electronics

Article

Explaining a Logic Dendritic Neuron Model by Using the Morphology of Decision Trees

Xingqian Chen [1], Honghui Fan [1], Wenhe Chen [1,2], Yaoxin Zhang [1], Dingkun Zhu [1] and Shuangbao Song [3,*]

[1] School of Computer Engineering, Jiangsu University of Technology, Changzhou 213001, China; xingzai@jsut.edu.cn (X.C.); fanhonghui@jsut.edu.cn (H.F.); chenwh@jsut.edu.cn (W.C.); zyx_ysh@jsut.edu.cn (Y.Z.); zhudingkun@jsut.edu.cn (D.Z.)
[2] Shanghai Huace Navigation Technology Co., Ltd., Shanghai 201700, China
[3] School of Computer Science and Artificial Intelligence, Changzhou University, Changzhou 213164, China
* Correspondence: leadingsong@cczu.edu.cn

Abstract: The development of explainable machine learning methods is attracting increasing attention. Dendritic neuron models have emerged as powerful machine learning methods in recent years. However, providing explainability to a dendritic neuron model has not been explored. In this study, we propose a logic dendritic neuron model (LDNM) and discuss its characteristics. Then, we use a tree-based model called the morphology of decision trees (MDT) to approximate LDNM to gain its explainability. Specifically, a trained LDNM is simplified by a proprietary structure pruning mechanism. Then, the pruned LDNM is further transformed into an MDT, which is easy to understand, to gain explainability. Finally, six benchmark classification problems are used to verify the effectiveness of the structure pruning and MDT transformation. The experimental results show that MDT can provide competitive classification accuracy compared with LDNM, and the concise structure of MDT can provide insight into how the classification results are concluded by LDNM. This paper provides a global surrogate explanation approach for LDNM.

Keywords: explainable artificial intelligence; dendritic neuron model; decision tree; global explanation approach; explainable machine learning

Citation: Chen, X.; Fan, H.; Chen, W.; Zhang, Y.; Zhu, D.; Song, S. Explaining a Logic Dendritic Neuron Model by Using the Morphology of Decision Trees. *Electronics* **2024**, *13*, 3911. https://doi.org/10.3390/electronics13193911

Academic Editors: Alberto Fernandez Hilario and Aryya Gangopadhyay

Received: 2 September 2024
Revised: 20 September 2024
Accepted: 30 September 2024
Published: 3 October 2024

1. Introduction

With the rapid development of artificial intelligence (AI) in recent years, machine learning (ML), a subset of AI, has been widely used in many fields of society [1–3]. However, some scientists and regulators have a cautious attitude toward the ML technique when it is applied in sensitive domains [4–7], such as law, finance, and health care [8], because numerous ML models can provide accurate predictions but lose explainability; thus, the inner workings and predictions of these ML models are not understandable for humans [9]. Hence, research concerning the explainability of ML has attracted great attention from researchers in recent years [10].

The ML model, whose intrinsic architecture can provide human-comprehensible explanations, is considered an interpretable (or transparent) model [11]. The typical interpretable models include linear models, decision trees, and k-nearest neighbors (KNN). However, numerous ML models, such as random forest, support vector machine (SVM), and multilayer neural networks, are considered black box (or opaque) models because they are not "interpretable by nature" [12]. Numerous methods have been proposed to provide explainability to black box models in the literature. These methods can be classified into the following two categories [6,12]: global explanation approaches and local explanation approaches. Global explanation approaches aim to provide insight into the whole logic of a black box model. A representative global explanation approach is partial dependence plots [13]. Local explanation approaches, such as Anchors [14], attempt to explain how the prediction is made by a black box model based on a specific instance and its neighbors [15].

Moreover, these two categories of explanation approaches can be further subdivided into two classes according to whether surrogate models are used. Thus, a surrogate model, which is much easier to interpret, approximates the black box model to provide explainability. Currently, inTrees [16] is considered a popular global surrogate explanation approach. LIME [17] and SHAP [18,19] are considered the most popular approaches among local surrogate explanation approaches.

Decision trees belong to a category of interpretable models because they construct tree-like structures that can be directly explained to gain explainability. Several explanation approaches employing decision trees as the core techniques have been proposed in recent years. For example, Zilke et al. proposed an approach called DeepRED to extract rules from deep neural networks [20]. The rules created by C4.5 decision trees for hidden layers are used to explain the whole network. Wu et al. trained a neural network model with novel tree regularization [21]. Decision trees are used to approximate the neural network model for interpretability. Deng proposed an explanation approach called inTrees to gain the explainability of tree ensembles [16]. This approach can extract key rules from a tree ensemble and calculate frequent variable interactions. Lundberg et al. proposed an explanation approach called TreeExplainer for explaining tree-based ML models [22]. The novelty of this approach is interpreting global model structure by combining numerous local explanations. These studies indicate that tree-based methods can serve as promising alternatives for solving the black box model explanation problem.

A neuron is the fundamental unit of the brain and consists of the following three components: dendrites, a cell body, and an axon. McCulloch and Pitts were the pioneers who first proposed an artificial neuron model in 1943 [23]. Subsequently, Rosenblatt improved the McCulloch–Pitts neuron and invented the perceptron [24]. These models laid a foundation for research involving artificial neural networks but are considered simplistic because they can solve linearly separable problems but are unable to solve nonlinear separable problems, such as the XOR problem [25]. Subsequently, scientists confirmed the importance of dendritic branches in neuron computational power [25,26]. Consequently, numerous neuron models involving the nonlinear mechanisms of dendrites have been proposed in the literature [27–31]. Moreover, synaptic pruning and dendritic pruning are considered two important mechanisms for learning [32,33]. Incorporating these relevant mechanisms into artificial neuron models is considered a promising way to improve their performance.

In recent years, researchers have paid continuous attention to developing and improving artificial neuron models with dendrite morphology. For example, Sossa et al. proposed a dendrite morphological neural network with an efficient training algorithm [34]. Gómez-Flores et al. incorporated smooth maximum and minimum functions into a dendrite morphological neuron model to generate smooth decision boundaries [35]. Luo et al. proposed a decision tree-based method to properly initialize the synaptic weights of a dendritic neuron model [36]. Bicknell et al. developed a pyramidal neuron model and a synaptic learning rule to investigate the nonlinear computations performed in dendrites [37]. In our previous studies, we proposed a series of dendritic neuron models to address several practical problems [38–42]. These studies indicate that a single neuron with dendrite morphology can perform network-level computations in various ML tasks. However, notably, these dendritic neuron models serve as black box models when applied to ML problems. Similar to many sophisticated ML models, dendritic neuron models can make accurate predictions but provide limited explainability. Although research concerning the explainability of ML has become an important issue in the field of artificial intelligence [10,12,43], to the best of the knowledge of the authors, providing explainability to a dendritic neuron model has not yet been well explored.

In this study, we attempt to provide explainability to the proposed logic dendritic neuron model (LDNM) by using a tree-based model called the morphology of decision trees (MDT). We first review the definition of LDNM. Then, a structure pruning mechanism based on the characteristics of LDNM is proposed to simplify the structure of a trained

LDNM. Next, we propose a method to transform a pruned LDNM into an MDT, which can provide explainability to LDNM. Finally, six benchmark classification problems are used to verify the classification performance of LDNM, the effectiveness of the structure pruning, and the MDT transformation. The contribution of this paper is threefold. First, we propose a method to transform a pruned LDNM into an MDT. To the best of the knowledge of the authors, this transformation method has not been explored. Second, a global surrogate explanation approach is proposed for LDNM. Third, the effectiveness of the explanation approach is experimentally investigated.

