mathematics

Article

A Dendritic Neuron Model Optimized by Meta-Heuristics with a Power-Law-Distributed Population Interaction Network for Financial Time-Series Forecasting

Yuxin Zhang [1], Yifei Yang [2], Xiaosi Li [1], Zijing Yuan [2], Yuki Todo [3] and Haichuan Yang [1,2,*]

[1] Department of Engineering, Wesoft Company Ltd., Kawasaki-shi 210-0024, Japan
[2] Faculty of Engineering, University of Toyama, Toyama-shi 930-8555, Japan
[3] Faculty of Electrical and Computer Engineering, Kanazawa University, Kanazawa-shi 920-1192, Japan
* Correspondence: yokaisen1994@gmail.com

Abstract: The famous McCulloch–Pitts neuron model has been criticized for being overly simplistic in the long term. At the same time, the dendritic neuron model (DNM) has been shown to be effective in prediction problems, and it accounts for the nonlinear information-processing capacity of synapses and dendrites. Furthermore, since the classical error back-propagation (BP) algorithm typically experiences problems caused by the overabundance of saddle points and local minima traps, an efficient learning approach for DNMs remains desirable but difficult to implement. In addition to BP, the mainstream DNM-optimization methods include meta-heuristic algorithms (MHAs). However, over the decades, MHAs have developed a large number of different algorithms. How to screen suitable MHAs for optimizing DNMs has become a hot and challenging area of research. In this study, we classify MHAs into different clusters with different population interaction networks (PINs). The performance of DNMs optimized by different clusters of MHAs is tested in the financial time-series-forecasting task. According to the experimental results, the DNM optimized by MHAs with power-law-distributed PINs outperforms the DNM trained based on the BP algorithm.

Keywords: dendritic neuron model; meta-heuristic algorithms; financial time-series forecasting; population interaction networks

MSC: 37M10; 68W50; 68W40; 68T20

Citation: Zhang, Y.; Yang, Y.; Li, X.; Yuan, Z.; Todo, Y.; Yang, H. A Dendritic Neuron Model Optimized by Meta-Heuristics with a Power-Law-Distributed Population Interaction Network for Financial Time-Series Forecasting. *Mathematics* **2023**, *11*, 1251. https://doi.org/10.3390/math11051251

Academic Editor: Janez Žerovnik

Received: 10 February 2023
Revised: 27 February 2023
Accepted: 1 March 2023
Published: 4 March 2023

1. Introduction

Stock markets are frequently affected by micro-economic variables, such as capital market expectations, personal trader choices, and commercial organization operating policies, along with macro-economic variables, such as the economic environment and political issues [1]. As a result, because the precise effects of these factors on the vibrant monetary system are unidentified, forecasting stock market price changes is a significant difficulty for investors, speculators, and businesses [2], implying that realistic factors place higher demands on the accuracy of financial time-series forecasting (FTSF).

When attempting to predict the future, one expects that events will take place as predicted based on information and data about the past and present. Nevertheless, financial time series are among the most non-linear and challenging signals to forecast [3]. Traditional classical algorithms, such as the seasonal autoregressive integrated moving average (ARIMA) [4] and Holt–Winters method [5,6] make up the majority of the popularly used time-series-forecasting algorithms. In between, the ARIMA model is predicated on the idea that the data series is smooth, meaning that the mean and variance should not change over time and that the series can be made smooth using a logarithmic transformation or difference.

The Holt [5] and Winters [6] seasonal approach consists of a forecasting equation and three smoothing equations: the level, the trend, and the seasonal component equation, which are then superimposed on the elements in a cumulative or multiplicative manner to form the forecast. However, both methods are mathematical models that, while robust, have limited learning freedom and are less interpretable.

In the last decade, a diverse variety of machine-learning algorithms for driving solutions to time-series-forecasting challenges have been proposed [7], and previous research has demonstrated that machine learning is a cutting-edge learning technique that outperforms traditional models in terms of accuracy of time-series prediction [8]. Machine-learning techniques, typically artificial neural networks (ANN), play a significant role in time-series-prediction studies [9]. With neural networks, it is unnecessary to make precise assumptions about the model because data mining is used to identify all relevant correlations [10].

The fact that neural networks use a data-driven approach is one of their greatest advantages in confronting a variety of challenging real-world prediction difficulties. For instance, a recurrent neural network (RNN) for predicting financial time series, which uses ANNs to capture memory features via a feedback mechanism, can extricate temporal dependencies in financial data [11]. This RNN has more competitiveness than the well-known multi-layer perceptron (MLP) [3]. However, studies have demonstrated that it is difficult for the RNN to learn knowledge that has been preserved for a long time [12].

To tackle this question, the RNN-deduced long short-term memory (LSTM), which can recall both short-term and long-term values, was developed [13]. Nevertheless, LSTM, on the other hand, requires more computational resources and suffers from overfitting issues due to its deep structure [14]. Other ANN architectures, such as the spiking network [15] as well as the adaptive network-based fuzzy inference system (ANFIS) [16] have been used to address this problem and deliver positive outcomes.

McCulloch and Pitts proposed a mathematical model of neural activity occurring in the brain in 1943, which inspired ANNs [17]. Although ANN-based models are prominent, there are some drawbacks, such as the complexity of adjusting hyper-parameters, the need for a significant number of labeled training data, more repetitive model sophistication than required, and low comprehensibility due to their inherent black-box properties [18,19]. Admittedly, none of these works considered the fundamental concepts of deep neural networks—namely, the useful and effective information processing of a single neuron.

On the contrary, they typically enhanced the performances of the models by incorporating theoretical statistical elements or even more complex learning mechanisms. Simultaneously, rather than building deeper neural networks, some researchers remodeled neurons from scratch [20]. In addition, McCulloch–Pitts neurons, which only use a weight to depict the strength of a connection between two neurons, have long been criticized for being oversimplified [21].

Furthermore, a true biological neuron, which has both temporal and spatial properties, is only so complex. Thus, with sigmoid functions for the synaptic processing and a multiplication operation for the interaction within dendrites, a novel and biologically plausible dendritic neuron model (DNM) was introduced [22]. The DNM is characterized from almost all other ANNs by its distinct model architecture and excitation functions. DNM*, which is an improvement on DNMs, was applied to the FTSF problem in 2021, showing high application potential [23].

Despite the fact that DNMs have achieved considerable success in a variety of applications [24], they are largely trained by using the back-propagation (BP) learning algorithm, which typically encounters saddle-point diffusion and local-minimum trapping issues [25]. Due to these limits of BP, an increasing number of researchers have turned to non-BP algorithms—those that do not involve derivative computation but instead use an enhanced comparison-based solution—to train neuron models. The metaheuristic algorithms (MHAs) are widely used in DNM training as a long-established and time-tested optimization method [26,27]. However, as MHA research continues, new algorithms and improvements to existing algorithms are increasingly proposed [28].

