_{i,m}$ are initialized randomly in the range $[-2, 2]$; then they are trained by the learning algorithms. Based on the values of $w_{i,m}$ and $q_{i,m}$, the threshold $\theta_{i,m}$ of the synaptic layer can be calculated as follows:

$$\theta_{i,m} = \frac{w_{i,m}}{q_{i,m}}. \quad (2)$$

In addition, according to different values of $w_{i,m}$ and $q_{i,m}$, the connection cases of the synaptic layer can be divided into four types, namely, the direct connection ($\bullet$), the inverse

FIGURE 1: Architecture of EDNM.

connection ($\Rightarrow$), the constant-0 connection ($\odot$), and the constant-1 connection ($\oplus$). The graphic symbols of the synapses in the four connection cases are provided in Figure 2.

- (i) *Type 1: Direct Connection.* Case (a): $0 < q_{i,m} < w_{i,m}$; for example, $w_{i,m} = 1.0$ and $q_{i,m} = 0.5$. As shown in Figure 3(a), direct connection means if the input x_i is greater than the threshold, the synapse will output “1.” Otherwise, it will output “0.”
- (ii) *Type 2: Inverse Connection.* Case (b): $w_{i,m} < q_{i,m} < 0$; e.g., $w_{i,m} = -1.0$ and $q_{i,m} = -0.5$. The sigmoid function of inverse connection is illustrated in Figure 3(b). Contrary to direct connection, if the input x_i is greater than the threshold, the synapse will output “0.” Otherwise, it will output “1.”
- (iii) *Type 3: Constant-1 Connection.* Case ((c_1)): $q_{i,m} < 0 < w_{i,m}$; e.g., $w_{i,m} = 1.0$ and $q_{i,m} = -0.5$. Case ((c_2)): $q_{i,m} < w_{i,m} < 0$; for example, $w_{i,m} = -1.0$ and $q_{i,m} = -1.5$. In Figures 3(c) and 3(d), it can be observed that, no matter what the input value is, the output of the synapse will remain “1.” Constant-1 connection plays a key role in the synaptic pruning.
- (iv) *Type 4: Constant-0 Connection.* Case ((d_1)): $0 < q_{i,m} < w_{i,m}$; e.g., $w_{i,m} = 1.0$ and $q_{i,m} = 1.5$. Case ((d_2)): $w_{i,m} < 0 < q_{i,m}$; for example, $w_{i,m} = -1.0$ and $q_{i,m} = 0.5$. Similarly, constant-0 connection implies that the output of the synapse remains “0” regardless of the input. It also contributes to the dendritic pruning, which will be introduced in the next section.

FIGURE 2: Four connection types in the synaptic layer.

2.2. Dendritic Layer. Dendritic structure plays an important role in neural computation. Different neurons own distinct dendritic structure; even a small variation in the dendritic morphology arouses a great change in the neural function. Thus, to realize the plasticity of the dendritic morphology, the simplest nonlinear operation named “multiplication” is adopted in DENM. Combined with four connection cases of the synaptic layer, it can implement neural pruning function to build a unit dendritic structure for each specific problem. The mathematical formula can be expressed as

$$Z_m = \prod_{i=1}^I Y_{i,m}. \quad (3)$$

2.3. Membrane Layer. The membrane layer receives the signals from each branch of dendrites and completes a sublinear summation operation. Then, it transfers the results to the cell body. Its equation is defined as follows:

$$V = \sum_{m=1}^M Z_m. \quad (4)$$

2.4. Cell Body (Soma). The output signal from the membrane layer is processed by a nonlinear sigmoid function in the cell body. It is the core part of the computation of the single neural model. The signal will be compared with the threshold of the soma; if it is larger, the neuron will fire; otherwise, it will not. The function of the cell body is expressed as follows:

$$O = \frac{1}{1 + e^{-k_{\text{soma}}(V - \theta_{\text{soma}})}}, \quad (5)$$

where k_{soma} denotes the positive constant parameters of the cell body. θ_{soma} represents the threshold of the cell body and its range is $[0, 1]$.

2.5. Neuron-Pruning Function. EDNM adopts the neuron-pruning function to realize the plasticity of the dendritic structure. Specifically, neuron-pruning function prunes unnecessary synapses and dendritic branches during the training process. Then it builds a unit structural morphology of EDNM for each specific problem. In EDNM, the pruning mechanism contains two parts, namely, synaptic pruning and dendritic pruning.

2.5.1. Synaptic Pruning. As introduced above, if one synaptic layer is in the constant-1 connection case, its output is fixed to 1 no matter what its input is. The fundamental math

FIGURE 3: Six function cases of the synaptic layer. (a) Direct connection. (b) Inverse connection. (c) Constant-1 connection (c_1, c_2). (d) Constant-0 connection (d_1, d_2).

operation of the dendritic layer is multiplication; it is known that any value multiplied by 1 is equal to itself. It implies that the output of this synaptic layer has no influence on the result of its local dendritic branch. Thus, we can ignore the synapse and the feature attribute it connects to, and this kind of synaptic layers needs to be discarded in EDNM.