The remainder of this paper is organized as follows. Section 2 presents the definition of the proposed LDNM and a description of the classification and regression tree (CART). Section 3 presents the learning method, structure pruning method, and MDT transformation method of LDNM. The experimental studies are provided in Section 4. Finally, the conclusion of this study is provided in Section 5.

2. Materials

In this section, we introduce two important concepts: LDNM and CART.

2.1. LDNM Formulation

A biological neuron is composed of the following three typical components: a cell body, dendrites, and a single axon. LDNM is composed of the following four components: synaptic layers, dendrite layers, a membrane layer, and a cell body. Figure 1 shows the architecture of the LDNM, and the four components are defined in the following.

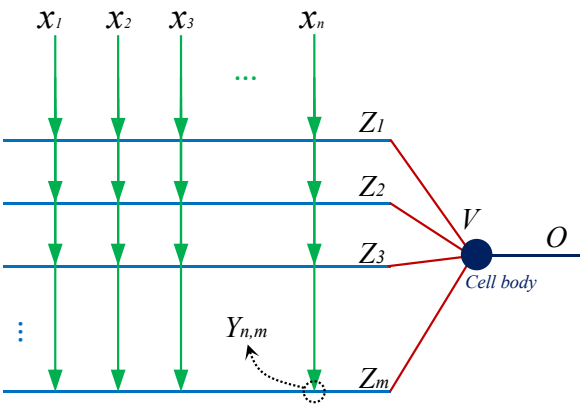


Figure 1. The architecture of LDNM.

Synaptic layer: The synaptic layers receive nerve signals from presynaptic neurons. A synapse connects a synaptic input with a dendritic branch, and a logistic function is employed to formulate this connection. The output $Y_{i,j}$ of the ith ($i = 1, 2, ..., n$) synaptic input located on the jth ($j = 1, 2, ..., m$) dendrite branch is expressed as follows:

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

where k is a positive parameter. x_i is the ith ($i = 1, 2, ..., n$) synaptic input and ranges from 0 to 1. $w_{i,j}$ and $q_{i,j}$ are the weights of a synapse and need to be adjusted by a learning algorithm. In addition, all $w_{i,j}$ and $q_{i,j}$ are randomly initialized in $[-1.5, 1.5]$ before training the LDNM.

Dendritic layer: These are multiplicative operations believed to exist on the dendrite tree of a biological neuron [44]. This operation can be considered a logical AND operation in the binary case. In the proposed LDNM, the jth dendritic layer is expressed as follows:

$$Z_j = \prod_{i=1}^{n} Y_{i,j} \tag{2}$$

where Z_j represents the output of the jth dendritic layer.

Membrane layer: The membrane layer processes the signals conveyed from all dendritic branches. A summation operation is employed to imitate the calculation performed in the membrane layer. This operation can be considered a logical OR operation when the dendritic signals are binary. The output of membrane layer V can be calculated as follows:

$$V = \sum_{j=1}^{m} Z_j \tag{3}$$

Cell body: The cell body is enclosed by its membrane. The cell body receives signals from the membrane and transports the processed signals up the axon. A logistic function is employed to simulate the information processing of the cell body as follows:

$$O = \frac{1}{1 + e^{-c(V-\gamma)}} \tag{4}$$

where c is a constant parameter. γ is the threshold used for the classification task. O is the actual output of the cell body.

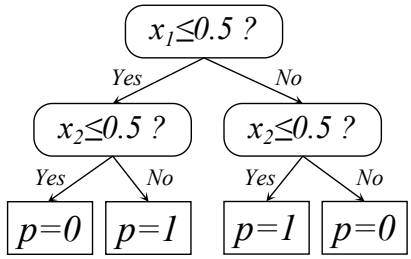
2.2. Classification and Regression Tree

Decision tree is a popular ML method that uses a tree-like structure to build a learning model. CART [45] is a typical decision tree model that constructs a binary tree for classification and regression tasks. There are two types of nodes in CART. Each internal node stores an integer d to label the tested feature and a threshold value φ. Each leaf node stores a variable p to represent a class label (for the classification task) or a real number (for the regression task). CART makes a prediction by executing a top-down search from root to leaf. In each internal node, if $x_d \leq \varphi$, the left child tree is further explored; otherwise, the right child tree is further explored. Figure 2 shows an example of CART for solving the XOR problem.

XOR problem

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)

Figure 2. An example of CART for solving the XOR problem. (**a**) Inputs and outputs of the XOR problem. (**b**) Desired solution for the XOR problem by using CART.

A trained CART can be translated into a set of decision rules. A decision rule is a decision path (starting from the root node to a leaf node) and has the following form:

IF(condition1 AND condition2 AND...) THEN outcome;

where the *condition* corresponds to the internal node, and the *outcome* corresponds to the leaf node. For example, the CART shown in Figure 2 can be translated into the following decision rules:

IF($x_1 <= 0.5$ *AND* $x_2 <= 0.5$) *THEN* $p = 0$;
IF($x_1 <= 0.5$ *AND* $x_2 > 0.5$) *THEN* $p = 1$;
IF($x_1 > 0.5$ *AND* $x_2 <= 0.5$) *THEN* $p = 1$;
IF($x_1 > 0.5$ *AND* $x_2 > 0.5$) *THEN* $p = 0$;

Moreover, for binary classification, the aforementioned *IF-THEN* statements with $p = 0$ can be merged into an *ELSE* statement.

The decision tree method is considered simpler and more interpretable than other ML methods because the workflow of decision trees is very similar to human decision behaviors. A trained decision tree can easily be visualized and understood.

3. Methodology

In this section, we provide the details of the proposed approach.

3.1. Learning Method

Since the proposed LDNM is a feedforward neural network, the backpropagation (BP) algorithm is employed as the learning method [39]. Given the training data with P samples, the error function of LDNM in the pth training sample is defined as follows:

$$E_p = \frac{1}{2}(T_p - O_p)^2 \tag{5}$$

T_p is the target output in the pth training sample, and O_p is the actual output of LDNM. At each training epoch, the synaptic weights $w_{i,j}$ and $q_{i,j}$ are adjusted based on the gradient descent method as follows:

$$w_{i,j} = w_{i,j} - \eta \sum_{p=1}^{P} \frac{\partial E_p}{\partial w_{i,j}} \tag{6}$$

$$q_{i,j} = q_{i,j} - \eta \sum_{p=1}^{P} \frac{\partial E_p}{\partial q_{i,j}} \tag{7}$$

where η is the learning rate.

3.2. Structure Pruning

Synaptic pruning is observed in the human brain, where many axons and dendrites are eliminated during youth in humans. Neural network pruning increases the efficiency of a trained neural network by removing redundant weights. A trained LDNM can also be pruned by specific mechanisms. The main idea of the specific mechanisms is that the synapses contributing minimally to the output of the trained LDNM can be removed. Thus, determining the unimportant synapses in a trained LDNM is the key issue in pruning LDNM.