The issue of how to filter the existing algorithms so that they can better train DNMs has become a major challenge for research. In fact, there have been some attempts to train DNMs using MHAs to predict financial time series [29,30]; however, these efforts compare and modify existing algorithms based on a specific algorithm and do not propose a generic mechanism for screening the algorithm. This leads to an alchemy dilemma [31] in the research of the FTSF problem, where researchers continue improving their optimization algorithms without being able to explain why specific changes are selected. This has further led to selecting suitable algorithms for training the DNMs without a theoretical basis and is highly dependent on researcher experience, which ultimately pushes up the time and economic costs.

The purpose of this research is to present the theory of population interaction networks (PINs) [32] to more accurately screen training algorithms for DNMs. PIN theory divides the cumulative distribution functions of evolutionary algorithms in MHAs into two types: power-law distribution and Poisson distribution. Inspired by the above theory, we propose a screening process for MHAs to quickly find a suitable DNM-optimizing algorithm. First, we chose the differential evolution (DE) [33] with a Poisson-distributed PIN and the adaptive differential evolution with optional external archive (JADE) [34] with a power-law-distributed PIN and compare the performance of a DNM trained by both algorithms on the FTSF problem.

Through testing, we found that the JADE-trained DNM outperformed the DE-trained DNM. This result suggests that MHAs with power-law-distributed PINs may be more advantageous for DNM training. Subsequently, we selected the success-history-based parameter adaptation for differential evolution (SHADE) [35] with a power-law-distributed PIN and spherical evolution (SE) [36] with a Poisson-distributed PIN for testing. The results demonstrate that the SHADE-trained DNM outperformed the SE-trained DNM for the FTSF problem.

Furthermore, this result also further validates our speculation that DNMs trained by MHAs with power-law-distributed PINs may be more advantageous in the FTSF problem. In addition, among the many MHAs, there are a large number of swarm-intelligence algorithms in addition to those based on evolutionary mechanisms. Among the swarm-intelligence algorithms, two typical algorithms, the genetic learning particle swarm optimization (GLPSO) [37] and spatial information sampling (SIS) algorithms [38], are examined. The results of the tests reveal that DNMs trained by the swarm-intelligence algorithm have no advantage over DNMs trained by MHAs with power-law-distributed PINs.

The main contributions of this study are summarized below:

(1) In this study, based on PIN theory, we build an efficient screening scheme for MHAs for the first time. This process enables researchers to select training algorithms for DNMs with a certain theoretical basis, thus, improving the screening efficiency of training algorithms.
(2) We find that DNMs optimized by MHA with power-law-distributed PINs may have higher accuracy in handling the FTSF problem.
(3) This research sheds new light on the design and improvement of DNM-training algorithms. Since a PIN is a branch of complex networks, more network architectures (e.g., small-world networks and scale-free networks) will be applied to the design and improvement of MHAs in the future to more efficiently optimize DNMs.

The remainder of the paper is structured as follows: the DNMs and MHAs are formulated in Section 2. The experimental results are analyzed in Section 3. Section 4 displays the discussions and conclusion.

2. Materials and Methods

In the following, we discuss DNMs for financial time-series forecasting as well as various types of MHAs for optimizing DNMs. The PIN-theory-based approach to classifying MHAs is the focus of this section, which concludes at the end of this section. After classifying MHAs, we test MHAs with different PINs and find the most suitable MHA to optimize DNMs to solve the FTSF problem.

2.1. Dendritic Neuron Model

A DNM is split into four different layers: the synaptic layer, the dendritic layer, the membrane layer, and the soma layer, as shown in Figure 1, which demonstrates the entire integration framework of a dendritic neuron model, which preferably resembles a biological neuron's features.

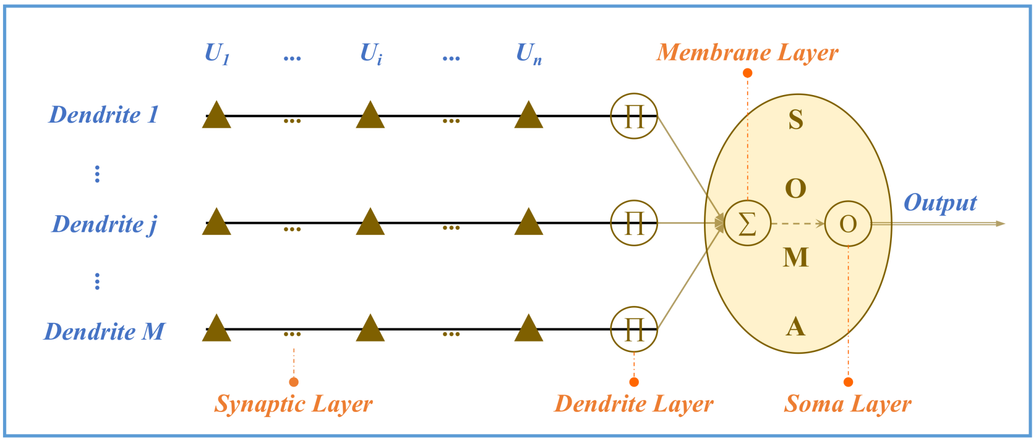


Figure 1. Dendritic neuron model.

2.1.1. Synaptic Layer

The synaptic layer processes signals from axon terminals to dendrites, and receptors on postsynaptic cells begin taking up a particular ion whose potential differs depending on the state of the synaptic connection (i.e., an inhibitory or excitatory synapse), where the n external input to the presynaptic axon terminal can be mathematically symbolized as $\{U_1, ..., U_i, ..., U_n\}$ $(i = 1, 2, ..., n)$. Furthermore, a sigmoid function is used to depict the synaptic layer's function [39], which is denoted as:

$$Z_{ij} = \frac{1}{1 + e^{-K(w_{ij}x_i - p_{ij})}} \quad (1)$$

In the equation, Z_{ij} indicates the ability of the jth $(j = 1, 2, ..., M)$ post-dendritic cell from the ith presynaptic axon terminal. K represents a distance parameter. In addition, there are two synaptic parameters: a synaptic weight called w_{ij}, which represents excitatory $(w_{ij} > 0)$ or inhibitory $(w_{ij} < 0)$ synaptic activity, and a threshold parameter called p_{ij}, both of which are trainable DNM parameters that together control the morphology of DNM dendrites and axons.