2.5.2. Dendritic Pruning. Similarly, if a synaptic layer is a constant-0 connection, whatever the input is, its output will remain 0. Because of the multiplication operation and the rule that any value multiplied by 0 is equal to 0, the output of the whole dendritic branch is fixed to 0. The branch makes no contribution to the output of the soma body. Therefore, we should eliminate this kind of dendritic layers which include all the synaptic layers on them and the connecting feature attributes.

In order to further demonstrate the mechanism of neuron pruning, an example of the pruning process in EDNM is illustrated in Figure 4. It can be observed that, before pruning, the neural structure owns two dendritic layers and each dendritic branch has four synaptic layers in Figure 4(a). Since the synapse that connects to the input x_2 on Dendrite-2 is in the constant-0 connection case, according to the mechanism of dendritic pruning, Dendrite-2 and all the synaptic layers on it need to be pruned simultaneously. Thus, the pruned parts of the neural structure are drawn in dotted lines as illustrated in Figure 4(b).

Besides, because the synaptic layer that connects to the input x_2 on Dendrite-1 is in the constant-1 connection case, on the basis of the synaptic pruning, this synapse should be detected. Finally, the simplified neural structure is presented in Figure 4(c).

2.6. Logical Circuit. Through the synaptic pruning and dendritic pruning, only the direct connections and inverse connections are retained and a unique simplified neural structure is formed according to the problem. Furthermore, the simplified structure can be transformed into a logical circuit by the comparators, logical AND, OR, and NOT gates. As shown in Figure 5, in the synaptic layer, the direct connection can be implemented by the comparator and a combination of the comparator and logical NOT gate can be used to replace the inverse connection. For the dendritic layer, multiple synaptic layers on a branch can be connected by the logical AND gate. All the dendritic layers are aggregated to the membrane layer, which can be equivalent to the logical OR gate. In the cell body, a simple nonlinear mapping operation is implemented and it can be replaced by a single wire. A unique logical circuit can be obtained through these processes, and since there is no float-point calculation in the logical circuit, the classification speed can be extremely improved without sacrificing the accuracy. In the era of big data, logical circuit classifier might be a talented technology owing to its simplicity.

FIGURE 4: An example of neural model pruning process.

FIGURE 5: An example of the logical circuit transformation process.

3. CS Algorithm

CS algorithm is inspired by a special lifestyle and aggressive reproduction strategy of cuckoo species. Cuckoo never hatched eggs by themselves and put their eggs in the nest of other bird species. Let other bird species help them to hatch eggs. Some cuckoo species (e.g., *ani* and *guira*) not only put their eggs in the communal host nest but also throw hosts' eggs away to upgrade the hatching probability of their own eggs [41]. Sometimes, the hosts have the possibility to find the alien eggs and take a counterattack through throwing these alien eggs away or abandoning the nest and building a new nest. Studies have found that, in addition to simple parasitic behavior, a cuckoo called *Tapera* mimics the color and pattern of the eggs of the selected host [41]. This behavior is more conducive to increasing the number of the eggs that are successfully hatched.

CS algorithm is first proposed by Yang and Deb in 2009 [39]; for the simplicity in describing the CS, the following three basic rules are utilized:

- (i) Each cuckoo lays an egg at one time and places it in a randomly selected nest.

- (ii) The nests with the highest quality of the eggs are carried over to the next generations.
- (iii) The number of the available host nests is constant, and the egg is discovered by the host bird with a probability of $P_a \in [0, 1]$. The latter assumption can be approximated by the fraction P_a where the n nests are replaced by new ones (new random solutions).

With these three rules, the basic steps of the CS are summarized as the pseudocode shown in Algorithm 1. In the CS algorithm, a global random walk combined with a local random walk is adopted. First, the equation of the local random walk can be expressed as follows:

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (x_j^t - x_k^t), \quad (6)$$

where x_j^t and x_k^t are two distinct random solutions in the current population, s represents the step size, and α denotes its scaling factor. $H(\cdot)$ represents a unit step function. p_a is a switching parameter that controls the balance between a local random walk and a global random walk, ϵ is a random value from a uniform distribution. The symbol $\otimes$ represents the entry-wise multiplications operation. Then, the global random walk that applies Lévy flights can be described as follows:

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda), \quad (7)$$

where α denotes the scaling factor of step size and the function $L(s, \lambda)$ can be calculated by