Equation (1) describes the relation between the synaptic input x_i and the output $Y_{i,j}$, namely, the connection state. Obviously, the connection state is uniquely determined by the synaptic weights $w_{i,j}$ and $q_{i,j}$. According to the values of $w_{i,j}$ and $q_{i,j}$, a synapse in the trained LDNM stays in one state among the following four types of connection states: direct connection, inverse connection, constant 1 connection, and constant 0 connection. Four types of symbols are used to represent these connection states, as shown in Figure 3. Figure 4 shows how the synaptic weights $w_{i,j}$ and $q_{i,j}$ determine the connection state of a trained synapse. A parameter $\theta_{i,j} = q_{i,j}/w_{i,j}$ is defined as the threshold of the trained synapse. Obviously, $\theta_{i,j}$ is the midpoint of the modified logistic function (Equation (1)) on the horizontal axis as shown in Figure 4. The four connection states are explained as follows.

State 1.1: $w_{i,j} > 0$, $q_{i,j} < 0 < w_{i,j}$; for example, $w_{i,j} = 1$, $q_{i,j} = -0.5$, and $\theta_{i,j} = -0.5$ as shown in Figure 4a. This state is a constant 1 connection because the output is almost 1 whenever x_i ranges in $[0, 1]$.

State 1.2: $w_{i,j} < 0$, $q_{i,j} < w_{i,j} < 0$; for example, $w_{i,j} = -1$, $q_{i,j} = -1.5$, and $\theta_{i,j} = 1.5$ as shown in Figure 4d. This state is also a constant 1 connection for the abovementioned reason.

State 2: $w_{i,j} > 0$, $0 < q_{i,j} < w_{i,j}$; for example, $w_{i,j} = 1$, $q_{i,j} = 0.5$, and $\theta_{i,j} = 0.5$ as shown in Figure 4b. This state is a direct connection because a larger input x_i leads to a larger output $Y_{i,j}$.

State 3: $w_{i,j} < 0$, $w_{i,j} < q_{i,j} < 0$; for example, $w_{i,j} = -1$, $q_{i,j} = -0.5$, and $\theta_{i,j} = 0.5$ as shown in Figure 4e. This state is an inverse connection because a larger input x_i leads to a smaller output $Y_{i,j}$.

State 4.1: $w_{i,j} > 0$, $0 < w_{i,j} < q_{i,j}$; for example, $w_{i,j} = 1$, $q_{i,j} = 1.5$, and $\theta_{i,j} = 1.5$ as shown in Figure 4c. This state is a constant 0 connection because the output is almost 0 whenever x_i ranges in $[0, 1]$.

State 4.2: $w_{i,j} < 0$, $w_{i,j} < 0 < q_{i,j}$; for example, $w_{i,j} = -1$, $q_{i,j} = 0.5$, and $\theta_{i,j} = -0.5$ as shown in Figure 4f. This state is also a constant 0 connection for the abovementioned reason.

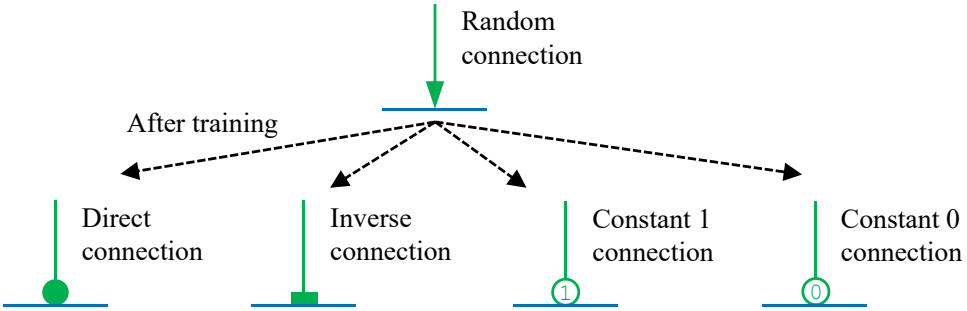


Figure 3. A synapse in a trained LDNM stays in one state among the following four types of connection states: direct connection, inverse connection, constant 1 connection, and constant 0 connection.

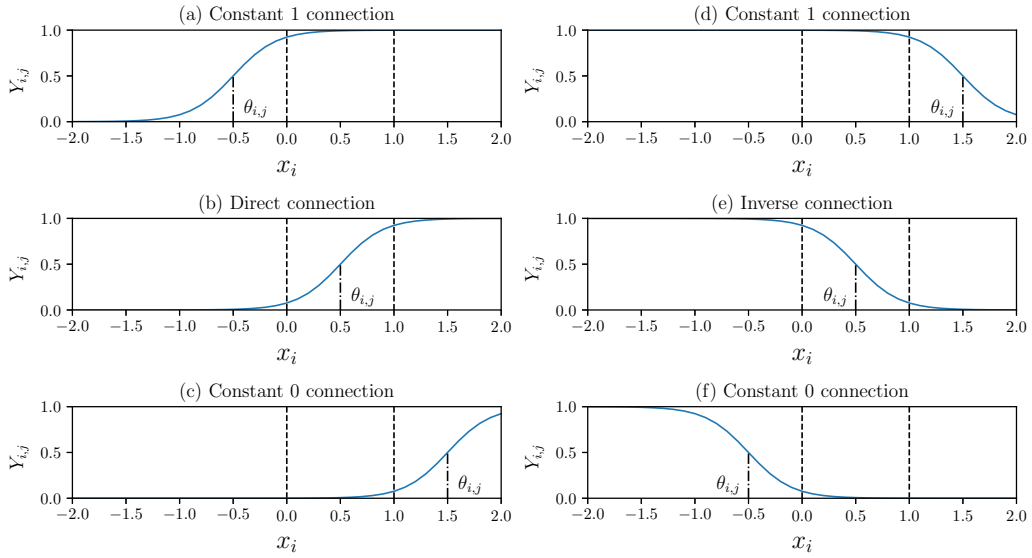
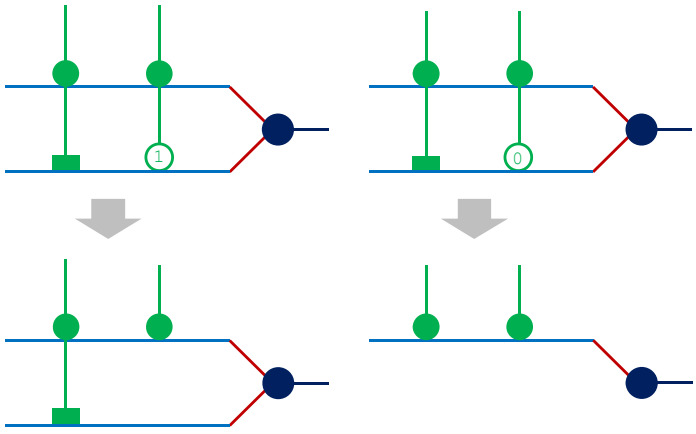


Figure 4. Four connection states are determined by the synaptic weights $w_{i,j}$ and $q_{i,j}$. Notably, a parameter $\theta_{i,j} = q_{i,j}/w_{i,j}$ is defined as the threshold of the trained synapse, and x_i ranges from 0 to 1.

After training, all synapses in an LDNM stay in one state of the four connection states. As described in Section 2.1, the multiplicative operations in the dendritic layers are considered logical AND operations in the binary case. Since the dendritic input of the constant 1 connection and the constant 0 connection are always 1 and 0, respectively, these two types of connection states can play special roles in the output of a trained LDNM. As a result, two pruning mechanisms can be performed in a trained LDNM.

Synaptic pruning: Given a synapse in a constant 1 state and a dendritic layer connected to this synapse, the synaptic output $Y_{i,j} = 1$ has no effect on the output of dendritic layer Z_j because of the Boolean operation "x AND 1 = x". Therefore, this synapse in constant 1 is redundant and can be removed from the trained LDNM. Figure 5a shows how the synaptic pruning operation functions.



