Article

Dendrite Net with Acceleration Module for Faster Nonlinear Mapping and System Identification

Gang Liu [1,2], Yajing Pang [1], Shuai Yin [2], Xiaoke Niu [1,3,*], Jing Wang [2,*] and Hong Wan [1,3,*]

1 School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China
2 Institute of Robotics and Intelligent Systems, Xi'an Jiaotong University, Xi'an 710049, China
3 Henan Provincial Key Laboratory of Brain Science and Brain-Computer Interface Technology, Zhengzhou 450001, China
* Correspondence: niuxiaoke@zzu.edu.cn (X.N.); wangpele@gmail.com (J.W.); wanhong@zzu.edu.cn (H.W.)

Abstract: Nonlinear mapping is an essential and common demand in online systems, such as sensor systems and mobile phones. Accelerating nonlinear mapping will directly speed up online systems. Previously the authors of this paper proposed a Dendrite Net (DD) with enormously lower time complexity than the existing nonlinear mapping algorithms; however, there still are redundant calculations in DD. This paper presents a DD with an acceleration module (AC) to accelerate nonlinear mapping further. We conduct three experiments to verify whether DD with AC has lower time complexity while retaining DD's nonlinear mapping properties and system identification properties: The first experiment is the precision and identification of unary nonlinear mapping, reflecting the calculation performance using DD with AC for basic functions in online systems. The second experiment is the mapping precision and identification of the multi-input nonlinear system, reflecting the performance for designing online systems via DD with AC. Finally, this paper compares the time complexity of DD and DD with AC and analyzes the theoretical reasons through repeated experiments. *Results:* DD with AC retains DD's excellent mapping and identification properties and has lower time complexity. *Significance:* DD with AC can be used for most engineering systems, such as sensor systems, and will speed up computation in these online systems.

Keywords: dendrite net; online systems; nonlinear mapping; time complexity; engineering

MSC: 68-04; 68M07; 68N30; 93C10; 94-10

1. Introduction

The development of online systems, such as sensor systems, mobile phones, and computers, is changing the world we live in [1,2]. Nowadays, attention has been paid to the running speed of online systems, as running speed is the evaluation index for system performance. Nonlinear mapping is an essential and common demand in online systems, such as basic function calculation (e.g., $y = sin(x)$, where map x to y) in computers [3–5]. The time complexity of nonlinear mapping directly affects the running speed of online systems [6]. However, after years of development, the speed of nonlinear mapping is becoming increasingly difficult to improve, and there is less and less research on improving the running speed of online systems by accelerating the nonlinear mapping speed.

It is typical to store and calculate basic nonlinear functions (nonlinear mapping) in a polynomial form in computers (e.g., $sin(x)$ in C [4], or $sin(x)$ in java [5]). This means that polynomial storage and calculation methods with lower time complexity will speed up the running speed of online systems. In mathematics and computer science, Horner's method (or Horner's scheme) is a well-established algorithm with lower time complexity ($O(n)$) for polynomial evaluation [7]. In 2021, the authors of this paper proposed DD [8]. DD can be seen as a polynomial form with lower time complexity than the traditional polynomial form, and the time complexity of DD is consistent with Horner's method.

During the year, DD has already been used in multiple areas, such as energy saving [9], spatiotemporal traffic flow data imputation [10], high-dimensional problems [11], image processing [12], multi-objective optimization [13], accuracy prediction of the RV reducer to be assembled [14], and precipitation correction in flood season [15]. Nevertheless, there are still some redundant calculations in DD when the order of the DD model is higher than the number of inputs.

This paper proposes an acceleration module for DD to reduce redundant calculation, named DD with AC, which further speeds up the computation in online systems. According to the theory of DD, DD with AC can be used for nonlinear mapping and system identification. Meanwhile, the time complexity of DD with AC should be lower than DD according to the aim of this study. Consequently, the corresponding theory of DD with AC is explored by the experiments here. The main contributions of this paper are presented as follows:

1. In this paper, the redundant calculations in DD are found, and some characteristics of the redundant calculations are given;
2. This paper presents an acceleration module for DD to reduce redundant calculations and presents DD with AC by theoretically analyzing the redundant calculations in DD;
3. The proposed concept is experimentally justified and computationally verified based on the theoretical analysis. After theoretical and experimental analysis, it is demonstrated that DD with AC can be used for nonlinear mapping and system identification with lower time complexity than DD.

The rest of the paper is organized as follows. Section 2 introduces DD and describes the design of DD with AC. Experiments and results are given in Section 3. Section 4 discusses some experimental results and the significance of this study, and the conclusions are drawn in Section 5.

2. Design of Dendrite Net with Acceleration Module

2.1. Dendrite Net and Its Redundant Calculations

2.1.1. Dendrite Net

In a previous study, the authors of this paper proposed a basic machine learning algorithm called Dendrite Net or DD with white-box properties, controlled accuracy to improve generalization, and low computational complexity [8]. DD's main concept is that if the output's logical expression contains the logical relationship of a class among inputs (*and\or\not*), the algorithm can recognize the class after learning [8]. DD is one of the dendrites of a Gang neuron (an improved artificial neuron) [16], and its essence is a specialized polynomial form.

DD consists of DD modules and linear modules (see Figure 1). The DD module is straightforward and is expressed as follows.

$$A^l = W^{l,l-1} A^{l-1} \circ X \tag{1}$$

where A^{l-1} and A^l are the inputs and outputs of the module. X denotes the inputs of DD. One of the elements in X can be set to 1 to generate a bias. $W^{l,l-1}$ is the weight matrix from the $(l-1)$-th module to the l-th module. $\circ$ denotes the Hadamard product. Hadamard product is used to construct interactive items (e.g., $x_1 x_2$). The last module of DD is the linear module, and the linear module is expressed as follows.

$$A^L = W^{L-1,L} A^{L-1} \tag{2}$$

where A^{L-1} and A^L are the inputs and outputs of the module. L expresses the number of modules. $W^{L,L-1}$ is the weight matrix from the $(L-1)$-th module to the L-th module.
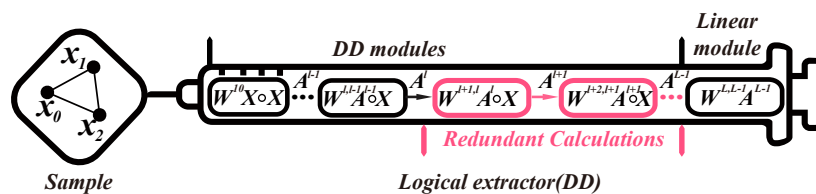
Figure 1. Schematic of dendrite net. DD aims to design the output's logical expression of the corresponding class (logical extractor).