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0), \quad (8)$$

where λ is the Lévy exponent, function Γ will be a constant for a given λ , and α represents the step size scaling factor. It is widely regarded that the Lévy flights can maximize the efficiency of the resource searches, and it has been observed from the foraging patterns of albatrosses, fruit flies, and spider monkeys [42–44]. In addition, empirical evidence has verified that CS is superior to PSO and genetic algorithms [39]. Therefore, the CS algorithm is employed as the training algorithm in our experiments (Algorithm 1).


```

(1) Objective function  $f_{(x)}$ ,  $x = (x_1, \dots, x_d)^T$ ;
(2) Initial a population of  $n$  host nests  $x_i (i = 1, 2, \dots, n)$ ;
(3) while Stop Criterion do
(4)   Lay an egg ( $x_i$ ) by cuckoo and put it into a nest randomly by adopting Lévy flights;
(5)   Evaluate the quality of the nest ( $F_j$ );
(6)   Randomly choose one of  $n$  host nests ( $(F_i)$ );
(7)   if  $F_i > F_j$  then
(8)     Replace  $x_j$  by the new solution  $x_i$ ;
(9)   end if
(10)  Abandon a part of the worse nests with the probability  $P_a$ ;
(11)  Apply Lévy flights to generate new nests;
(12)  Evaluate the quality of the new nests;
(13)  Rank the nests to find the current best one;
(14)  Update: replace  $t$  by  $t + 1$ ;
(15) end while

```

ALGORITHM 1: Cuckoo search algorithm.

TABLE 1: Dataset description.

Dataset	No. of samples	No. of attributes	No. of classes (samples divided)	Attributes characteristics
GID	214	9	2 (163 : 51)	Numerical
CVRD	232	16	2 (124 : 108)	Categorical

4. Simulation

4.1. Dataset Description. In this section, to compare the classification performances of the EDNM, CDNM, and MLP, we conduct the simulations on two benchmark datasets, namely, the Glass Identification Dataset (GID) and Congressional Voting Records Dataset (CVRD), which are chosen from the UCI machine learning repository. The details of these datasets are shown in Table 1.

4.1.1. GID. GID is obtained by measuring the chemical constitution of glass, fabricated by two different processes [45]. The dataset contains 163 samples of window glass and 51 samples of nonwindow glass. Each record has 9 attributes, which include its refractive index and the proportion of its eight chemical components (Na, Mg, Al, Si, K, Ca, Ba, and Fe). These analytical results are recorded as 9 numerical continuous values.

4.1.2. CVRD. CVRD records the voting results of the 98th Congress. It contains 435 samples that record the data of votes for each of the U.S. House of Representatives Congressmen on the 16 key votes (attributes) identified by the CQA. Its classification task is to find the correct political party affiliation of each congressman [46]. Since some attributes of CVRD include missing attribute values, the attribute with missing values needs to be deleted. Finally, 232 complete samples are left, which include 124 “Democrat” samples and 108 “Republican” samples. CVRD is recorded as categorical attributes; thus, 16 categorical attributes, “Yes” and “No,” are converted to numerical “1” and “0”; two categorical classes “Republican” and “Democrat” are changed to numerical “1” and “0,” respectively [47].

TABLE 2: Confusion matrix.

Total population		True condition	
		P	N
Predicted condition	Y	True positive (TP)	False positive (FP)
	N	False negative (FN)	True negative (TN)

TABLE 3: Number of samples in the training and testing parts.

Dataset	No. in training data	No. in testing data	Total no.
GID	107	107	214
CVRD	116	116	232

4.2. Evaluation Metrics. In our experiments, to measure the performance of each model, we adopt four performance evaluation criteria, namely, accuracy rate, receiver operator characteristic curve (ROC), convergence speed, and non-parametric statistical test.

(a) **Accuracy Rate.** The most important evaluation metric is the accuracy rate, which can be expressed as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

where TP, TN, FP, and FN indicate true positive, true negative, false positive, and false negative, respectively. To understand the equation better, the confusion matrix constructed by TP, TN, FP, and FN is shown in Table 2. Although accuracy is the simplest, most intuitive, and

TABLE 4: Structures of EDNM, CDNМ, and MLP for GID and CVRD.

Dataset	Model	No. of inputs	No. of branches	No. of outputs	No. of adjusted weights
GID	EDNM	9	12	1	216
	CDNM	9	12	1	216
	MLP	9	20	1	221
CVRD	EDNM	16	18	1	576
	CDNM	16	18	1	576
	MLP	16	32	1	557

TABLE 5: Simulation parameter levels for two benchmark datasets in the EDNM.

Dataset	k	M (branch)	θ_{soma}
GID	2, 5, 8, 10	9, 10, 11, 12	0.3, 0.5, 0.7, 0.9
CVRD	2, 5, 8, 10	16, 18, 20, 22	0.3, 0.5, 0.7, 0.9

TABLE 6: $L_{16}(4^3)$ orthogonal array and factor assignment of GID.