(a) Synaptic pruning (b) Dendritic pruning

Figure 5. Two examples showing how synaptic pruning and dendritic pruning are performed in trained LDNMs.

Dendritic pruning: Given a synapse in a constant 0 state and a dendritic layer connected to this synapse, the synaptic output $Y_{i,j} = 0$ leads dendritic layer Z_j to always output 0 because of the Boolean operation "x AND 0 = 0". Moreover, the dendritic layer with $Z_j = 0$ has no effect on the output of the membrane layer because of the logical OR operation in the membrane layer. As a result, this dendritic layer is redundant and can be removed from the trained LDNM. Figure 5b shows how the dendritic pruning operation functions.

After synaptic pruning and dendritic pruning, all synapses in the constant 1 connection state and the constant 0 connection state and the redundant dendritic layers are removed from a trained LDNM. In addition, the structure of the pruned LDNM is much simpler than that of the original LDNM.

3.3. Transformation into a Morphology of Decision Trees

The logistic function and the step function are two common activation functions used in artificial neural networks [46]. The logistic function has the following form:

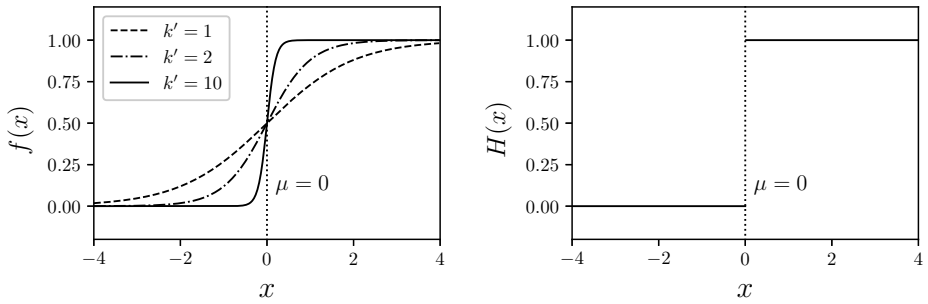
$$f(x) = \frac{1}{1 + e^{-k'(x-\mu)}} \tag{8}$$

where μ is the midpoint of the logistic function. k' ($k' > 0$) determines the steepness of the curve as shown in Figure 6a. The step function is defined as follows:

$$H(x) = \begin{cases} 0 & , x \leq \mu \\ 1 & , x > \mu \end{cases} \tag{9}$$

where μ is the threshold of the step function as shown in Figure 6b. The step function can be considered a smooth approximation of the logistic function if k' approaches positive infinity, suggesting that these two functions are interchangeable in some cases. In fact, the step function serves as the activation function in early artificial neural networks (ANNs), such as the McCulloch–Pitts neuron, where the aggregation of the weighted input is compared with a predefined threshold. Since the logistic functions are monotonic and derivable,

they fit the gradient-based training method and have become more popular in the field of ANNs.



(a) Logistic function (b) Step function

Figure 6. Curves of the logistic function and the step function. The step function can be considered a smooth approximation of the logistic function if k' approaches positive infinity.

As shown in Equation (1), a modified logistic function is used to model the synaptic layer. After synaptic pruning and dendritic pruning, only direct connections and inverse connections remain in the pruned LDNM. The discussion above suggests that the step function can serve as a substitute for modified logistic functions. For simplicity, the direct connection (Figure 4b) can be replaced with the following step function:

$$Y_{i,j} = \begin{cases} 0 & , x_i \leq \theta_{i,j} \\ 1 & , x_i > \theta_{i,j} \end{cases} \tag{10}$$

The inverse connection (Figure 4e) can be replaced with the following step function:

$$Y_{i,j} = \begin{cases} 1 & , x_i \leq \theta_{i,j} \\ 0 & , x_i > \theta_{i,j} \end{cases} \tag{11}$$

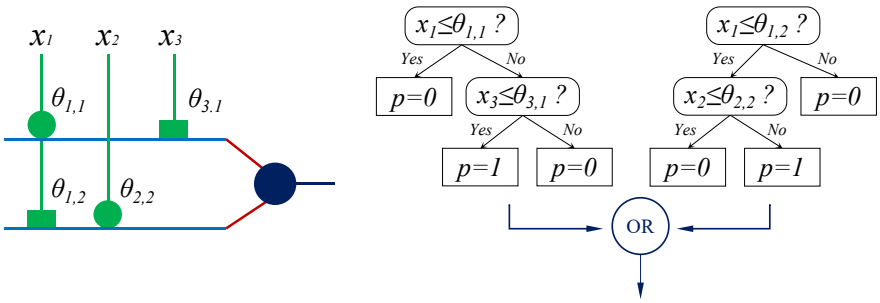
How a pruned LDNM outputs 1 is analyzed in the following. According to Equation (4), the membrane layer output $V = 1$ leads to the cell body outputting $O = 1$ in the binary case, and vice versa. Moreover, since the membrane layer performs a logical OR operation, the membrane layer outputs $V = 1$, while at least one dendritic layer outputs $Z_j = 1$. Furthermore, all synapses in the constant 1 connection state and the constant 0 connection state and the redundant dendritic layers are removed, and only the synapses in the direct connection state and inverse connection state exist on the dendritic branches. Since the dendritic layers perform a logical AND operation, a dendritic layer outputs $Z_j = 1$, while all synapses are in the direct connection state and the inverse connection output $Y_{i,j} = 1$. Considering that the synapses in the direct connection state and inverse connection state have the form of a step function (Equations (10) and (11)), the synaptic layer output $Y_{i,j}$ is determined by the comparison between the feature x_i and the threshold $\theta_{i,j}$. Therefore, the pruned LDNM can be expressed as a set of decision rules consisting of comparing, AND, and OR logistic operations.

Given the example of a pruned LDNM shown in Figure 7a, based on the aforementioned analysis, the structure of the pruned LDNM can be translated as a set of decision rules as follows:

IF($x_1 > \theta_{1,1}$ *AND* $x_3 <= \theta_{3,1}$) *THEN* $p = 1$;
IF($x_1 <= \theta_{1,2}$ *AND* $x_2 > \theta_{2,2}$) *THEN* $p = 1$;
ELSE $p = 0$;

It is obvious that this set of decision rules is similar to a set of decision rules of CART. For simplicity, each *IF-THEN* statement (corresponding to a dendritic branch) is combined with an *ELSE* $p = 0$ statement to construct an equivalent CART. As a result, two CARTs are

constructed, and these CARTs have a parallel relation. Finally, a morphology of decision trees (MDT) is generated based on the pruned LDNM as shown in Figure 7b. The prediction of MDT is achieved by performing the logistic OR operation based on the prediction of all individual CARTs.



(a) A pruned LDNM (b) Its corresponding MDT

Figure 7. An example of transforming a pruned LDNM into the MDT.

4. Experimental Studies

In this section, we present experimental studies to evaluate the performance of the proposed approach.

4.1. Experimental Setup

All algorithms used in this study are implemented in Python and C language. The experiments are conducted on a Linux system with a Core-i5 CPU and 16 GB RAM. According to our previous studies [39], the parameters of LDNM are set as follows: the number of dendritic branches $m = 2n$ (n is the number of synaptic inputs), the positive parameter $k = 5$, the constant parameter $c = 5$, and the threshold $\gamma = 0.5$. The parameters of the learning method of LDNM are set as follows: the learning rate $\eta = 0.01$, and the maximum number of epochs is set to 2000.