The following set of equations describes the gradient descent rule of DD.
The forward propagation of DD module and linear module:

$$\begin{cases} A^l = W^{l,l-1} A^{l-1} \circ X \\ A^L = W^{L,L-1} A^{L-1} \end{cases} \tag{3}$$

The error-backpropagation of DD module and linear module:

$$dA^L = \widehat{Y} - Y \tag{4}$$

$$\begin{cases} dZ^L = dA^L \\ dZ^l = dA^l \circ X \end{cases} \tag{5}$$

$$dA^{l-1} = (W^{l,l-1})^T dZ^l \tag{6}$$

The weight adjustment of DD:

$$dW^{l,l-1} = \frac{1}{m} dZ^l (A^{l-1})^T \tag{7}$$

$$W^{l,l-1(new)} = W^{l,l-1(old)} - \alpha dW^{l,l-1} \tag{8}$$

where $\widehat{Y}$ and Y are DD's outputs and labels, respectively. m denotes the number of training samples in one batch. The learning rate α can either be adapted with epochs or fixed to a small number based on heuristics.

The most attractive feature is that there are only matrix multiplication and Hadamard product in DD operation, which confers a white-box property and lower time complexity onto DD. *White-box property*: The trained DD model can be translated into the Relation spectrum about inputs and outputs by formula simplification with software (e.g., *MATLAB*, an example in Figure 2). Concretely, the optimized weights are assigned to the corresponding matrices in Equations (1) and (2). Then the Relation spectrum was obtained through formula simplification in software. The white-box property solves the "black-box" issue of ML in Figure 3 [17]; thus, DD integrates nonlinear mapping/pattern recognition and system identification, compared to other ML algorithms (see Table 1). The lower time complexity makes it possible for DD to become a common algorithm in online systems.

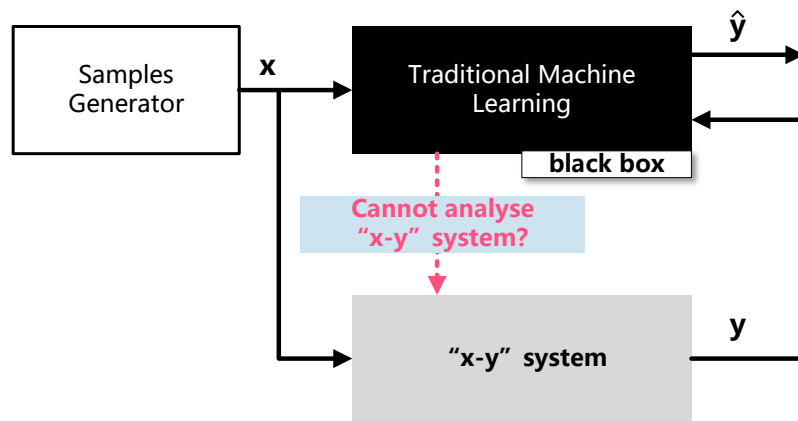**Figure 2.** The expanded form of DD.

Figure 3. "Black-box" issue of traditional machine learning (ML) [17]. Traditional ML can generate $\hat{y}$ approaching y via x, but cannot analyze the "x-y" system. Interestingly, the trained DD model can be transformed into a Relation spectrum such as the Taylor series for system identification [18,19]. [Analogous to the Fourier transform and Fourier spectrum used to decompose the signal, the DD and the Relation spectrum decompose the system.].

Table 1. Comparison between DD and typical algorithms.

Algorithms	Readability	Nonlinear Mapping (Online)	Analysis (Offline)
SVM, traditional NN, etc. [17,20]	Black box	Yes	No
Fourier transform and Fourier Spectrum [21]	White box	No	Yes (decomposing signal)
DD and Relation spectrum [8–15,18,19]	White box	Yes	Yes (decomposing system\model)

2.1.2. Redundant Calculations in Dendrite Net

DD constructs the interaction terms of the input variables and increases the order by concatenating DD modules (see Figures 1 and 2) [8]. However, when the order of the DD model is higher than the number of inputs, the previous DD module will build all the interaction terms, while the later DD module will only increase the order without increasing the interaction terms. For instance, in Figure 1, if the number of inputs is $l + 1$, l DD modules have constructed all the interactive items (the order of l DD modules is $l + 1$.), the later red DD modules will only increase the order and cannot increase the interactive items. Hence, the later DD modules have redundant calculations, and we can increase the order with lower computation than the DD module. Therefore, we replace the later DD modules with an acceleration module to increase the order faster in this paper (see Figure 4).
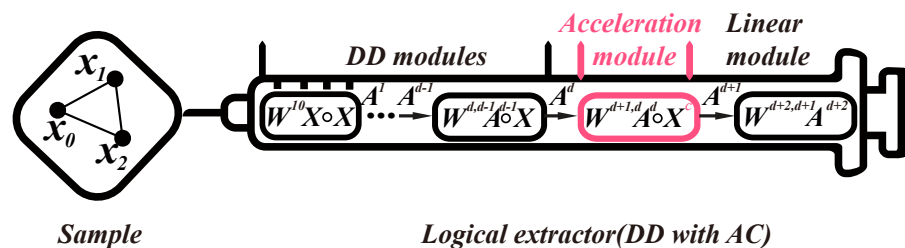

Figure 4. Schematic of the dendrite net with the acceleration module.