No. of experiments	k	M	θ_{soma}	Testing accuracy means $\pm$ std (%)
1	2	9	0.3	92.59 $\pm$ 2.39
2	2	10	0.5	91.48 $\pm$ 2.25
3	2	11	0.7	91.90 $\pm$ 2.58
4	2	12	0.9	92.87 $\pm$ 3.09
5	5	9	0.5	90.97 $\pm$ 3.01
6	5	10	0.3	92.15 $\pm$ 2.17
7	5	11	0.9	92.62 $\pm$ 2.46
8	5	12	0.7	92.02 $\pm$ 1.87
9	8	9	0.7	91.90 $\pm$ 2.20
10	8	10	0.9	90.90 $\pm$ 2.62
11	8	11	0.3	91.43 $\pm$ 2.46
12	8	12	0.5	93.27 $\pm$ 2.85
13	10	9	0.9	92.87 $\pm$ 1.88
14	10	10	0.7	91.46 $\pm$ 4.11
15	10	11	0.5	92.34 $\pm$ 2.59
16	10	12	0.3	91.12 $\pm$ 2.49

commonly used performance comparison method, it is not enough for a complete performance evaluation [48].

- (b) *ROC*. ROC curve is a widely used method to display complete information on the set of all possible combinations of sensitivity and specificity and is also useful as a graphical characterization of the magnitude of separation between the case and control marker distributions [49]. The area under the ROC curve, known as the AUC, is more intuitive and has been considered as the standard method to assess the accuracy of predictive distribution models. When continuous probability derived scores are converted to a binary presence-absence variable, the supposed subjectivity in the threshold selection process can be avoided by summarizing the overall model performances with all possible thresholds [50]. If case measurements and control measurements have no overlap, then the AUC takes the value “1” for any false positive rate greater than 0; the marker is perfect in discriminating between cases and controls. Alternatively, if the case and control distributions are

TABLE 7: $L_{16}(4^3)$ orthogonal array and factor assignment of CVRD.

No. of experiments	k	M	θ_{soma}	Testing accuracy means $\pm$ std (%)
1	2	16	0.3	96.18 $\pm$ 1.50
2	2	18	0.5	96.29 $\pm$ 1.59
3	2	20	0.7	94.28 $\pm$ 8.16
4	2	22	0.9	95.14 $\pm$ 2.44
5	5	16	0.5	94.40 $\pm$ 7.93
6	5	18	0.3	96.41 $\pm$ 1.53
7	5	20	0.9	93.59 $\pm$ 7.64
8	5	22	0.7	94.54 $\pm$ 8.64
9	8	16	0.7	95.57 $\pm$ 2.55
10	8	18	0.9	95.09 $\pm$ 7.78
11	8	20	0.3	94.20 $\pm$ 8.15
12	8	22	0.5	95.95 $\pm$ 1.74
13	10	16	0.9	95.11 $\pm$ 4.19
14	10	18	0.7	93.51 $\pm$ 5.35
15	10	20	0.5	96.12 $\pm$ 2.35
16	10	22	0.3	94.91 $\pm$ 3.11

identical, the marker is in a random classification case. Correspondingly, the equation of AUC can be described as follows:

$$\text{AUC} = \frac{1}{2} \left(\frac{\text{TP}}{\text{TP} + \text{FN}} + \frac{\text{TN}}{\text{TN} + \text{FP}} \right) \times 100. \quad (10)$$

- (c) *Convergence Speed*. High convergence speed indicates the high efficiency of the model. Thus, it is necessary to use mean squared error at each iteration to compare the convergence speeds of different classifiers [51, 52]. The mean squared error is calculated and expressed as follows:

$$\text{Fitness} = \frac{1}{2J} \sum_{j=1}^J (O_j - T_j)^2, \quad (11)$$

where O_j and T_j indicate the actual output and the predicted output, respectively. J is the sample number of the training dataset.

- (d) *Nonparametric Statistical Test*. Since the nonparametric statistical test is not limited by the overall distribution and its assumption is relatively fewer, it is more robust and has wider applicability than a parametric statistical test. The nonparametric analysis test based on the assumption of the normal

FIGURE 6: The convergence speeds of EDNM, CDNM, and MLP for GID and CVRD.

distribution is more sensitive and reliable than t -test which ignores the absolute magnitudes of the differences [53]. In our experiments, Wilcoxon's rank-sum test [54] is adopted to complete the nonparametric statistical test.