As demonstrated in Figure 2, learning algorithms, such as SHADE, GLPSO, and SIS, can be employed to imitate the synaptic plasticity mechanism by optimizing both w_{ij} and p_{ij}. Moreover, for the morphological conversion process, the four connection states biologically indicate the neuron's morphology by locally recognizing the position and synapse kinds of each dendrite. The mechanism of synaptic plasticity can be implemented after learning by dividing synaptic connections into four distinct states [24], as shown in Figure 3, including:

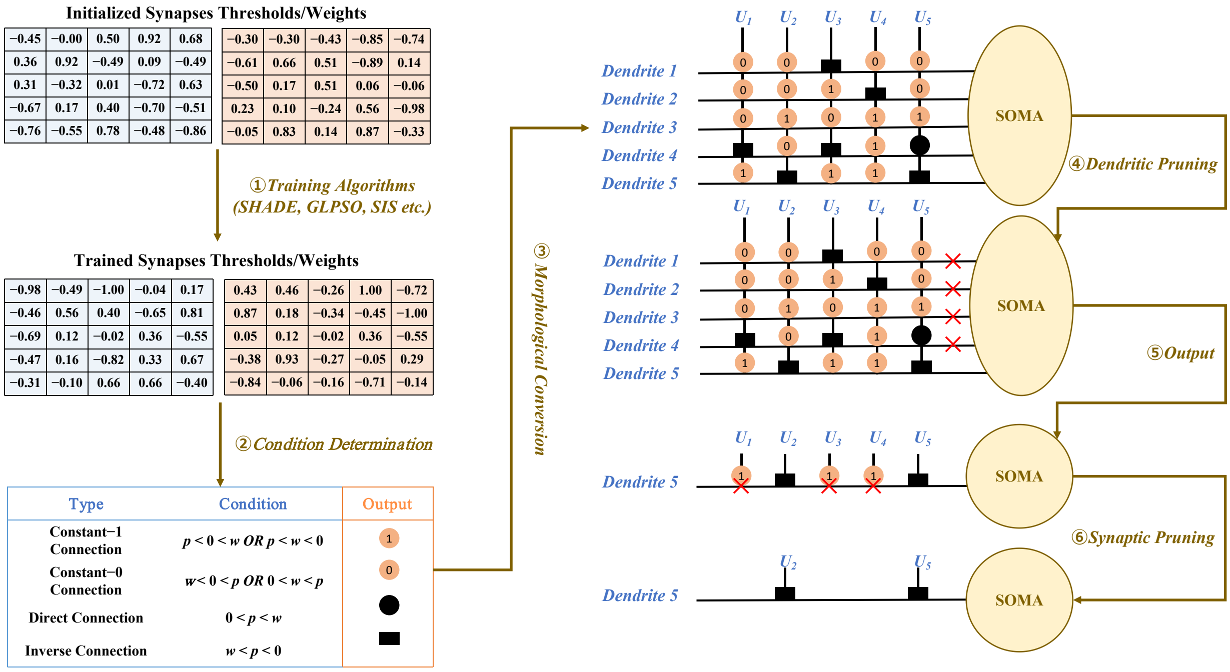


Figure 2. Learning process of a DNM.

(1) A constant-1 connection: when (1) $p_{ij} < 0 < w_{ij}$ or (2) $p_{ij} < w_{ij} < 0$, indicates that, despite the input varying from 0 to 1, the postsynaptic cell's potential remains approximately 1.

(2) A constant-0 connection: when (1) $w_{ij} < 0 < p_{ij}$ or (2) $0 < w_{ij} < p_{ij}$, implying that, irrespective of when the input is shifted from 0 to 1, the resulting potential is always close to 0.

(3) An excitatory connection: when $0 < p_{ij} < w_{ij}$, regardless of whether the input goes from 0 to 1, the potential is proportional to the input signal.

(4) An inhibitory connection: when $w_{ij} < p_{ij} < 0$, regardless of whether the input goes from 0 to 1, the potential is inversely proportional to the input signal.

2.1.2. Dendritic Layer

All synaptic layer signals are received by M dendrites in the dendrite layer. In this layer, multiplication is employed. Multiplicative activity is the primary mechanism for visuomotor responses to motor input in insects [40], and the multiplicative operations performed by each dendrite are considered to be the simplest non-linear operations [41] that are commonly detected in neurons, such as the auditory spatial sensory region. The multiplication operator is equivalent to the logical AND operation once the dendrites' inputs are 1 or 0 (i.e., constant 1 or 0 connections). The jth output function of the dendritic branch can be printed as:

$$S_j = \prod_{i=1}^{n} Z_{ij}, \tag{2}$$

2.1.3. Membrane Layer

Following that, all signals from dendritic branches are captured in the membrane layer. A summation operator, similar to a logic OR operation, is incorporated in the membrane layer, and the summed signal is conveyed to the soma body to stimulate the neuron. Its formula is as follows:

$$L = \sum_{j=1}^{M} S_j, \tag{3}$$

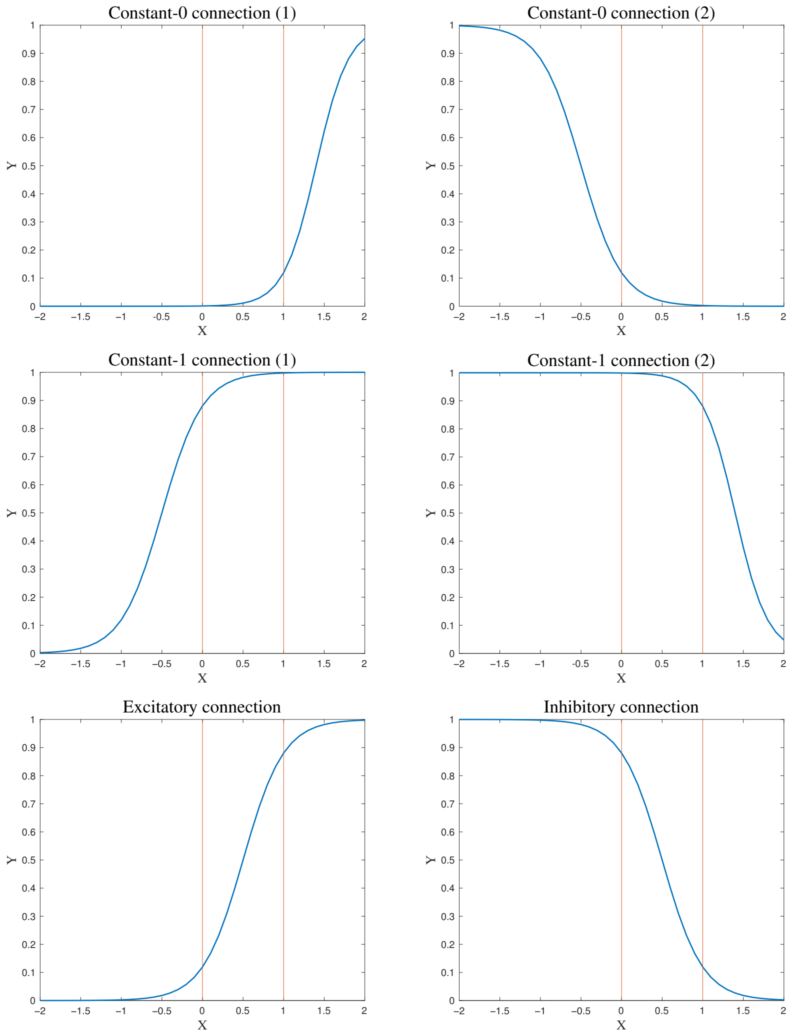


Figure 3. Connection cases of the synaptic layer.

2.1.4. Soma Layer

Ultimately, a sigmoid function is used in the soma body to deal with the somatic membrane potential L. Its purpose is to determine whether the neuron fires based on the overall model's output. This procedure is written as follows:

$$O = \frac{1}{1 + e^{-K(L-P)}} \tag{4}$$

where P is the firing parameter that varies from 0 to 1 and represents the threshold in the soma body.