2.2. Dendrite Net with Acceleration Module

2.2.1. Architecture

Figure 4 shows the dendrite net with the acceleration module. It contains DD modules, a linear module, and an acceleration module. The overall architecture of DD with the acceleration module is shown in Figure 5. The architecture can be represented according to the following formula.

$$Y = W^{d+2,d+1}[W^{d+1,d}(\cdots W^{21}(W^{10}X \circ X) \circ X \cdots) \circ X^c], d \in N+ \tag{9}$$

where X and Y denote the input space and the output space. One of the elements in X can be set to 1 to generate a bias. $W^{i,i-1}$ is the weight matrix from the $(i-1)$-th module to the i-th module. The last module is linear. d denotes the number of DD modules. c denotes Power of Number. $\circ$ denotes Hadamard product.
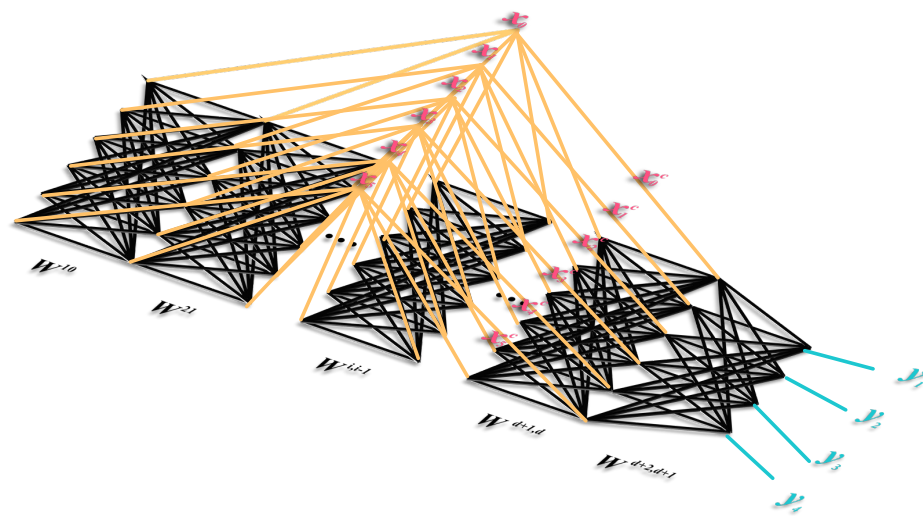

Figure 5. Overall architecture of DD with AC with six inputs and four outputs. This figure is a visual example of Equation (9).

The DD modules and the linear module have been described previously. The acceleration module is expressed as follows:

$$A^{d+1} = W^{d+1,d}A^d \circ X^c \tag{10}$$

where A^d and A^{d+1} are the inputs and outputs of the module. X denotes the inputs of DD with AC. One of the elements in X can be set to 1 to generate a bias. d denotes the number of DD modules. c denotes Power of Number. X^c represents X to the power of c. $W^{d+1,d}$ is the weight matrix from the d-th module to the $d+1$-th module. $\circ$ denotes Hadamard product.

In order for the DD with AC to include all terms under the target order and use fewer modules, the order of DD with AC n, the number of DD modules d, the power of AC c, and the input dimension of DD with AC a should satisfy the following equations:

$$\begin{cases} d+1+c = n \\ (c-1) \times a < n \\ c \times a \geq n \end{cases} \tag{11}$$

where $d+1+c = n$ means that the order of DD with AC is equal to the sum of the order of DD modules ($d+1$) and the order of the acceleration module (c). $(c-1) \times a < n$ means that if c exceeds this range (the value of c is too large), DD with AC can not construct all the interactive items. $c \times a \geq n$ means that if the value of c is too small, there are still redundant computations. d and c are calculated by Algorithm 1 in the case of a given order n.

Algorithm 1: Design of DD with AC

Input: The order of DD with AC *n*, the input dimension of DD with AC *a*
Output: The number of DD modules *d*, the power of AC *c*

1 for *i = 0 to n* **do**
2　**if** $(i-1) \times a < n$ *and* $i \times a \geq n$ **then**
3　　$c \leftarrow i$
4　　$d \leftarrow n - c - 1$
5　　**if** $d < 1$ **then**
6　　　$d \leftarrow 1$
7　　　$c \leftarrow n - d - 1$

2.2.2. Learning Rules

The graphical illustration of the learning rule is shown in Figure 6. As an example, we use one-half of the mean squared error (MSE) as the loss function. The learning rules of DD modules and the linear module have been described previously. The following set of equations describes the error back-propagation-based learning rule of the acceleration module (see Figure 6) [22].
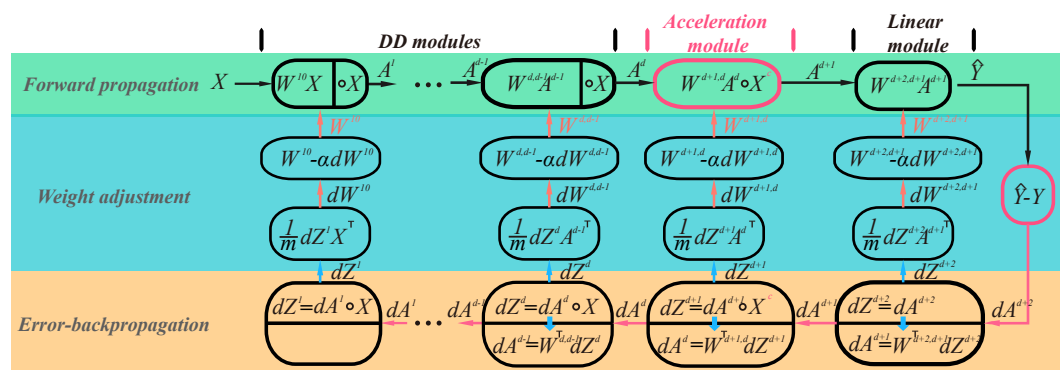
Figure 6. Graphical illustration of the learning rule.