Six classification datasets, which can be accessed in the UCI Machine Learning Repository, are used as benchmark problems to evaluate the effectiveness of LDNM and other classifiers. These six datasets are iris, blood transfusion service center, glass identification, origin of wines, heart disease, and heart failure clinical records. In addition, all datasets are processed as binary classification problems. All features of each dataset are normalized in $[0, 1]$ to fit the input of LDNM. More details of these datasets are presented in Table 1.

Table 1. Details of the six benchmark datasets.

Dataset	Number of Samples	Number of Features	Number of Classes
Iris	150	4	2
Transfusion	748	4	2
Glass	214	9	2
Wine	178	13	2
Heart	270	13	2
HeartFailure	299	12	2

In the experiments, the stratified 5-fold cross-validation is conducted six times using each dataset to evaluate a classifier. Specifically, in a 5-fold cross-validation, the samples of a given dataset are randomly divided into five equal-sized groups. Four groups of samples are used to train the classifier, and the other group of samples is used as test data. Then, the cross-validation process is repeated five times in sequence. Finally, the experiments of training and testing a classifier in a benchmark problem are performed a total of 30 times.

4.2. Verifying the Classification Performance of LDNM

First of all, we confirm whether LDNM can serve as an effective ML model. To verify the classification performance of LDNM, we compare it with seven common classifiers. These classifiers include SVM, KNN, multilayer perceptron (MLP), naive Bayes (NB) classifier, decision tree, random forest, and AdaBoost. These classifiers are implemented using the ML library scikit-learn [47]. Their parameters are set according to the recommendation of scikit-learn and are presented in Table 2. For each classifier, the prepared datasets for the six classification problems are successively implemented 30 times as described above. The classification accuracies of LDNM and the seven classifiers are summarized in Table 3.

Table 2. Parameter settings of the seven classifiers.

Classifier	Key Parameters	Setting
SVM	Kernel Regularization parameter	Radial basis function 1.0
KNN	Number of neighbors	5
MLP	Size of the hidden layer Learning method	100 Adam
NB	Assumption for features	Gaussian distribution
Decision tree	Maximum depth	5
Random forest	Number of trees Maximum depth of each tree	10 3
AdaBoost	Number of estimators Base estimator	50 Decision tree

Table 3. Comparing the classification accuracies (%) of the eight classifiers.

Datasets	KNN	SVM	Decision Tree	Random Forest	MLP	NB	AdaBoost	LDNM
Iris	**95.56 ± 2.90**	67.22 ± 1.51	94.44 ± 3.88	92.33 ± 4.73	72.11 ± 11.17	91.67 ± 4.28	93.78 ± 3.92	95.56 ± 3.26
Transfusion	77.25 ± 2.62	76.20 ± 0.25	77.18 ± 2.64	76.38 ± 0.61	77.05 ± 1.38	75.06 ± 2.01	78.12 ± 2.25	**79.36 ± 1.97**
Glass	92.91 ± 3.72	91.12 ± 3.43	94.40 ± 3.32	90.03 ± 3.96	92.29 ± 3.66	90.58 ± 3.20	**94.54 ± 3.59**	93.29 ± 3.72
Wine	95.79 ± 3.32	**97.74 ± 1.70**	91.83 ± 4.94	92.14 ± 3.79	97.56 ± 1.90	95.58 ± 3.09	96.61 ± 3.22	94.82 ± 4.24
Heart	80.25 ± 5.16	83.46 ± 4.38	76.79 ± 6.36	79.01 ± 5.35	84.14 ± 4.87	**84.69 ± 4.78**	79.38 ± 5.28	81.73 ± 5.51
HeartFailure	68.39 ± 4.17	71.57 ± 1.87	79.54 ± 4.49	72.80 ± 2.90	82.44 ± 4.06	76.47 ± 3.68	80.66 ± 4.09	**83.95 ± 4.27**

According to Table 3, KNN, SVM, NB, AdaBoost, and LDNM exhibit the best performance in one, one, one, one, and two classification problems, respectively. Decision tree, random forest, and MLP do not yield the best results in any classification problem. It is obvious that the proposed LDNM is considered a very effective classifier for solving these classification problems with respect to the number of best performances. Moreover, the Friedman test is used to quantitatively compare these classifiers. The Friedman test ranks the eight classifiers according to their performance and provides the statistical significance. A smaller ranking value represents better performance by a classifier. The statistical results obtained by the Friedman test are presented in Table 4. According to Table 4, LDNM is considered to exhibit the best performance in six classification problems because it obtains the smallest ranking value of 2.67. In addition, the p-values of each classifier are provided in Table 4. To avoid a Type I error [48], a post hoc method, i.e., Holm's procedure, is used to adjust the p-values to p_{Holm} values. As shown in Table 4, the p_{Holm} value of random forest is smaller than the significance level (0.05). This finding suggests that the performance of LDNM is significantly better than that of random forest in the six classification problems. However, the p_{Holm} values of SVM, NB, decision tree, KNN, MLP, and AdaBoost are larger than 0.05. Thus, LDNM is not significantly superior to these six classifiers. Overall, the comparison result suggests that the proposed LDNM can achieve very competitive performance compared to these classic classifiers. LDNM can be considered an effective classifier in terms of classification accuracy.

Table 4. Friedman test ranks the eight classifiers according to their classification performance (in Table 3).

Classifier	Ranking	p	p_{Holm}
Random forest	6.50	0.006717	0.047017
SVM	5.33	0.059346	0.356079
NB	5.33	0.059346	0.356079
Decision tree	4.83	0.125506	0.502026
KNN	4.33	0.238593	0.715778
MLP	3.83	0.409395	0.818791
AdaBoost	3.17	0.723674	0.818791
LDNM	2.67	-	-

4.3. Investigating the Effectiveness of Structure Pruning and MDT Transformation

The effectiveness of the structure pruning and the MDT transformation is verified in this subsection. For each benchmark problem, the LDNM is applied to the training datasets 30 times as described above. Then, the performance of the trained LDNMs, the corresponding pruned LDNMs and the corresponding MDTs is verified in the same test datasets. The classification accuracies of the three classifiers are presented in Table 5. It is obvious that compared with the trained LDNM, the degradation of the classification capability of the pruned LDNM is less than 1%, except for two problems (i.e., Heart and HeartFailure). Moreover, compared with the trained LDNM, the degradation of the classification capability of the MDT is commonly less than 3%. This finding indicates that the structure pruning and the MDT transformation are considered effective because the classification accuracy is not greatly sacrificed. Moreover, we analyze the precision and recall scores of the three classifiers. As shown in Tables 6 and 7, the precision scores of the pruned LDNM and MDT are lower than those of the trained LDNM in most cases. However, the recall scores of the pruned LDNM and MDT are higher than those of the trained LDNM in most cases. This finding indicates that structure pruning and MDT transformation improve the ability to detect positives but sacrifice the ability to predict true positives accurately.

Table 5. Comparisons of the classification accuracies of the LDNM, the pruned LDNM, and the MDT.