4.3. Simulation Setup. In our experiment, each dataset is split into a training set and a testing set. Each set contains 50% of the samples [55], as shown in Table 3. Before training, the dataset will be normalized to the range of $[0, 1]$. The normalized rule follows the maximum and minimum normalization method, which can be expressed as follows:

$$X_{\text{normalized}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}. \quad (12)$$

In order to maintain the fairness of the comparison, the number of the parameters in each model should be set to the same or approximately equal as possible. The modal structure of MLP is different from CDNM and EDNM; the numbers of the weights and thresholds in MLP can be calculated as follows:

$$N_{\text{MLP}} = (I + 1)W + (W + 1), \quad (13)$$

where I and W represent the numbers of neurons in the input layer and hidden layer, respectively. In the neural structure of EDNM and CDNM, since each synapse owns two parameters $w_{i,m}$ and $q_{i,m}$, when the number of dendritic layers M is determined, the total parameter number in EDNM and CDNM structure is expressed by the following equation:

$$N_{\text{EDNM}} = 2 \times I \times M. \quad (14)$$

In our experiment, when a benchmark dataset is chosen, the value of I will be determined. Based on equations (13) and (14), setting suitable values of W and M will make N_{MLP} and N_{EDNM} approximately equal to each other. Table 4

summarizes the model structures of MLP, CDNM, and EDNM on the two benchmark datasets. It is easy to observe that all the three methods have nearly the same parameter numbers for both datasets. In addition, the transfer functions of MLP in the hidden layer and output layer are both set to "Log-sigmoid." The learning rate of CDNM and MLP is 0.01. The population size of EDNM is set to 50. The iteration times of three methods are set to be 1000; each method runs 30 times in our experiments independently.

4.4. Optimal Parameter Setting. In EDNM, there are three parameters, namely, k , θ_{soma} , and M , which need to be defined by users. k is a constant value in the sigmoid function of the synaptic layer, θ_{soma} denotes the threshold of the soma, and M represents the number of the dendritic branches. In order to find an optimal combination of these parameters, the Taguchi method is adopted in our experiment, which can reduce the number of experimental running times and ensure the dependability of the experiment [56, 57]. According to the Taguchi method, only 16 experiments out of 64 are run; and 16 experiments are enough to find the optimal parameter setting quickly and efficiently. Table 5 shows four levels of interest for the two benchmark datasets. The $L_{16}(4^3)$ orthogonal arrays are shown in Tables 6 and 7, respectively. It can be observed that, in Table 6, the 12th parameter setting ($k = 8$, $M = 12$, and $\theta_{\text{soma}} = 0.5$) holds the highest testing accuracy; in Table 7, the highest testing accuracy is obtained by the 6th parameter setting ($k = 5$, $M = 18$, and $\theta_{\text{soma}} = 0.3$). Through the above experiments, the optimal parameter settings of the two benchmark datasets can be determined.

4.5. Performance Comparison. In order to verify the classification performance of EDNM, we compare it with MLP and the original CDNM on two benchmark datasets. Table 8 presents the experimental results. It is easy to observe that EDNM obtains higher accuracy than MLP and CDNM on

FIGURE 7: The AUC curves of EDNM, CDNM, and MLP for GID and CVRD.

FIGURE 8: Evolution of neuronal morphology on GID problem.

FIGURE 9: Evolution of neuronal morphology on CVRD problem.

TABLE 8: Results of accuracy, p value, and AUC for EDNM, CDNM, and MLP.

Dataset	Model	Accuracy (%)	p value (Acc)	AUC	p value (AUC)
GID	EDNM	93.27 ± 2.85	—	0.9868	—
	CDNM	86.95 ± 7.28	$8.07E-04$	0.7782	$2.42E-02$
	MLP	84.21 ± 6.58	$9.60E-06$	0.7444	$5.37E-06$
CVRD	EDNM	96.41 ± 1.53	—	0.9725	—
	CDNM	67.16 ± 20.46	$7.38E-06$	0.6147	$5.91E-06$
	MLP	84.63 ± 8.89	$8.94E-07$	0.8927	$1.85E-06$

both problems. To detect the significant differences between EDNM and the other models, Wilcoxon's rank-sum test is utilized in our experiment. Its significance level is set to 0.05. If the p value is less than 0.05, the null hypothesis that there are no significant differences between the two comparison objects can be rejected. The statistical results are shown in Table 8. From Table 8, it is implied that EDNM performs significantly better than both MLP and CDNM on the two benchmark problems.

In addition, for the comprehensive evaluation of model performance, the convergence curves of three models on two

benchmark problems are illustrated in Figure 6. As shown in Figure 6, EDNM has a higher convergence speed than MLP and CDNM, obviously. In Table 8, the statistical results demonstrate that the AUC value of EDNM is significantly larger than those of the other models on both problems. The corresponding ROC curves are compared and presented in Figure 7.

Based on the above experimental results, it can be concluded that EDNM is capable of providing more powerful classification performances to solve GID and CVRD problems compared to MLP and CDNM. Higher convergence speed indicates that EDNM is a more efficient classifier, which will save computation time in the practical applications.

4.6. Neuron-Pruning Analysis. The neuron-pruning function of EDNM has been introduced in Section 3. During the training process, it prunes the unnecessary synapses and dendritic branches and then produces a specific dendritic structure for each problem. The high plasticity of EDNM's structure brings about the following benefits: First, neuron pruning can realize feature selection for EDNM; only the

FIGURE 10: Logical circuit classifiers of GID and CVRD problems.