The error back-propagation of the acceleration module:

$$dZ^{d+1} = dA^{d+1} \circ X^c \tag{12}$$

$$dA^d = (W^{d+1,d})^T dZ^{d+1} \tag{13}$$

The weight adjustment of the acceleration module:

$$dW^{d+1,d} = \frac{1}{m} dZ^{d+1} (A^d)^T \tag{14}$$

$$W^{d+1,d(new)} = W^{d+1,d(old)} - \alpha dW^{d+1,d} \tag{15}$$

where dA^{d+1} represents the error from the later module, *m* denotes the number of training samples in one batch, α is the learning rate, and the other symbols represent the intermediate variables or have been explained in Equation (10).

3. Experiments and Results

Ref. [8] demonstrates that DD has lower time complexity than traditional polynomials or ML with nonlinear functions, which will speed up the computation of online systems. In addition, DD has white-box properties and controllable accuracy for nonlinear mappings. The main purpose of the following experiments is to demonstrate the feasibility of DD with AC to further speed up the computation while retaining the properties of DD.

3.1. Precision and Identification of Unary Nonlinear Mapping

In order to investigate the precision and identification of unary nonlinear mapping, we considered the normalized Bessel function defined by:

$$f(x) = \frac{sin(x)}{x^2} - \frac{cos(x)}{x} \tag{16}$$

where we defined $x \in [-10, 0) \cup (0, 10]$, then x and $f(x)$ were normalized to $[-1, 1]$, respectively. We gradually increase the order of DD with AC to approximate the normalized Bessel function (from order 4 to order 15). The architectures of DD are designed by Ref. [8] and are shown in Table 2. The architectures of DD with AC are obtained from Algorithm 1, and the results are shown in Table 2. It is worth noting that the input dimension of DD or DD with AC is set to 2 and one of the inputs is always 1 (Bias).

Table 2. Architectures of DD and DD with AC for unary nonlinear mapping/four-input nonlinear system.

Order	DD	DD with AC	Number of Modules in DD	Number of Modules in DD with AC
4	3DD	1DD + AC2	3	2
5	4DD	1DD + AC3	4	2
6	5DD	2DD + AC3	5	3
7	6DD	2DD + AC4	6	3
8	7DD	3DD + AC4	7	4
9	8DD	3DD + AC5	8	4
10	9DD	4DD + AC5	9	5
11	10DD	4DD + AC6	10	5
12	11DD	5DD + AC6	11	6
13	12DD	5DD + AC7	12	6
14	13DD	6DD + AC7	13	7
15	14DD	6DD + AC8	14	7

A linear module follows each model, and the statistics for this module are not included in this table. "k"DD + AC"j": The model contains "k" DD modules and one acceleration module, and the power of *X* in the acceleration module is "j".

Figure 7 displays the precision of unary nonlinear mapping using DD with AC. What stands out in this figure is the gradual increase in precision with the increasing order, which is present in both DD and DD with AC. Furthermore, the precision of DD with AC is lower than DD in the same order, especially at higher orders. This may be due to the fact that it is more difficult to find the optimal solution using DD with AC than using DD. Although DD with AC has a drawback, it retains the key property of DD; that is, the precision increased with the number of modules, which corresponds to the property in Taylor's expansion [8].
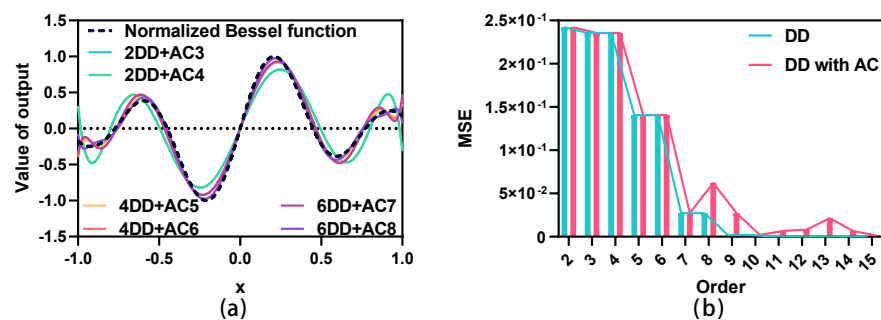

Figure 7. Precision comparison between DD and DD with AC for nonlinear mapping. (**a**) Nonlinear mapping using DD with AC. (**b**) Precision comparison between DD and DD with AC as the target order increases. "k"DD + AC"j": The model contains "k" DD modules and one acceleration module, and the power of *X* in the acceleration module is "j".

The trained DD with AC models were translated into the relation spectrum about inputs and outputs by formula simplification with MATLAB 2019b [13,18,19]. Concretely, we set system input variables $[1\ x]$ and use it to express the forward propagation formula (see Figure 4). Then, the optimized weights were assigned to the corresponding matrices in DD with AC. Finally, the Relation spectrum was obtained through formula simplification in MATLAB. *Note*: The Relation spectrum, similar to the Fourier spectrum, focuses on transforming and observing the corresponding phenomenon in the spectrum for analysis. The Relation spectrum presents the polynomial itself after the format transformation, so the transformation can be proved to be valid by observing whether there are similar relations in the Relation spectrum. More explanations and applications can be found in previous researches, such as Ref. [19], Ref. [18], and Ref. [13].