Particularly, Figure 2 also reveals that the DNM has a similar structure to some other neuronal models, but it is a general model with distinct features for such a single neuron. As shown in step 4 of the figure, the DNM initially has a complete connection structure after the morphological transformation; however, some dendrites and synapses may be unnecessary for the execution of a specific task, and thus the DNM can clear some dendrites that are no longer required from the neuron by learning, which is called dendritic pruning. After undergoing dendritic pruning, dendrites 1–4 have constant-0 connections, and because any number multiplied by 0 equals 0, these dendrites can be ignored and removed from the neuron, leaving only dendrite 5.

In addition, regarding step 6, since they are constant-1 connections and the multiplication operation is performed on each dendrite, meaning that the outcome of any number multiplied by 1 does not shift, the three synaptic inputs (U_1, U_3, and U_4) on dendrite 5 are excluded by the neuron after learning. This is known as synaptic pruning of the DNM. Furthermore, synaptic pruning can be used to infer the status of neurons' local synaptic connectivity. Thus, given a neuron with organized dendrites, the DNM can, through its ability to detect neuron morphology, potentially infer by dendritic pruning and synaptic pruning the number of dendrites situated at the location where the presynaptic axon terminal is connected to the dendrites and the way they are connected, i.e., an excitatory or inhibitory connection.

2.2. Meta-Heuristic Algorithms

A typical MHA contains an initialization operation, a search operation, an evaluation operation, and a selection operation. In this section, four algorithms—SE, DE, JADE, and SHADE—are described in detail.

2.2.1. Initialization Operation

The initialization operation includes the assignment of teh initial parameters as well as the process of producing initial individuals in the solution space. Furthermore, N individuals are randomly initialized. When the DNM is trained using MHAs, the individuals of the algorithm are represented as:

$$X_i = \left[w_{1,1}, w_{1,2}, ..., w_{i,j}, ..., w_{n,M}, p_{1,1}, p_{1,2}, ..., p_{i,j}, ..., p_{n,M}\right] \tag{5}$$

The evaluation operation function $f(\cdot)$ evaluates each individual of the population. In addition, $f(\cdot)$ is the set of output equations as shown in Equations (1)–(4) and (10).

2.2.2. Search Operation

The search operation of SE can be represented as:

$$V_{i,j} = X_{r1,j} + \begin{cases} F \cdot \|X_{r2,*} - X_{r3,*}\|_2 \cdot \prod_{k=j}^{D_c-1} \sin(\theta_j), & \text{if } j = 1 \\ F \cdot \|X_{r2,*} - X_{r3,*}\|_2 \cdot \cos(\theta_{j-1}) \cdot \prod_{k=j}^{D_c-1} \sin(\theta_j), & \text{if } 1 < j \le D_c - 1 \\ F \cdot \|X_{r2,*} - X_{r3,*}\|_2 \cdot \cos(\theta_{j-1}), & \text{if } j = D_c \end{cases} \tag{6}$$

where V_i is a temporary offspring individual. The scale factor F stands for a variable that can be used to adjust the search radius. $(r1, r2, r3)$ are the indices of the individual, and they are random integers between 1 and N. The sphere's radius, calculated using the Euclidean norm in high dimensions, is indicated by the symbol $\|X_{r2,*} - X_{r3,*}\|_2$. A uniformly distributed random number between $[0, 2\pi]$ produces the value of θ. D_c is the dimension of the original D after the dimensionality reduction operation. $* = \{1, 2, ...D_c\}$ indicates the set of dimensions after dimensionality reduction.

The search operation of DE can be represented as:

$$V_{i,j} = X_{r1,j} + \begin{cases} F \cdot (X_{r2,j} - X_{r3,j}), & \text{if } b < CR \text{ or } j = j_r \\ 0, & \text{if } otherwise \end{cases} \tag{7}$$

F is the scaling factor. j_r is a random integer between 1 and D. b is a random real number between 0 and 1. The crossover probability, denoted by CR, determines the extent to which the mutant individual displaces parts of the parent individual.

The search operation of JADE and SHADE can be represented as:

$$V_{i,j} = X_{i,j} + \begin{cases} F \cdot (X_{o,j} - X_{i,j} + X_{r1,j} - Y_{r,j}), & \text{if } b < CR \text{ or } j = j_r \\ 0, & \text{if } otherwise \end{cases} \tag{8}$$

where F and CR are adaptive parameters. $X_{o,j}$ is the top-ranked individual in the parent population. X_{r1} is a random individual of the parent population. Y_r is a random individual selected from the set of the current and previous generations of populations.

2.2.3. Evaluation Operation

Once the search operation is executed, the fitness values of the total individuals in V are evaluated. With a greedy selection strategy, V_i is compared to the original population X_i, with individuals who outperform remaining in the next generation. The selection operation is formulated as follows:

$$X_i' = \begin{cases} V_i, & \text{if } f(V_i) < f(X_i) \\ X_i, & \text{if } otherwise \end{cases} \tag{9}$$

This optimization case uses the minimum value approach. The algorithm moves on to the subsequent iteration of the search operation after finishing the selection operation. Taking DE as an example, we show the complete execution flow of MHAs using Algorithm 1.

Algorithm 1: Pseudocode of DE.

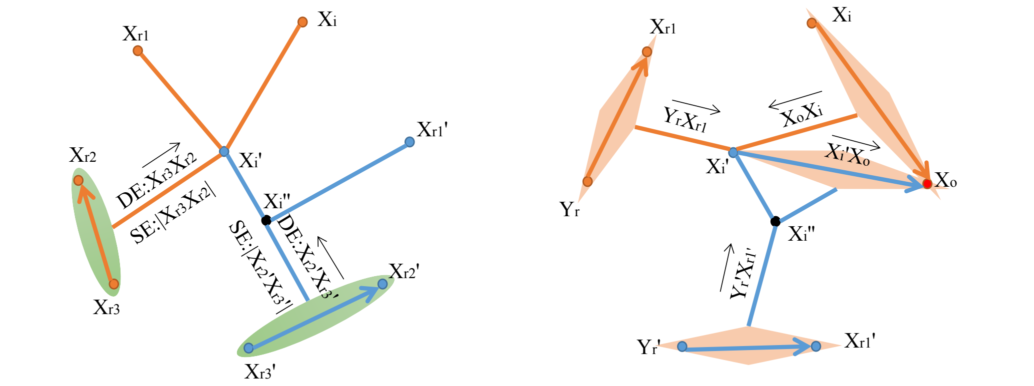
begin
  /*Initialization */
  Initialize parameter and randomly initialize N individuals
  $f(X) = evaluate(X)$
  while *Terminal Condition* **do**
    /*Search Operation */
    for $i = 1 : N$ **do**
      $V_{i,j} = X_{r1,j} + \begin{cases} F \cdot (X_{r2,j} - X_{r3,j}), & \text{if } b < CR \text{ } or \text{ } j = j_r \\ 0, & \text{if } otherwise \end{cases}$
    /*Selection Operation */
    for $i = 1 : N$ **do**
      $X_i' = \begin{cases} V_i, & \text{if } f(V_i) < f(X_i) \\ X_i, & \text{if } otherwise \end{cases}$

2.2.4. Analysis of MHAs Using Population Interaction Networks

Existing nodes and edges, which represent individuals and connections between individuals, respectively, are used to construct complex networks. Comparing and analyzing network properties requires taking each network's structure into account. In general, the topology of the network depicts the final product of population evolution that is ongoing and clearly affects the algorithm performance [42,43]. The knowledge that each individual carries is conveyed by a particular evolutionary mechanism, and the interaction of individuals is achieved. The network's final graph demonstrates the results of certain laws and the dissemination of knowledge to the entire population. The PIN describes how individuals interact with one another in MHAs.