TABLE 9: Simplification results of neuron pruning on GID and CVRD problems.

Dataset	No. of features		No. of branches		No. of synapses	
	Before	After	Before	After	Before	After
GID	9	3	12	2	108	4
CVRD	16	1	18	1	288	1

useful feature attributes which can contribute to the final results are retained in the structure. Second, it simplifies the neural structure, which will reduce computational cost and increase computational speeding (Figure 8).

Firstly, we present the evolution of the structural morphology of the GID problem in Figure 3. It can be observed that there are 9 feature attributes and 12 dendritic branches in the structure before learning; the connection cases of all the 108 synapses are randomly set in Figure 3(a). After training by CS algorithm, the structural morphology of EDNM is presented in Figure 3(b). According to the rules of dendritic pruning, 10 branches of dendrites are deleted and only “Dendrite – 5” and “Dendrite – 8” are left. Figure 3(c) illustrates the simplified structure of EDNM after dendritic pruning. Then, based on the rule of synaptic pruning, 14 unnecessary synaptic layers are ruled out and only 4 synapses are retained. Finally, the mature structural morphology of EDNM on the GID problem is provided in Figure 3(d). Similarly, Figure 9 illustrates the evolution of the structural morphology of the CVRD problem. The pruning results of the two benchmark problems are summarized in Table 9. It is easy to conclude that the pruned neural structures are much more simplified than the original ones; the neuron-pruning function can significantly simplify the structural morphology of EDNM.

4.7. Logical Circuit Analysis. As mentioned above, the simplified structures of EDNM can be completely substituted by the logical circuits. In this section, we attempt to verify the effectiveness of the logical circuit classifiers. According to the final neural structures in Figures 3 and 9,

TABLE 10: Accuracy comparison of EDNM and logical circuits.

Dataset	Accuracy (%)	
	EDNM (%)	Logical circuit (%)
GID	93.27	92.52
CVRD	96.41	96.55

the logical circuit classifiers of two benchmark problem are presented in Figure 10. As shown in Figure 10, the logical circuit classifiers consist of the comparators, logical AND, OR, and NOT gates, where the comparators are used for comparison with the input signals with their thresholds θ . If the inputs exceed the thresholds, the outputs are 1 and 0 otherwise. It is noteworthy that since the final neural structure of CVRD only has one synaptic layer left, there are no logical AND, OR, and NOT gates in the corresponding logical circuit classifier, except a comparator.

Besides, we compare the classification performances of the logical circuit classifiers and the normal EDNM in Table 10. As illustrated in the table, the logical circuits do not sacrifice the accuracies on both benchmark problems. In addition, once the logical circuit classifiers are implemented on hardware, the classification speed will be much higher than that in all the other classifiers in the literature. According to the above characteristics, the logical circuit classifier is considered as a satisfactory and efficient classifier for real-world classification tasks.

5. Conclusion

In this study, an EDNM is proposed to solve the classification problems. It consists of four layers, namely, the synaptic layer, the dendritic layer, the membrane layer, and the soma. The unique structure makes EDNM implement the neural pruning mechanism, which can rule out the unnecessary synapse and dendritic branches. Compared with the original BP algorithm of CDNM, CS algorithm has higher convergence speed and great classification accuracy on two benchmark problems, where the statistical results demonstrate that EDNM performs significantly better than

MLP and CDNM. Besides, we also present the logical circuit classifiers produced by EDNM and verify their accuracy rate. The experimental results show that the logical circuits maintain satisfying classification performances. It is noted that, to the best of our knowledge, when the logical circuit classifiers run on hardware, the classification speed will be higher than that in all the other classifiers in the literature. In our future research, we will attempt to adopt the multi-objective optimization algorithms to train the structure and weights of EDNM, simultaneously, which may be able to produce a more simplified and high-accuracy logical circuit for each classification problem.

Data Availability

The benchmark classification datasets could be downloaded freely at <https://archive.ics.uci.edu/ml/index.php>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Guangdong Basic and Applied Basic Research Fund Project (No. 2019A151111139) and JSPS KAKENHI (Grant no. 19K12136).