　　　　Turning now to the identification by DD with AC in Figure 8, the comparison between DD and DD with AC shows that DD with AC also retains the properties of the relation spectrum of DD. The difference in the Relation spectrum corresponds to the difference in precision in Figure 7. Models with large differences in precision also have large differences in the Relation spectrum. In other words, the Relation spectrum in Figure 8 explain the models in Figure 7.
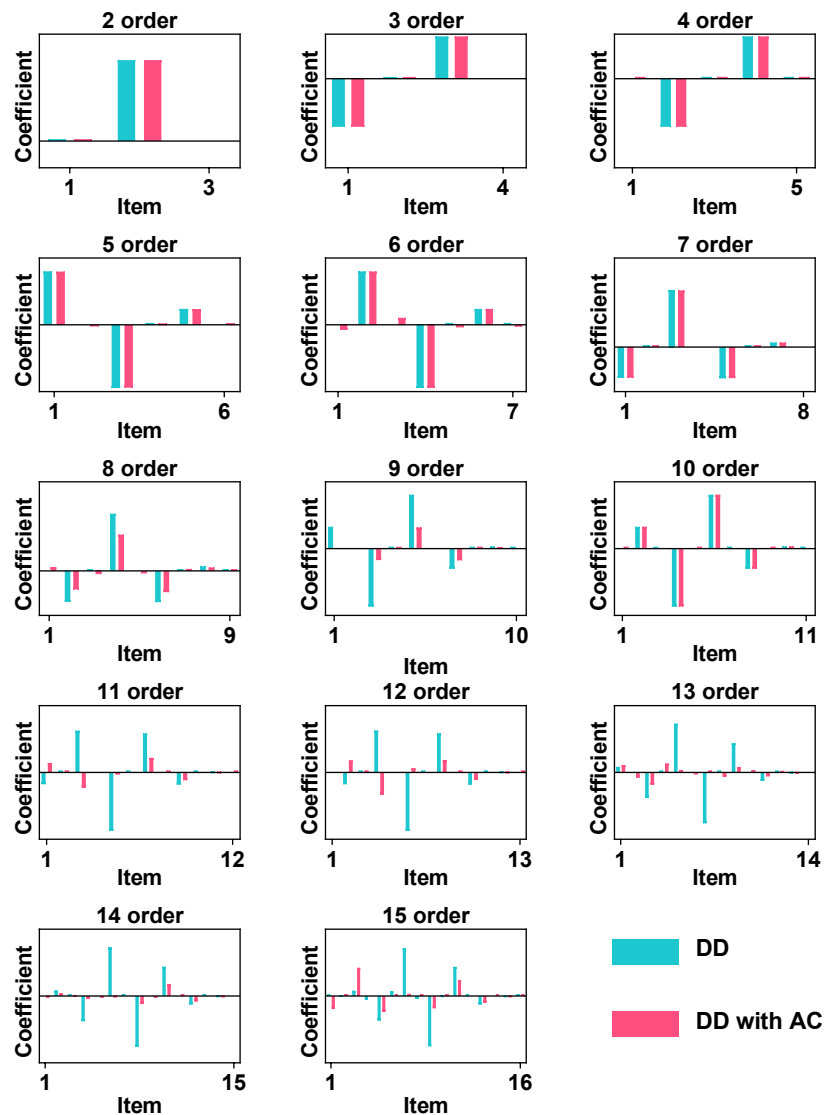

Figure 8. Identification comparison between DD and DD with AC for nonlinear mapping. The abscissa indicates the relation items. In analogy to the abscissa of Fourier spectrum, there are many relation items. Therefore, the relation items will not be listed here and can be obtained by looking up the table in the software.

3.2. Mapping Precision and Identification of Multi-Input Nonlinear System

Modeling a multi-input nonlinear system is an essential and common demand in online systems, such as sensor systems. We randomly constructed a multi-input nonlinear system, whose output is shown in Figure 9 (dotted line) and the inputs are defined by:

$$\begin{cases} I_1(t) = sin(2t) \\ I_2(t) = sin(3t) \\ I_3(t) = sin(5t) \end{cases} \quad (17)$$
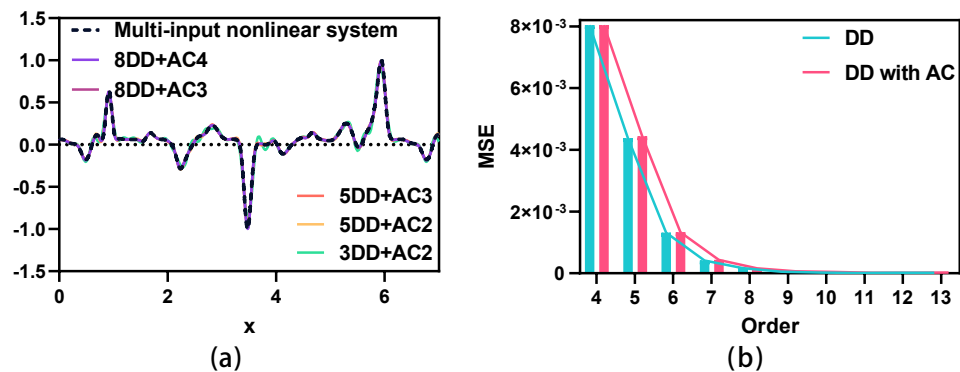
where we defined $t \in [0, 7]$.

Figure 9. Precision comparison between DD and DD with AC for multi-input nonlinear system. (**a**) Multi-input nonlinear system using DD with AC. (**b**) Precision comparison between DD and DD with AC as the target order increases. "k"DD + AC"j": The model has "k" DD modules and one acceleration module, and the power of X in the acceleration module is "j".

We gradually increase the order of DD and DD with AC to approximate the multi-input nonlinear system (from order 4 to order 13). The architectures of DD were designed according to the literature [8] (see Table 3). The architectures of DD with AC were obtained by Algorithm 1, and the results are given in Table 3. It is worth noting that the input dimension of DD or DD with AC is set to 4, and one of the inputs is always 1 (Bias).

Table 3. Architectures of DD and DD with AC for a four-input nonlinear system.

Order	DD	DD with AC	Number of Modules in DD	Number of Modules in DD with AC
4	3DD	2DD + AC1	3	3
5	4DD	2DD + AC2	4	3
6	5DD	3DD + AC2	5	4
7	6DD	4DD + AC2	6	5
8	7DD	5DD + AC2	7	6
9	8DD	5DD + AC3	8	6
10	9DD	6DD + AC3	9	7
11	10DD	7DD + AC3	10	8
12	11DD	8DD + AC3	11	9
13	12DD	8DD + AC4	12	9

A linear module follows each model, and the statistics for this module are not included in this table. "k"DD + AC"j": The model contains "k" DD modules and one acceleration module, and the power of X in the acceleration module is "j".