Our previous results showed that DE had a Poisson-distributed PIN, while JADE and SHADE had power-law-distributed PINs [32]. In this study, the analysis of SE is added (as shown in Appendix A), and the results showed that SE had a Poisson-distributed PIN. The statistical and fitting methods for the degrees are consistent with [32]. Figure 4 shows the PINs of MHAs. In Figure 4, (1), X_i'' is an offspring of X_i'. The yellow lines are the network connections of X_i' to its parents X_{r1}, X_{r2}, X_{r3}, and X_i during the update process. The blue lines represent the network connections of X_i'' to its parents X'_{r1}, X'_{r2}, X'_{r3}, and X_i' during the update process.

At one iteration, the newly generated individuals X_i' and X'_{r1}, X'_{r2}, X'_{r3}, and X_i are related. The node X_i' and the four edges (degrees) associated with it are then recorded in the PIN. In Figure 4 (2), X_{r2} and X_{r3} are replaced by X_o and Y_r. Figure 4 (1) and Figure 4 (2) are the PINs of SE and DE, and two variants of DE, respectively. The search operations Equations (6) and (7) of SE and DE are represented by the green ellipse, and the search operation Equation (8) of two variants of DE is represented by the orange rhombus. The distinctions between SE, DE, JADE, and SHADE are discovered through the comparison of three search operations.



(1) The population interaction network of SE and DE. (2) The population interaction network of JADE and SHADE.

Figure 4. The population interaction networks of MHAs.

In Equation (8), X_o is the current best point involved in each X_i' update process, which interacts with every point in the population and, therefore, has a large number of links. This is consistent with the characteristics of a power-law distribution. When X_o is not updated, the nodes connected to X_o will grow with the numbers of iterations of the algorithm, and this will make JADE and SHADE more biased towards a power-law distribution. Relatively, there is no central node of X_o in the iterative process of SE and DE, which implies that the inter-individual information interaction between SE and DE presents an ergodic and random nature. This makes the PINs of DE and SE exhibit a Poisson distribution.

3. Results

3.1. Dataset Description

In order to evaluate the performance of the DNM trained by MHAs, we experimentally selected five-year intercepts (the Nikkei Stock Average is a four-year intercept) of the daily closing prices of five stock indexes from the world's top two typical stock markets—the United States and Japan—in terms of the share of total world stock market value. The dataset can be obtained from the sources listed below Table 1. The five stock price index datasets are described in detail in Table 1.

Table 1. Experimental datasets.

Dataset			Data Period	Amount of Data
Market Indices [1]	**Region**	**Abbreviation**	**D/M/Y**	**Days**
Dow Jones Industrial Average	USA	DJIA	11/02/2018–13/11/2022	1169
NASDAQ Composite Index	USA	COMP	12/02/2018–13/11/2022	1168
NASDAQ 100 Index	USA	NDX	25/01/2018–26/10/2022	1169
Nikkei Stock Average (Nikkei225)	Japan	N225	18/02/2019–14/11/2022	882
Standard & Poor's 500	USA	SPX	14/12/2017–13/09/2022	1169

[1] Sources of market indices. Dow Jones Industrial Average: https://us.spindices.com (accessed on 14 November 2022). NASDAQ Composite Index: https://www.nasdaq.com (accessed on 14 November 2022). NASDAQ 100 Index: https://www.nasdaq.com (accessed on 27 October 2022). Nikkei Stock Average(NIKKEI225): https://indexes.nikkei.co.jp/en/nkave/index/profile?idx=nk225 (accessed on 15 November 2022). Standard & Poor's 500: https://www.spglobal.com/spdji/en/ (accessed on 14 September 2022).

3.2. Evaluation Criteria

In order to make a comprehensive analysis and comparison, three commonly utilized metrics were employed to assess prediction model performances: the mean square error

(MSE), mean absolute percentage error (MAPE), and mean absolute error (MAE), with smaller values of MSE, MAPE, and MAE indicating better prediction model performances. The calculations are represented by the equations below:

$$f(X_i) = \begin{cases} MSE(X_i) = \frac{1}{n}\sum_{h=1}^{n}(T_h - O_h)^2 \\ MAPE(X_i) = \frac{1}{n}\sum_{h=1}^{n}\left|\frac{T_h - O_h}{T_h}\right| \\ MAE(X_i) = \frac{1}{n}\sum_{h=1}^{n}|T_h - O_h| \end{cases} \tag{10}$$

The fitness function of an individual X_i between the outcome of the DNM and the target output was computed using MSE, MAPE, and MAE. n refers to the total number of samples. The hth target output and the actual outcome of the DNM are given by T_h and O_h, respectively. For the predicting phrases, the correlation coefficient R is determined as follows:

$$R = \frac{\sum_{h=1}^{n}(T_h - \overline{T}_h)(O_h - \overline{O}_h)}{\sqrt{\sum_{h=1}^{n}(T_h - \overline{T}_h)^2 \sum_{h=1}^{n}(O_h - \overline{O}_h)^2}} \tag{11}$$

where the value of R increases with the accuracy of the forecast.

Moreover, to determine if the prediction models in this work differ significantly from one another, the Wilcoxon rank-sum test and the Friedman test were used. The Wilcoxon rank sum test is a non-parametric assessment of the null hypothesis versus the competing hypothesis for two samples from the same group. In addition, in order to determine the differences between SHADE and other algorithms, the Wilcoxon rank-sum test was employed to compare the $f(\cdot)$ obtained after 30 repetitions of DNM training, and $W/T/L$ was used to represent these differences.

The number of functions for which SHADE was considerably superior to other algorithms is represented by W, T denotes the number of functions for which SHADE performed no better than other algorithms, and L denotes the number of functions for which SHADE performed significantly inferior to other algorithms. The alternative hypothesis in the Friedman test is that the median values of two or more algorithms vary, while the null hypothesis is that the median values of the various algorithms are equal. In this study, a lower value of *Rank* shows that the model performed better on the FTSF problem.

3.3. Experimental Setup

In the experiment part, we executed two different sets of experiments. The first set of experiments is the performance comparison of DNMs trained by MHAs on the FTSF problem. The MHAs compared in the first set of experiments are SHADE and JADE with power-law-distributed PINs, DE and SE with Poisson-distributed PINs, and GLPSO and SIS, which are swarm-intelligence algorithms.