Dataset	LDNM		Pruned LDNM		MDT
	Accuracy (%)	Num. of Branches	Accuracy (%)	Num. of Branches	Accuracy (%)
Iris	95.56 ± 3.26	8.00 ± 0.00	95.67 ± 3.12	1.43 ± 0.50	95.44 ± 3.04
Transfusion	79.36 ± 1.97	8.00 ± 0.00	79.01 ± 2.96	2.67 ± 0.60	76.22 ± 4.09
Glass	93.29 ± 3.72	18.00 ± 0.00	93.60 ± 4.01	2.63 ± 0.71	93.14 ± 4.10
Wine	94.82 ± 4.24	26.00 ± 0.00	94.83 ± 4.17	2.30 ± 0.69	92.94 ± 3.96
Heart	81.73 ± 5.51	26.00 ± 0.00	74.63 ± 9.49	3.17 ± 0.82	77.04 ± 6.30
HeartFailure	83.95 ± 4.27	24.00 ± 0.00	81.88 ± 5.43	3.47 ± 0.96	80.72 ± 5.26

Table 6. Comparisons of the precision scores (%) of the LDNM, the pruned LDNM, and the MDT.

Dataset	LDNM	Pruned LDNM	MDT
Iris	92.74 ± 6.68	92.74 ± 6.68	93.16 ± 5.74
Transfusion	64.87 ± 9.69	62.70 ± 11.48	51.05 ± 8.17
Glass	89.00 ± 9.65	88.23 ± 10.28	86.67 ± 10.84
Wine	96.17 ± 5.22	95.13 ± 5.20	91.51 ± 5.29
Heart	84.43 ± 7.30	68.93 ± 9.68	73.74 ± 8.84
HeartFailure	79.63 ± 7.39	69.96 ± 8.03	71.13 ± 10.96

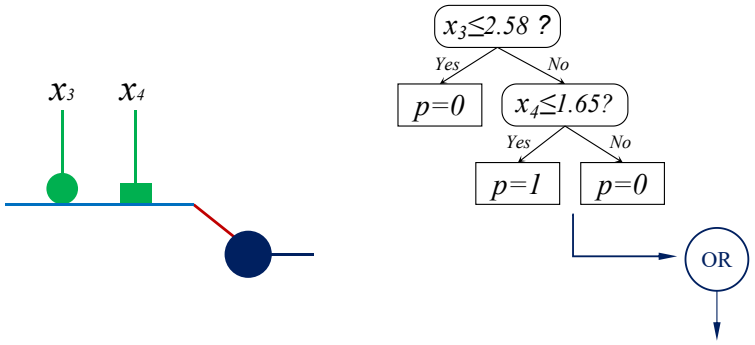
Table 7. Comparisons of the recall scores (%) of the LDNM, the pruned LDNM, and the MDT.

Dataset	LDNM	Pruned LDNM	MDT
Iris	94.67 ± 7.18	95.00 ± 6.19	93.67 ± 7.95
Transfusion	31.64 ± 8.47	34.64 ± 9.72	52.33 ± 7.72
Glass	83.27 ± 13.05	85.88 ± 12.72	85.52 ± 11.40
Wine	90.78 ± 8.50	91.95 ± 8.99	91.21 ± 9.17
Heart	72.78 ± 10.30	84.17 ± 9.77	77.64 ± 9.83
HeartFailure	68.11 ± 12.47	78.67 ± 12.25	72.41 ± 12.99

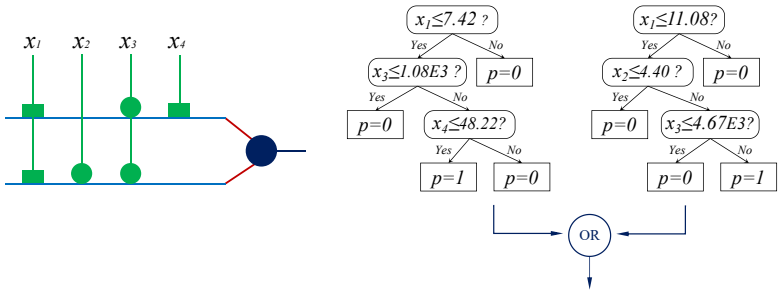
4.4. Exhibiting the Proposed Explanation Approach

For each benchmark problem, a typical pruned LDNM and the corresponding MDT are exhibited in Figures 8 and 9. It can be observed that the structures of the pruned LDNMs are concise. Table 5 reports the number of branches of the trained LDNM and the pruned LDNM for each benchmark problem. The number of branches of the pruned LDNMs is significantly reduced compared with the trained LDNM. Less than four dendritic branches are retained in most pruned LDNMs. Moreover, as shown in Figures 8 and 9, a few feature inputs are retained in the pruned LDNM. Therefore, some irrelevant feature inputs are also removed by the structure pruning operations. Overall, the structure pruning mechanisms can greatly simplify the structure of a trained LDNM. Usually, a small number of synapses in the direct connection and the inverse connection are retained in the pruned LDNMs.

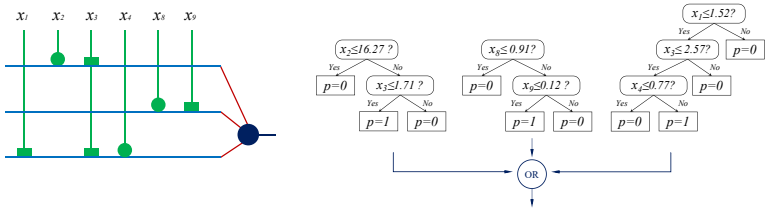
As shown in Figures 8 and 9, the MDTs are considered more interpretable than the pruned LDNMs because MDTs mainly comprise a small number of independent CARTs. Moreover, these CARTs have reasonable depths (usually less than four). The structures of MDTs can provide insight into how the classification results are obtained by the MDTs, the pruned LDNMs, and the trained LDNMs. As a result, the structures of MDTs help us understand the intrinsic characteristics of these benchmark problems and make LDNM explainable.