Figure 9 presents the precision of multi-input nonlinear system using DD with AC. What stands out in this figure is that the precision gradually improved with the increasing order, which exists in DD and DD with AC.

The trained DD with AC models were transformed into a Relation spectrum of the inputs and outputs by simplifying the equations in MATLAB 2019b [13,18,19]. Figure 10

provides an identification comparison between DD and DD with AC for a multi-input nonlinear system. The results between DD and DD with AC are similar, revealing that DD with AC also retains the properties of the Relation spectrum in DD. These properties have a wide range of applications, such as analyzing the human brain [19] and physical design [13].
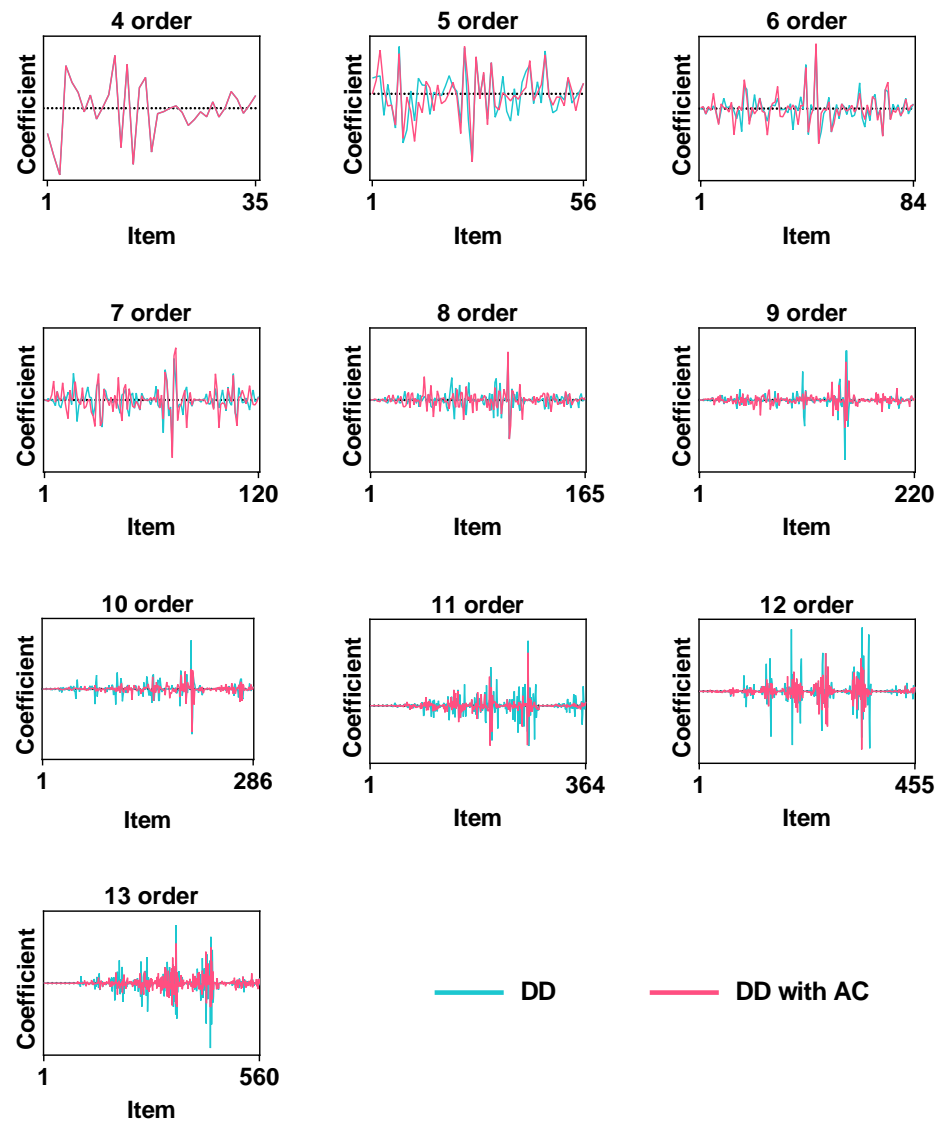
Figure 10. Identification comparison between DD and DD with AC for multi-input nonlinear system. The abscissa indicates the relation items. In analogy to the abscissa of Fourier spectrum, there are many relation items. Therefore, the relation items will not be listed here and can be obtained by looking up the table in the software.

3.3. Time Complexity

3.3.1. Computation of Time Complexity

Here we take unary nonlinear mapping as an example to calculate the time complexity. The time complexity of the modules is summarized in Table 4.

Table 4. Time complexity in DD and DD with AC for unary nonlinear mapping.

Module	Time Complexity
DD module ("$WA \circ X$")	6 multiplication, 2 addition
Linear module ("WA")	2 multiplication, 1 addition
Acceleration module ("$WA \circ X^c$")	$(4 + 2c)$ multiplication, 2 addition

Unary Nonlinear Mapping means that the input vector X contains two elements, one of which is 1, for example: $X = [1 \quad x]$. c denotes Power of Number. X^c represents X to the power of c.

DD is composed of DD modules and a linear module [8]. Therefore, DD contains $6(n-1) + 2$ multiplication and $2(n-1) + 1$ addition, where n denotes the order of polynomial. The time complexity of DD is $O(n)$, which happens to be consistent with Horner's method [7].