The purpose of the first set of experiments is to demonstrate that DNMs trained by MHAs with power-law-distributed PINs outperformed DNMs trained by other MHAs on FTSF. The second set of experiments is a comparison of the performance of various mainstream prediction models on the FTSF problem. In the second set of experiments, DNM-SHADE, DNM*, DNM-BP, ANFIS, LSTM, the Elman network, and MLP are compared. The purpose of the second set of experiments is to demonstrate that DNMs trained with MHAs with power-law-distributed PINs perform competitively on FTSF compared to mainstream prediction models.

Furthermore, each dataset was divided into two sets: a training set that includes 70% samples and a testing set that includes the remaining 30% samples. The experiments for each prediction model were individually run 30 times on each dataset. The overall performance of each model in terms of accuracy in prediction was evaluated using the average and standard deviation of the evaluation metrics. Additionally, the raw data needed to undergo a normalization step in order to lower the sophistication of the data before applying the particular model. This step was first performed on the training samples, then on the test samples with the same parameters, and finally on the prediction model output using an inverse normalization operation and in a range of values of $[0,1]$ by the given equation:

$$x_i^{(n)} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (12)$$

where the maximum and minimum values of the data are denoted by x_{max} and x_{min}.

3.4. Parameter Settings

In order to obtain robust and dependable performance, it is essential to identify the ideal parameter settings for each algorithm. Thus, the training of a DNM requires tuning of K, P, and M using Genichi Taguchi's orthogonal experiment approach [44] as given in Table 2. Group 10 was found to be the optimal combination of parameters for a DNM on the FTSF problem. The parameter settings for the prediction models on the five stock price index datasets are shown in Table 3.

Table 2. Orthogonal experimental parameters.

Parameter	K	P	M	Parameter	K	P	M
group 1	1	0.1	2	group 14	10	0.7	20
group 2	1	0.3	10	group 15	10	1	2
group 3	1	0.5	20	group 16	15	0.1	10
group 4	1	0.7	5	group 17	15	0.3	20
group 5	1	1	15	group 18	15	0.5	5
group 6	5	0.1	20	group 19	15	0.7	15
group 7	5	0.3	5	group 20	15	1	2
group 8	5	0.5	15	group 21	20	0.1	5
group 9	5	0.7	2	group 22	20	0.3	15
group 10	**5**	**1**	**10**	group 23	20	0.5	2
group 11	10	0.1	15	group 24	20	0.7	10
group 12	10	0.3	2	group 25	20	1	20
group 13	10	0.5	10				

3.5. Performance Comparison of MHA-Trained DNMs

In this part, we compare the performance of DNMs trained by SHADE, JADE, SIS, GLPSO, SE, and DE on the FTSF problem. Table 4 shows the prediction performance of DNMs trained by each MHA on five stock indices. In addition, the DNM optimized by BP is also compared in the table. As can be seen from the $W/T/L$ in the table, except for DNM-BP, the performance of DNM-SHADE was far better than its competitors. It should be noted that the test results of the Wilcoxon rank-sum test show that there was no distinction between DNM-SHADE and DNM-BP, and the values of W and L were both 6.

This is due to the fact that DNM-BP is extremely susceptible to fall into local optima; thus, it is not very robust, and the average value of DNM-SHADE after multiple runs has a great advantage compared to DNM-BP, which is also reflected in the Friedman test. Furthermore, according to the rank obtained from the Friedman test, it can be found that SHADE and JADE with power-law-distributed PINs outperform the swarm-intelligence algorithms (SIS and GLPSO) and SE and DE with Poisson-distributed PINs. This result demonstrates that MHAs with power-law-distributed PINs may be more advantageous in optimizing the parameters of DNMs to predict financial time series.

Table 3. Parameter settings of the prediction models on the five stock price index datasets.

Models	Parameter	Value
ANFIS	Iteration number	100
	Initial step size	0.01
	Step size decrease rate	0.9
	Step size increase rate	1.1
Elman	Iteration number	1000
	Learning rate	0.01
	Hidden layer number	15
MLP	Iteration number	1000
	Learning rate	0.1
	Hidden layer number	15
LSTM	Iteration number	1000
	Learning rate	0.1
	Learning rate drop factor	0.1
	Hidden units	15
DNM-BP	Iteration number	1000
	Learning rate	0.1
	Dendrite number	10
DNM-DE	Iteration number	1000
	Dendrite number	10
	Scale factor	0.5
	Crossover rate	0.9
DNM-JADE	Iteration number	1000
	Dendrite number	10
	Scale factor	Adaptive value
	Crossover rate	Adaptive value
	Memory size	100
DNM-SHADE	Iteration number	1000
	Dendrite number	10
	Scale factor	Adaptive value
	Crossover rate	Adaptive value
	Memory size	100
DNM-SE	Iteration number	1000
	Dendrite number	10
	Scale factor	Adaptive value
	Number of variable dimensions	[5, 10]
DNM-GLPSO	Iteration number	1000
	Dendrite number	10
	Mutation probabilty	0.01
	Stopping gap	7
	Inertia weight	0.7298
	Accelerate coefficient	1.49618
DNM-SIS	Iteration number	1000
	Dendrite number	10
	Scale factor	Adaptive value
DNM*	Iteration number	1000
	Learning rate	0.05
	Dendrite number	1

Table 4. Performance of DNMs trained by MHAs on the FTSF problem.