(a) Iris

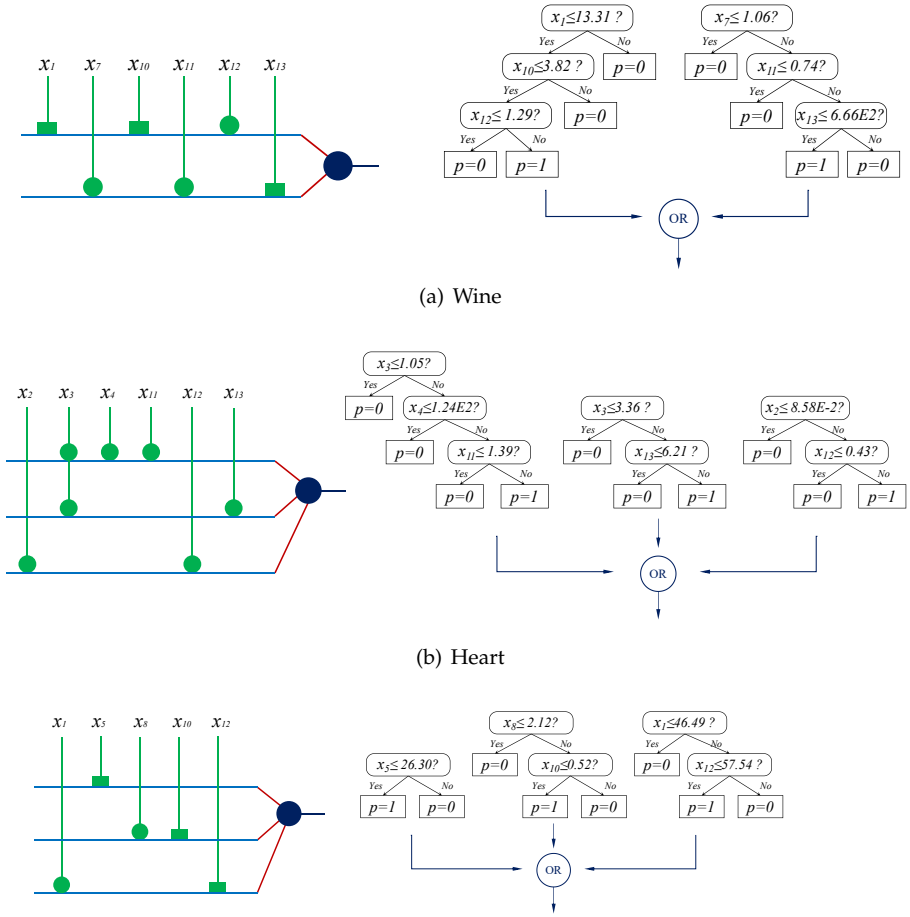


(b) Transfusion



(c) Glass

Figure 8. Typical pruned LDNMs and corresponding MDTs for three datasets (Iris, Transfusion, and Glass). Specifically, the thresholds in each MDT are reverted to the primitive values according to the original datasets (i.e., before normalization).



(a) Wine

(b) Heart

(c) HeartFailure

Figure 9. Typical pruned LDNMs and corresponding MDTs for three datasets (Wine, Heart, and HeartFailure). Specifically, the thresholds in each MDT are reverted to the primitive values according to the original datasets (i.e., before normalization).

4.5. Evaluation of the Explanation Approach

Evaluating an explanation approach is commonly considered subjective [9]. The aforementioned experimental studies have revealed the comprehensibility and accuracy of the proposed explanation approach. In this subsection, we provide qualitative evaluations, including fidelity and certainty, of the proposed explanation approach.

Since the proposed explanation approach is a global surrogate explanation approach, it is necessary to measure how well MDT approximates LDNM. The fidelity score is a common metric used to measure fidelity [10]. It is defined as the percentage of the matched prediction of the black box model and the surrogate explanation approach. Table 8 reports the fidelity scores to evaluate how well MDT approximates LDNM on the test datasets. It is obvious that the fidelity scores are considered high and are larger than 90% in most cases. This finding indicates that MDT can serve as a satisfactory global surrogate explanation approach for LDNM.

Table 8. Fidelity scores to evaluate how well MDT approximates LDNM on the test datasets.

Dataset	Fidelity Score (%)
Iris	99.22 ± 1.86
Transfusion	86.16 ± 5.10
Glass	97.97 ± 2.33
Wine	96.45 ± 2.97
Heart	88.89 ± 6.50
HeartFailure	90.41 ± 4.72

Although MDT is considered more concise and can provide satisfactory classification performance, we should note that MDT cannot completely replace LDNM in some practical applications. This is because MDT does not reflect the certainty of LDNM. LDNM performs a real number computation, while MDT performs a logical operation. LDNM can provide a prediction with a probability to show the confidence. However, MDT does not support probability prediction and is not a probabilistic classifier.

5. Conclusions

In this study, we proposed an artificial neuron model called LDNM for classification tasks. To provide explainability to LDNM, we attempted to transform a trained LDNM into a concise tree-based model called MDT. Specifically, we introduced a proprietary structure pruning mechanism to simplify a trained LDNM. Then, we proposed a method to transform the pruned LDNM into an MDT. Since the structure of the MDT mainly comprises some simple decision trees, the explainability of the LDNM for a specific problem can be gained based on the corresponding MDT. Finally, six benchmark classification problems were used to confirm the effectiveness of the proposed explanation approach for LDNM. The experimental results demonstrate that the MDT can approximate a trained LDNM well and provide explainability. As a result, an effective global surrogate explanation approach for LDNM is provided in this paper. The results of this study suggest that the explainability of other dendritic neuron models can be gained by similar decision tree-based explanation approaches.

Nevertheless, this study has two limitations. First, although the experimental results demonstrate the effectiveness of the proposed explanation approach, more high-dimensional classification problems could be used to further verify its effectiveness. Second, the proposed LDNM and its explanation approach are designed to solve binary classification problems. Multiclass classification techniques, such as the one-vs-rest strategy or the one-vs-one strategy, could be used to extend the proposed approach to multiclass classification problems. This issue deserves our further research.

In future studies, we intend to apply the proposed LDNM to more problems in sensitive domains because the explainability of LDNM can contribute to understanding the intrinsic characteristics of these problems. In addition, how to utilize the explainability of LDNM to improve its learning method is worth investigating.

Author Contributions: Conceptualization, X.C., H.F. and S.S.; methodology, X.C.; software, X.C. and S.S.; validation, W.C. and D.Z.; formal analysis, Y.Z. and D.Z.; investigation, H.F.; resources, H.F., W.C. and D.Z.; data curation, X.C. and Y.Z.; writing—original draft preparation, X.C.; writing—review and editing, H.F., W.C., Y.Z., D.Z. and S.S.; visualization, Y.Z. and D.Z.; supervision, H.F.; project administration, S.S.; and funding acquisition, W.C. and S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 62203069), the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20220619), the Qingpu District Industry University Research Cooperation Development Foundation of Shanghai (No. 202314).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: All data are contained within the article.