We take the two-input system (unary nonlinear mapping) and the four-input system using DD with AC as an example and display the number of modules required for the target order in Figure 11. Concretely, the architecture of DD with AC is obtained by Algorithm 1, and the results are shown in Tables 2 and 3. It is evident from Figure 11 and Table 4 that, compared with increasing one order by adding a DD module, the time complexity is reduced by 2 multiplications and 1 addition by adjusting the acceleration module. **Therefore, the time complexity of DD with AC is less than that of DD or Horner's method [7].**
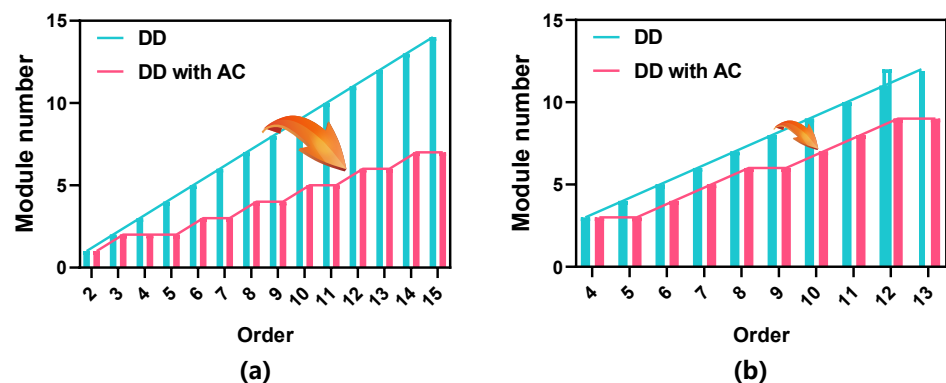

Figure 11. The number of modules required for target order. (**a**) Module number for two-input system. When one of the inputs is 1, it corresponds to the unary nonlinear mapping above. (**b**) Module number for four-input system. When one of the inputs is 1, it corresponds to the above multi-input nonlinear system.

3.3.2. Experiments of Time Complexity

In addition, to further verify the above results, 20 runs of DD and DD with AC were performed for the two-input system and the four-input system, and the run times (online speeds) were recorded. The tests were executed in *MATLAB 2021b* on a 2.2-GHz laptop Personal Computer (PC). Among them, we recorded the running time for 10,000 forward-propagation with 1000 samples for the 2-input systems and the running time for 10,000 forward-propagation with 7000 samples for the 4-input systems.

The results are ideal. All online speeds from the online tests met our expectations (see Figure 12). By comparing the results of Figures 11 and 12, it can be concluded that the online speed corresponds with the number of modules.
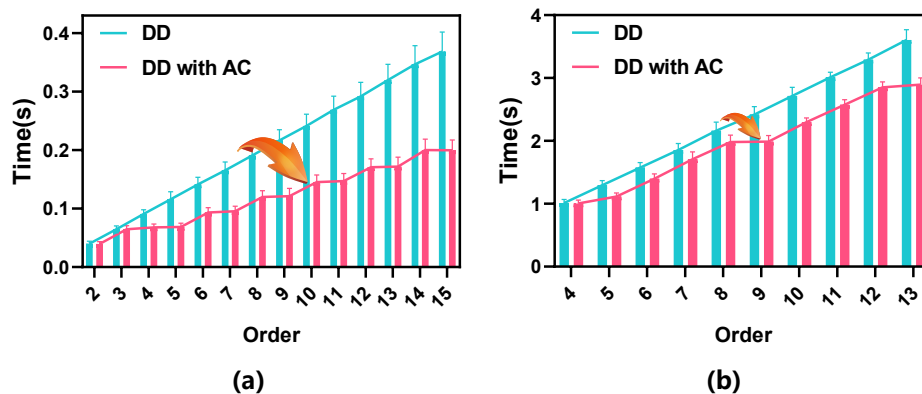
Figure 12. Online speed for a certain order. (**a**) Online speed for a two-input system. (**b**) Online speed for a four-input system. Data: $Mean \pm SD$.

4. Discussion

Prior studies have noted the importance of nonlinear mapping in online computation [1–5]. Our previous studies about DD observed faster speeds of nonlinear mapping using DD [8]. Due to the white-box attribute, controllable precision, and lower time complexity, DD was comprehensively applied in the areas of energy, traffic, weather, and physical design [9–15]. However, there has still been a redundant calculation in DD. This study aimed to eliminate the redundant computation while retaining DD's properties.

According to the analysis of DD, we designed an acceleration module, presented DD with AC, and conducted three experiments. These experimental results are in accord with the theoretical results. DD with AC has a lower time complexity than DD (see Figures 11 and 12), has controllable precision (see Figures 7 and 9), and can also be used for nonlinear mapping and system identification (see Figures 8 and 10). This paper was the continued work of previous studies about DD, which may improve the application of DD in various fields. Table 5 shows some examples of current applications of DD by referring to the reviewer's suggestions.

Table 5. Some applications of DD.

Applications	Literature
A hybrid data-driven framework for spatiotemporal traffic flow data imputation	Literature [10]
Energy saving of buildings for reducing carbon dioxide emissions using novel dendrite net integrated adaptive mean square gradient	Literature [9]
Unsteady aerodynamics modeling method based on dendrite-based gated recurrent neural network model	Literature [23]
A radial sampling-based subregion partition method for dendrite network-based reliability analysis	Literature [24]
An Algorithm for Precipitation Correction in Flood Season Based on Dendritic Neural Network	Literature [15]
Multi-Objective Optimization for the Radial Bending and Twisting Law of Axial Fan Blades	Literature [13]
An Accuracy Prediction Method of the RV Reducer to Be Assembled Considering Dendritic Weighting Function	Literature [14]
An Adaptive Dendrite-HDMR Metamodeling Technique for High-Dimensional Problems	Literature [11]
Convolutional dendrite net detects myocardial infarction based on ECG signal measured by flexible sensor	Literature [25]
Photovoltaic Power Prediction Under Insufficient Historical Data Based on Dendrite Network and Coupled Information Analysis	Literature [26]

The limitation of this paper comes from the lack of engineering data. In this paper, DD with AC is fundamentally verified by many experiments, which is a stronger way to prove it than using special data. Future experiments will be conducted on engineering problems.

5. Conclusions

This paper presents a Dendrite Net with an Acceleration module for nonlinear mapping and system identification. The theoretical and experimental results suggest that DD with AC retains DD's nonlinear mapping properties and system identification properties. Interestingly, the time complexity of DD is lower than the traditional polynomial or ML with a nonlinear function and is consistent with Horner's method. The time complexity of DD with AC is lower than DD or Horner's method, which provides a new strategy for online systems that require lower time complexity and has the potential to speed up the calculation of basic functions in computers.