		DNM-SHADE		DNM-JADE			DNM-SIS			DNM-GLPSO		
	metrics	**mean**	**std.**	**mean**	**std.**		**mean**	**std.**		**mean**	**std.**	
DJIA	MSE	9.652×10^{4}	6.545×10^{4}	2.444×10^{5}	2.054×10^{5}	+	1.555×10^{6}	2.062×10^{6}	+	2.241×10^{5}	4.183×10^{5}	=
	MAPE	2.353×10^{-5}	1.485×10^{-5}	3.552×10^{-5}	2.488×10^{-5}	+	7.937×10^{-5}	8.986×10^{-5}	=	2.596×10^{-5}	3.031×10^{-5}	=
	MAE	2.526×10^{2}	1.102×10^{2}	3.905×10^{2}	1.980×10^{2}	+	8.543×10^{2}	8.076×10^{2}	+	3.234×10^{2}	2.832×10^{2}	=
COMP	MSE	5.607×10^{4}	2.917×10^{4}	1.100×10^{5}	1.256×10^{5}	+	1.511×10^{5}	1.836×10^{5}	+	1.989×10^{5}	4.130×10^{5}	=
	MAPE	4.449×10^{-5}	3.349×10^{-5}	6.316×10^{-5}	4.429×10^{-5}	=	8.464×10^{-5}	5.849×10^{-5}	+	6.462×10^{-5}	7.614×10^{-5}	=
	MAE	1.777×10^{2}	5.778×10^{1}	2.400×10^{2}	1.079×10^{2}	+	2.638×10^{2}	1.542×10^{2}	+	2.311×10^{2}	2.169×10^{2}	=
NDX	MSE	2.397×10^{5}	1.523×10^{5}	3.762×10^{5}	3.657×10^{5}	=	2.129×10^{5}	7.895×10^{4}	=	8.117×10^{5}	1.179×10^{6}	+
	MAPE	1.457×10^{-4}	4.932×10^{-5}	1.599×10^{-4}	9.360×10^{-5}	=	1.161×10^{-4}	6.475×10^{-5}	=	2.345×10^{-4}	1.673×10^{-4}	+
	MAE	3.885×10^{2}	1.172×10^{2}	4.560×10^{2}	2.074×10^{2}	=	3.812×10^{2}	7.470×10^{1}	=	6.155×10^{2}	4.292×10^{2}	+
N225	MSE	4.005×10^{5}	3.454×10^{5}	1.169×10^{6}	1.227×10^{6}	+	1.061×10^{6}	9.946×10^{5}	+	7.384×10^{5}	8.487×10^{5}	+
	MAPE	2.217×10^{-5}	1.123×10^{-5}	4.107×10^{-5}	3.024×10^{-5}	+	5.183×10^{-5}	4.393×10^{-5}	+	2.157×10^{-5}	1.588×10^{-5}	=
	MAE	3.579×10^{2}	1.415×10^{2}	5.995×10^{2}	2.814×10^{2}	+	6.783×10^{2}	4.170×10^{2}	+	4.607×10^{2}	2.197×10^{2}	+
SPX	MSE	1.237×10^{5}	8.399×10^{3}	1.327×10^{5}	3.560×10^{4}	=	1.245×10^{5}	7.521×10^{3}	=	1.734×10^{5}	1.841×10^{5}	-
	MAPE	1.977×10^{-4}	1.073×10^{-5}	2.006×10^{-4}	2.365×10^{-5}	=	1.966×10^{-4}	1.197×10^{-5}	=	2.138×10^{-4}	8.140×10^{-5}	-
	MAE	2.950×10^{2}	1.473×10^{1}	3.014×10^{2}	3.439×10^{1}	=	2.974×10^{2}	1.400×10^{1}	=	3.172×10^{2}	1.200×10^{2}	-
W/T/L		-/-/-		8/7/0			8/7/0			5/7/3		
Rank		1.5333		3.4667			3.6			3.7333		
		DNM-SHADE		**DNM-SE**			**DNM-BP**			**DNM-DE**		
	metrics	**mean**	**std.**	**mean**	**std.**		**mean**	**std.**		**mean**	**std.**	
DJIA	MSE	9.652×10^{4}	6.545×10^{4}	2.988×10^{5}	2.575×10^{5}	+	4.430×10^{6}	2.378×10^{7}	-	1.660×10^{6}	1.340×10^{6}	+
	MAPE	2.353×10^{-5}	1.485×10^{-5}	3.998×10^{-5}	2.253×10^{-5}	+	5.203×10^{-5}	2.295×10^{-4}	-	1.133×10^{-4}	5.961×10^{-5}	+
	MAE	2.526×10^{2}	1.102×10^{2}	4.360×10^{2}	2.016×10^{2}	+	4.760×10^{2}	2.042×10^{3}	-	1.095×10^{3}	5.213×10^{2}	+
COMP	MSE	5.607×10^{4}	2.917×10^{4}	1.930×10^{5}	3.022×10^{5}	+	9.697×10^{6}	2.457×10^{7}	+	2.446×10^{5}	1.696×10^{5}	+
	MAPE	4.449×10^{-5}	3.349×10^{-5}	8.511×10^{-5}	6.713×10^{-5}	+	4.766×10^{-4}	1.065×10^{-3}	+	1.443×10^{-4}	6.589×10^{-5}	+
	MAE	1.777×10^{2}	5.778×10^{1}	2.966×10^{2}	1.710×10^{2}	+	1.284×10^{3}	2.829×10^{3}	+	4.150×10^{2}	1.545×10^{2}	+
NDX	MSE	2.397×10^{5}	1.523×10^{5}	9.291×10^{5}	1.250×10^{6}	+	1.211×10^{7}	3.032×10^{7}	+	6.194×10^{5}	1.716×10^{6}	-
	MAPE	1.457×10^{-4}	4.932×10^{-5}	2.207×10^{-4}	1.461×10^{-4}	+	6.326×10^{-4}	1.230×10^{-3}	+	9.398×10^{-5}	1.904×10^{-4}	-
	MAE	3.885×10^{2}	1.172×10^{2}	6.204×10^{2}	3.458×10^{2}	+	1.597×10^{3}	3.079×10^{3}	+	4.396×10^{2}	4.446×10^{2}	-
N225	MSE	4.005×10^{5}	3.454×10^{5}	7.702×10^{5}	6.366×10^{5}	+	2.944×10^{6}	1.469×10^{7}	-	1.997×10^{6}	1.631×10^{6}	+
	MAPE	2.217×10^{-5}	1.123×10^{-5}	4.523×10^{-5}	2.898×10^{-5}	+	5.664×10^{-5}	2.735×10^{-4}	-	9.355×10^{-5}	4.598×10^{-5}	+
	MAE	3.579×10^{2}	1.415×10^{2}	5.499×10^{2}	2.270×10^{2}	+	5.393×10^{2}	1.558×10^{3}	-	1.045×10^{3}	3.584×10^{2}	+
SPX	MSE	1.237×10^{5}	8.399×10^{3}	1.563×10^{5}	8.022×10^{4}	=	1.235×10^{5}	4.593×10^{3}	=	1.529×10^{5}	8.842×10^{4}	=
	MAPE	1.977×10^{-4}	1.073×10^{-5}	2.119×10^{-4}	3.964×10^{-5}	=	1.988×10^{-4}	7.227×10^{-6}	=	2.132×10^{-4}	6.263×10^{-5}	=
	MAE	2.950×10^{2}	1.473×10^{1}	3.175×10^{2}	5.862×10^{1}	=	2.947×10^{2}	7.439×10^{0}	=	3.287×10^{2}	8.910×10^{1}	+
W/T/L		-/-/-		12/3/0			6/3/6			10/2/3		
Rank		1.5333		4.7333			5.3333			5.6		

Figure 5 is the MAE box plots of six MHAs and BP-trained DNMs on DJIA, COMP, NDX, N225, and SPX. From the figure, it can be seen that the SHADE-trained DNMs have lower means and higher robustness in most cases compared to other MHAs. In addition, the DNM trained by BP also possesses a low mean value. However, the red "+" symbol in the figure illustrates that BP possesses the largest extreme values in most cases compared to MHAs. This indicates that MHAs have a strong exploration ability to jump out of a local optimum, and their trained DNMs have higher robustness compared to the BP-trained DNMs.