References

- [1] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 30, no. 4, pp. 451–462, 2000.
- [2] S. D. Bay, "Combining nearest neighbor classifiers through multiple feature subsets," in *Proceedings of the 15th International Conference on Machine Learning ICML*, Citeseer, Madison, WI, USA, July 1998.
- [3] J. R. Quinlan, *C4. 5: Programs for Machine Learning*, Elsevier, Amsterdam, Netherlands, 2014.
- [4] R. J. Schalkoff, *Artificial Neural Networks*, McGraw-Hill, New York, NY, USA, 1997.
- [5] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, no. 4, pp. 261–283, 1989.
- [6] I. Rish, "An empirical study of the Naive Bayes classifier," in *Proceedings of the IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, IBM, Seattle, WA, USA, pp. 41–46, August 2001.
- [7] M. Li and B. Yuan, "2D-LDA: a statistical linear discriminant analysis for image matrix," *Pattern Recognition Letters*, vol. 26, no. 5, pp. 527–532, 2005.
- [8] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [9] G. Ou and Y. L. Murphey, "Multi-class pattern classification using neural networks," *Pattern Recognition*, vol. 40, no. 1, pp. 4–18, 2007.
- [10] J. Khan, J. S. Wei, M. Ringnér et al., "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Medicine*, vol. 7, no. 6, pp. 673–679, 2001.
- [11] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.
- [12] A. Krogh, "What are artificial neural networks?" *Nature Biotechnology*, vol. 26, no. 2, pp. 195–197, 2008.
- [13] R. P. Costa and P. J. Sjöström, "One cell to rule them all, and in the dendrites bind them," *Frontiers in Synaptic Neuroscience*, vol. 3, no. 5, 2011.
- [14] A. Prieto, B. Prieto, E. M. Ortigosa et al., "Neural networks: an overview of early research, current frameworks and new challenges," *Neurocomputing*, vol. 214, pp. 242–268, 2016.
- [15] A. J. Koleske, "Molecular mechanisms of dendrite stability," *Nature Reviews Neuroscience*, vol. 14, no. 8, pp. 536–550, 2013.
- [16] K. L. Thompson-Peer, L. DeVault, T. Li, L. Y. Jan, and Y. N. Jan, "In vivo dendrite regeneration after injury is different from dendrite development," *Genes & Development*, vol. 30, no. 15, pp. 1776–1789, 2016.
- [17] N. G. Pavlidis, O. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis, "Spiking neural network training using evolutionary algorithms," vol. 4, pp. 2190–2194, in *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks*, vol. 4, IEEE, Montreal, Canada, July 2005.
- [18] A. Tavanaei and A. Maida, "BP-STDP: approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330, pp. 39–47, 2019.
- [19] E. Salinas and L. Abbott, "A model of multiplicative neural responses in parietal cortex," *Proceedings of the National Academy of Sciences*, vol. 93, no. 21, pp. 11 956–11 961, 1996.
- [20] F. Gabbiani, H. G. Krapp, C. Koch, and G. Laurent, "Multiplicative computation in a visual neuron sensitive to looming," *Nature*, vol. 420, no. 6913, pp. 320–324, 2002.
- [21] C. Koch, T. Poggio, and V. Torre, "Retinal ganglion cells: a functional interpretation of dendritic morphology," *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, vol. 298, no. 1090, pp. 227–263, 1982.
- [22] C. Koch, T. Poggio, and V. Torre, "Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing," *Proceedings of the National Academy of Sciences*, vol. 80, no. 9, pp. 2799–2802, 1983.
- [23] A. Destexhe and E. Marder, "Plasticity in single neuron and circuit computations," *Nature*, vol. 431, no. 7010, p. 789, 2004.
- [24] R. Legenstein and W. Maass, "Branch-specific plasticity enables self-organization of nonlinear computation in single neurons," *Journal of Neuroscience*, vol. 31, no. 30, pp. 10 787–10 802, 2011.
- [25] J. Ji, S. Song, Y. Tang, S. Gao, Z. Tang, and Y. Todo, "Approximate logic neuron model trained by states of matter search algorithm," *Knowledge-Based Systems*, vol. 163, pp. 120–130, 2019.
- [26] G. X. Ritter and G. Urcid, "Learning in lattice neural networks that employ dendritic computing," in *Computational Intelligence Based on Lattice Theory*, pp. 25–44, Springer, Berlin, Germany, 2007.
- [27] H. Sossa and E. Guevara, "Efficient training for dendrite morphological neural networks," *Neurocomputing*, vol. 131, pp. 132–142, 2014.
- [28] Z. Tang, H. Tamura, M. Kuratu, O. Ishizuka, and K. Tanno, "A model of the neuron based on dendrite mechanisms," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 84, no. 8, pp. 11–24, 2001.
- [29] Z. Tang, H. Tamura, O. Ishizuka, and K. Tanno, "A neuron model with interaction among synapses," *IEEE Transactions on Electronics, Information and Systems*, vol. 120, no. 7, pp. 1012–1019, 2000.
- [30] T. Jiang, S. Gao, D. Wang, J. Ji, Y. Todo, and Z. Tang, "A neuron model with synaptic nonlinearities in a dendritic tree