Author Contributions: Conceptualization, G.L.; validation, G.L., Y.P. and S.Y.; formal analysis, X.N.; writing—original draft preparation, G.L and Y.P.; writing—review and editing, S.Y. and X.N.; supervision, J.W. and H.W.; project administration, H.W.; funding acquisition, X.N., J.W. and H.W. All authors have read and agreed to the published version of the manuscript.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pantelopoulos, A.; Bourbakis, N.G. A Survey on Wearable Sensor-Based Systems for Health Monitoring and Prognosis. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2010**, *40*, 1–12. [CrossRef]
2. Lane, N.D.; Miluzzo, E.; Lu, H.; Peebles, D.; Choudhury, T.; Campbell, A.T. A survey of mobile phone sensing. *IEEE Commun. Mag.* **2010**, *48*, 140–150. [CrossRef]
3. Sammon, J.W. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.* **1969**, *100*, 401–409. [CrossRef]
4. Opcactivex. TAYLOR_SIN in C. Available online: https://sourceware.org/git/?p=glibc.git;a=blob;f=sysdeps/ieee754/dbl-64/s_sin.c;hb=HEAD#l281 (accessed on 2 November 2022).
5. penJDK. How Is Sine Implemented in Java? Available online: https://stackoverflow.com/questions/26146190/how-is-sine-implemented-in-java (accessed on 2 November 2022).
6. Lim, T.S.; Loh, W.Y.; Shih, Y.S. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach. Learn.* **2000**, *40*, 203–228. [CrossRef]
7. Cajori, F. Horner's method of approximation anticipated by Ruffini. *Bull. Am. Math. Soc.* **1911**, *17*, 409–414. [CrossRef]
8. Liu, G.; Wang, J. Dendrite Net: A White-Box Module for Classification, Regression, and System Identification. *IEEE Trans. Cybern.* **2021**, 1–14. [CrossRef] [PubMed]
9. Han, Y.; Li, J.; Lou, X.; Fan, C.; Geng, Z. Energy saving of buildings for reducing carbon dioxide emissions using novel dendrite net integrated adaptive mean square gradient. *Appl. Energy* **2022**, *309*, 118409. [CrossRef]
10. Wang, P.; Hu, T.; Gao, F.; Wu, R.; Guo, W.; Zhu, X. A hybrid data-driven framework for spatiotemporal traffic flow data imputation. *IEEE Internet Things J.* **2022**, *9*, 16343–16352 [CrossRef]
11. Zhang, Q.; Wu, Y.; Lu, L.; Qiao, P. An Adaptive Dendrite-HDMR Metamodeling Technique for High-Dimensional Problems. *J. Mech. Des.* **2022**, *144*, 081701. [CrossRef]
12. Li, P.; Zhang, L.; Qiao, J.; Wang, X. A semantic segmentation method based on improved U-net network. In Proceedings of the 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Changsha, China, 26–28 March 2021; pp. 600–603.
13. Ding, Y.; Wang, J.; Jiang, B.; Li, Z.; Xiao, Q.; Wu, L.; Xie, B. Multi-Objective Optimization for the Radial Bending and Twisting Law of Axial Fan Blades. *Processes* **2022**, *10*, 753. [CrossRef]
14. Jin, S.; Chen, Y.; Shao, Y.; Wang, Y. An Accuracy Prediction Method of the RV Reducer to Be Assembled Considering Dendritic Weighting Function. *Energies* **2022**, *15*, 7069. [CrossRef]

15. Li, T.; Qiao, C.; Wang, L.; Chen, J.; Ren, Y. An Algorithm for Precipitation Correction in Flood Season Based on Dendritic Neural Network. *Front. Plant Sci.* **2022**, *13*, 862558. [CrossRef] [PubMed]

16. Liu, G. It may be time to improve the neuron of artificial neural network. *TechRxiv* **2020**. [CrossRef]

17. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef] [PubMed]

18. Liu, G.; Wang, J. A relation spectrum inheriting Taylor series: Muscle synergy and coupling for hand. *Front. Inf. Technol. Electron. Eng.* **2022**, *23*, 145–157. [CrossRef]

19. Liu, G.; Wang, J. EEGG: An Analytic Brain-Computer Interface Algorithm. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2022**, *30*, 643–655. [CrossRef] [PubMed]

20. Noble, W.S. What is a support vector machine? *Nat. Biotechnol.* **2006**, *24*, 1565–1567. [CrossRef] [PubMed]

21. Bracewell, R.N.; Bracewell, R.N. *The Fourier Transform and Its Applications*; McGraw-Hill: New York, NY, USA, 1986; Volume 31999.

22. LeCun, Y.; Touresky, D.; Hinton, G.; Sejnowski, T. A theoretical framework for back-propagation. In *Proceedings of the 1988 Connectionist Models Summer School*; Mogen Kaufmann: San Mateo, CA, USA, 1988; Volume 1, pp. 21–28.

23. Liu, K.; Huang, J.; Liu, Z.; Wang, Q. Unteady aerodynamics modeling method based on dendrite-based gated recurrent neural network model. In Proceedings of the 2022 International Conference on Machine Learning and Intelligent Systems Engineering (MLISE), Guangzhou, China, 5–7 August 2022; pp. 437–441.

24. Lu, L.; Wu, Y.; Zhang, Q.; Qiao, P.; Xing, T. A radial sampling-based subregion partition method for dendrite network-based reliability analysis. *Eng. Optim.* **2022**, 1–20. [CrossRef]

25. Ma, X.; Fu, X.; Sun, Y.; Wang, N.; Ning, X.; Gao, Y. Convolutional dendrite net detects myocardial infarction based on ecg signal measured by flexible sensor. In Proceedings of the 2021 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS), Manchester, UK, 20–23 June 2021; pp. 1–4.

26. Lu, T.; Wang, C.; Cao, Y.; Chen, H. Photovoltaic Power Prediction Under Insufficient Historical Data Based on Dendrite Network and Coupled Information Analysis. *SSRN* **2022**, 4184484. [CrossRef]