Conflicts of Interest: Author W. Chen was employed by the company Shanghai Huace Navigation Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Vamathevan, J.; Clark, D.; Czodrowski, P.; Dunham, I.; Ferran, E.; Lee, G.; Li, B.; Madabhushi, A.; Shah, P.; Spitzer, M.; et al. Applications of machine learning in drug discovery and development. *Nat. Rev. Drug Discov.* **2019**, *18*, 463–477. [CrossRef] [PubMed]
2. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A.; et al. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 583–589. [CrossRef] [PubMed]
3. Kobayashi, K.; Alam, S.B. Explainable, interpretable, and trustworthy AI for an intelligent digital twin: A case study on remaining useful life. *Eng. Appl. Artif. Intell.* **2024**, *129*, 107620. [CrossRef]
4. Jiménez-Luna, J.; Grisoni, F.; Schneider, G. Drug discovery with explainable artificial intelligence. *Nat. Mach. Intell.* **2020**, *2*, 573–584. [CrossRef]
5. Bibal, A.; Lognoul, M.; De Streel, A.; Frénay, B. Legal requirements on explainability in machine learning. *Artif. Intell. Law* **2021**, *29*, 149–169. [CrossRef]
6. Ding, W.; Abdel-Basset, M.; Hawash, H.; Ali, A.M. Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey. *Inf. Sci.* **2022**, *615*, 238–292. [CrossRef]
7. Sarvaiya, H.; Loya, A.; Warke, C.; Deshmukh, S.; Jagnade, S.; Toshniwal, A.; Kazi, F. Explainable Artificial Intelligence (XAI): Towards Malicious SCADA Communications. In *ISUW 2020: Proceedings of the 6th International Conference and Exhibition on Smart Grids and Smart Cities, Chengdu, China, 22–24 October 2022*; Springer: Singapore, 2022; pp. 151–162.
8. Imran, S.; Mahmood, T.; Morshed, A.; Sellis, T. Big data analytics in healthcare- A systematic literature review and roadmap for practical implementation. *IEEE/CAA J. Autom. Sin.* **2020**, *8*, 1–22. [CrossRef]
9. Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* **2019**, *267*, 1–38. [CrossRef]
10. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A survey of methods for explaining black box models. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–42. [CrossRef]
11. Belle, V.; Papantonis, I. Principles and Practice of Explainable Machine Learning. *Front. Big Data* **2021**, *4*, 688969. [CrossRef]
12. Burkart, N.; Huber, M.F. A survey on the explainability of supervised machine learning. *J. Artif. Intell. Res.* **2021**, *70*, 245–317. [CrossRef]
13. Goldstein, A.; Kapelner, A.; Bleich, J.; Pitkin, E. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *J. Comput. Graph. Stat.* **2015**, *24*, 44–65. [CrossRef]
14. Ribeiro, M.T.; Singh, S.; Guestrin, C. Anchors: High-precision model-agnostic explanations. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
15. Delgado-Panadero, Á.; Hernández-Lorca, B.; García-Ordás, M.T.; Benítez-Andrades, J.A. Implementing local-explainability in gradient boosting trees: Feature contribution. *Inf. Sci.* **2022**, *589*, 199–212. [CrossRef]
16. Deng, H. Interpreting tree ensembles with intrees. *Int. J. Data Sci. Anal.* **2019**, *7*, 277–287. [CrossRef]
17. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016 ; KDD '16, p. 1135–1144.
18. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In Proceedings of *the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Long Beach, CA, USA 2017; Volume 30.
19. Chen, H.; Covert, I.C.; Lundberg, S.M.; Lee, S.I. Algorithms to estimate Shapley value feature attributions. *Nat. Mach. Intell.* **2023**, *5*, 590–601. [CrossRef]
20. Zilke, J.R.; Loza Mencía, E.; Janssen, F. DeepRED—Rule Extraction from Deep Neural Networks. In *Proceedings of the Discovery Science*; Springer: Cham, Switzerlands, 2016; pp. 457–473.
21. Wu, M.; Hughes, M.; Parbhoo, S.; Zazzi, M.; Roth, V.; Doshi-Velez, F. Beyond sparsity: Tree regularization of deep models for interpretability. In Proceedings of the the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
22. Lundberg, S.M.; Erion, G.; Chen, H.; DeGrave, A.; Prutkin, J.M.; Nair, B.; Katz, R.; Himmelfarb, J.; Bansal, N.; Lee, S.I. From local explanations to global understanding with explainable AI for trees. *Nat. Mach. Intell.* **2020**, *2*, 56–67. [CrossRef]
23. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
24. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386. [CrossRef]
25. Gidon, A.; Zolnik, T.A.; Fidzinski, P.; Bolduan, F.; Papoutsi, A.; Poirazi, P.; Holtkamp, M.; Vida, I.; Larkum, M.E. Dendritic action potentials and computation in human layer 2/3 cortical neurons. *Science* **2020**, *367*, 83–87. [CrossRef]
26. Euler, T.; Detwiler, P.B.; Denk, W. Directionally selective calcium signals in dendrites of starburst amacrine cells. *Nature* **2002**, *418*, 845–852. [CrossRef]
27. Poirazi, P.; Brannon, T.; Mel, B.W. Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell. *Neuron* **2003**, *37*, 977–987. [CrossRef] [PubMed]
28. Todo, Y.; Tamura, H.; Yamashita, K.; Tang, Z. Unsupervised learnable neuron model with nonlinear interaction on dendrites. *Neural Netw.* **2014**, *60*, 96–103. [CrossRef] [PubMed]

29. Abbott, L.F.; DePasquale, B.; Memmesheimer, R.M. Building functional networks of spiking model neurons. *Nat. Neurosci.* **2016**, *19*, 350–355. [CrossRef]
30. Ji, J.; Gao, S.; Cheng, J.; Tang, Z.; Todo, Y. An approximate logic neuron model with a dendritic structure. *Neurocomputing* **2016**, *173*, 1775–1783. [CrossRef]
31. Gao, S.; Zhou, M.; Wang, Z.; Sugiyama, D.; Cheng, J.; Wang, J.; Todo, Y. Fully complex-valued dendritic neuron model. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 2105–2118. [CrossRef]
32. Kanamori, T.; Kanai, M.I.; Dairyo, Y.; Yasunaga, K.i.; Morikawa, R.K.; Emoto, K. Compartmentalized calcium transients trigger dendrite pruning in Drosophila sensory neurons. *Science* **2013**, *340*, 1475–1478. [CrossRef] [PubMed]
33. Faust, T.E.; Gunner, G.; Schafer, D.P. Mechanisms governing activity-dependent synaptic pruning in the developing mammalian CNS. *Nat. Rev. Neurosci.* **2021**, *22*, 657–673. [CrossRef]
34. Sossa, H.; Guevara, E. Efficient training for dendrite morphological neural networks. *Neurocomputing* **2014**, *131*, 132–142. [CrossRef]
35. Gómez-Flores, W.; Sossa, H. Smooth dendrite morphological neurons. *Neural Netw.* **2021**, *136*, 40–53. [CrossRef]
36. Luo, X.; Wen, X.; Zhou, M.; Abusorrah, A.; Huang, L. Decision-Tree-Initialized Dendritic Neuron Model for Fast and Accurate Data Classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 4173–4183. [CrossRef]
37. Bicknell, B.A.; Häusser, M. A synaptic learning rule for exploiting nonlinear dendritic computation. *Neuron* **2021**, *109*, 4001–4017. [CrossRef] [PubMed]
38. Ji, J.; Song, S.; Tang, Y.; Gao, S.; Tang, Z.; Todo, Y. Approximate logic neuron model trained by states of matter search algorithm. *Knowl.-Based Syst.* **2019**, *163*, 120–130. [CrossRef]
39. Song, S.; Chen, X.; Song, S.; Todo, Y. A neuron model with dendrite morphology for classification. *Electronics* **2021**, *10*, 1062. [CrossRef]
40. Song, S.; Xu, Q.; Qu, J.; Song, Z.; Chen, X. Training a Logic Dendritic Neuron Model with a Gradient-Based Optimizer for Classification. *Electronics* **2022**, *12*, 94. [CrossRef]
41. Song, S.; Zhang, B.; Chen, X.; Xu, Q.; Qu, J. Wart-Treatment Efficacy Prediction Using a CMA-ES-Based Dendritic Neuron Model. *Appl. Sci.* **2023**, *13*, 6542. [CrossRef]
42. Song, Z.; Tang, C.; Song, S.; Tang, Y.; Li, J.; Ji, J. A complex network-based firefly algorithm for numerical optimization and time series forecasting. *Appl. Soft Comput.* **2023**, *137*, 110158. [CrossRef]
43. Bonifazi, G.; Cauteruccio, F.; Corradini, E.; Marchetti, M.; Terracina, G.; Ursino, D.; Virgili, L. A model-agnostic, network theory-based framework for supporting XAI on classifiers. *Expert Syst. Appl.* **2024**, *241*, 122588. [CrossRef]
44. Gabbiani, F.; Krapp, H.G.; Koch, C.; Laurent, G. Multiplicative computation in a visual neuron sensitive to looming. *Nature* **2002**, *420*, 320–324. [CrossRef]
45. Carrizosa, E.; Molero-Río, C.; Romero Morales, D. Mathematical optimization in classification and regression trees. *Top* **2021**, *29*, 5–33. [CrossRef]
46. Apicella, A.; Donnarumma, F.; Isgrò, F.; Prevete, R. A survey on modern trainable activation functions. *Neural Netw.* **2021**, *138*, 14–32. [CrossRef]
47. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
48. García, S.; Fernández, A.; Luengo, J.; Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Inf. Sci.* **2010**, *180*, 2044–2064. [CrossRef]