- for liver disorders,” *IEEE Transactions on Electrical and Electronic Engineering*, vol. 12, no. 1, pp. 105–115, 2017.
- [31] Z. Sha, L. Hu, Y. Todo, J. Ji, S. Gao, and Z. Tang, “A breast cancer classifier using a neuron model with dendritic non-linearity,” *IEICE Transactions on Information and Systems*, vol. E98.D, no. 7, pp. 1365–1376, 2015.
 - [32] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, and Z. Tang, “Financial time series prediction using a dendritic neuron model,” *Knowledge-Based Systems*, vol. 105, pp. 214–224, 2016.
 - [33] Y. Tang, J. Ji, S. Gao, H. Dai, Y. Yu, and Y. Todo, “A pruning neural network model in credit classification analysis,” *Computational Intelligence and Neuroscience*, vol. 2018, Article ID 9390410, 22 pages, 2018.
 - [34] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang, “Unsupervised learnable neuron model with nonlinear interaction on dendrites,” *Neural Networks*, vol. 60, pp. 96–103, 2014.
 - [35] Y. Todo, Z. Tang, H. Todo, J. Ji, and K. Yamashita, “Neurons with multiplicative interactions of nonlinear synapses,” *International Journal of Neural Systems*, vol. 29, no. 8, Article ID 1950012, 2019.
 - [36] J. Ji, S. Gao, J. Cheng, Z. Tang, and Y. Todo, “An approximate logic neuron model with a dendritic structure,” *Neurocomputing*, vol. 173, pp. 1775–1783, 2016.
 - [37] M. Bianchini and M. Gori, “Optimal learning in artificial neural networks: a review of theoretical results,” *Neurocomputing*, vol. 13, no. 2–4, pp. 313–346, 1996.
 - [38] X. G. Wang, Z. Tang, H. Tamura, M. Ishii, and W. D. Sun, “An improved backpropagation algorithm to avoid the local minima problem,” *Neurocomputing*, vol. 56, pp. 455–460, 2004.
 - [39] X.-S. Yang and S. Deb, “Cuckoo search via lévy flights,” in *Proceedings of the World Congress on Nature & Biologically Inspired Computing NaBIC 2009*, pp. 210–214, IEEE, Coimbatore, India, December 2009.
 - [40] I. Pavlyukevich, “Lévy flights, non-local search and simulated annealing,” *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.
 - [41] M. D. Sorenson, *The Cuckoos*, Oxford University Press, Oxford, UK, 2005.
 - [42] C. T. Brown, L. S. Liebovitch, and R. Glendon, “Lévy flights in dove Ju/hoansi foraging patterns,” *Human Ecology*, vol. 35, no. 1, pp. 129–138, 2007.
 - [43] A. M. Reynolds and M. A. Frye, “Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search,” *PLoS One*, vol. 2, no. 4, Article ID e354, 2007.
 - [44] P. Barthelemy, J. Bertolotti, and D. S. Wiersma, “A lévy flight for light,” *Nature*, vol. 453, no. 7194, pp. 495–498, 2008.
 - [45] H. Brunzell and J. Eriksson, “Feature reduction for classification of multidimensional data,” *Pattern Recognition*, vol. 33, no. 10, pp. 1741–1748, 2000.
 - [46] R. Setiono and H. Liu, “Neural-network feature selector,” *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 654–662, 1997.
 - [47] C.-M. Wang and Y.-F. Huang, “Evolutionary-based feature selection approaches with new criteria for data mining: a case study of credit approval data,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 5900–5908, 2009.
 - [48] W. Zhu, N. Zeng, and N. Wang, “Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations,” in *Proceedings of the NESUG: Health Care and Life Sciences*, vol. 19, p. 67, Baltimore, MA, USA, November 2010.
 - [49] P. J. Heagerty and Y. Zheng, “Survival model predictive accuracy and roc curves,” *Biometrics*, vol. 61, no. 1, pp. 92–105, 2005.
 - [50] J. M. Lobo, A. Jiménez-Valverde, and R. Real, “AUC: a misleading measure of the performance of predictive distribution models,” *Global Ecology and Biogeography*, vol. 17, no. 2, pp. 145–151, 2008.
 - [51] X.-H. Zhou and J. Harezlak, “Comparison of bandwidth selection methods for kernel smoothing of roc curves,” *Statistics in Medicine*, vol. 21, no. 14, pp. 2045–2055, 2002.
 - [52] K. Horsch, M. L. Giger, L. A. Venta, and C. J. Vyborny, “Computerized diagnosis of breast lesions on ultrasound,” *Medical Physics*, vol. 29, no. 2, pp. 157–164, 2002.
 - [53] J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.
 - [54] F. Wilcoxon, “Individual comparisons by ranking methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
 - [55] A. Khashman, “Credit risk evaluation using neural networks: emotional versus conventional models,” *Applied Soft Computing*, vol. 11, no. 8, pp. 5477–5484, 2011.
 - [56] R. Jugulum and S. Taguchi, *Computer-Based Robust Engineering: Essentials for DFSS*, ASQ Quality Press, Milwaukee, WI, USA, 2004.
 - [57] J. F. C. Khaw, B. S. Lim, and L. E. N. Lim, “Optimal design of neural networks using the Taguchi method,” *Neurocomputing*, vol. 7, no. 3, pp. 225–245, 1995.