Figure 6 shows the predicted outcomes of MHA-trained DNMs for the dataset DJIA. The sequences predicted by the primary seven models are directly compared with the actual values on the training and test sets in the top half of the figure. We also subtracted the real values from the predicted results. As a result, the polylines in the lower half of the figure, which represent the deviations fluctuating around the straight line, were generated. From Figure 6, it can be seen that the output values predicted by DNM-SHADE are closer to the actual values of the training and test sets of the dataset and have smaller deviation values.

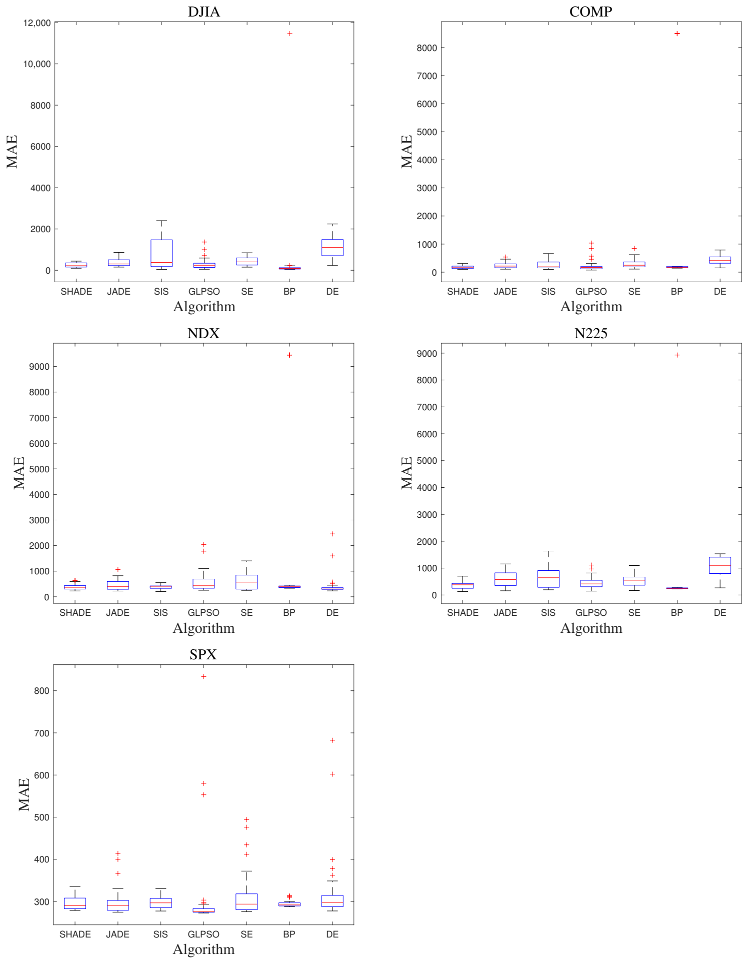


Figure 5. MAE plots of six MHAs and BP-trained DNMs on DJIA, COMP, NDX, N225, and SPX.

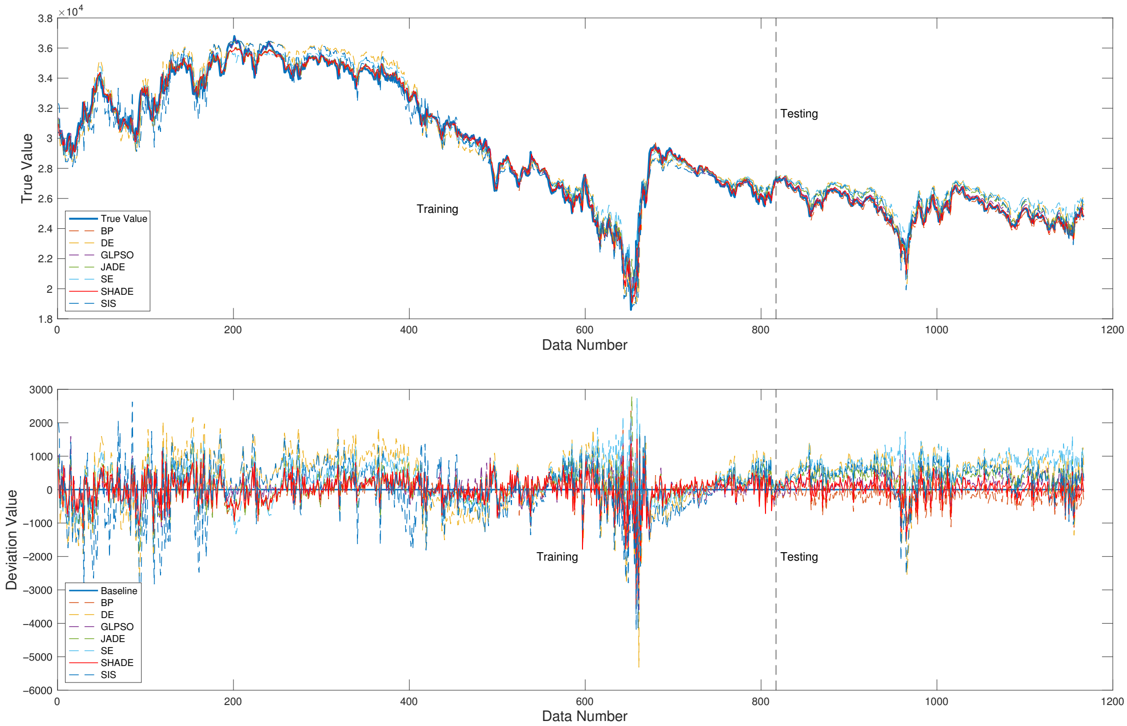


Figure 6. Prediction and deviation plots of six MHAs and BP-trained DNMs on DJIA.

3.6. Performance Comparison of Mainstream Time-Series-Forecasting Models

In this section, we test the performance of DNM-SHADE, DNM*, DNM-BP, LSTM, Elman, ANFIS, and MLP on the FTSF problem. Table 5 shows the performance and ranking of the above models. As can be seen from the table, the performance of DNM-SHADE is competitive among the mainstream models. It should be emphasized that the performance of the DNM family of models is not the strongest, due to the fact that a DNM itself is a single neuron and still has limitations compared to the ANN family of models.

Therefore, the focus of this study was to select suitable MHAs for DNM optimization. Consequently, we used the simplest DNM in our study and did not attempt to compose DNMs into neural networks or perform data-preprocessing operations. The experimental results prove that the simplest DNM optimized by SHADE was able to beat MLP and is competitive with mainstream models, such as ANFIS, LSTM, and Elman. This result fully demonstrates the development potential of DNMs optimized by MHAs.

Figure 7 depicts the correlation coefficient curves for the first four models on the dataset DJIA, from which we can see that DNM-SHADE has an advantage over LSTM. Figure 8 shows the prediction and deviation plots of seven types of time-series-forecasting models on DJIA.

Table 5. Performance of various types of time-series-forecasting models.

		ANFIS		LSTM		Elman		DNM-SHADE	
	metrics	**mean**	**std.**	**mean**	**std.**	**mean**	**std.**	**mean**	**std.**
DJIA	MSE	5.594×10^{4}	0	1.684×10^{5}	1.191×10^{5}	8.463×10^{4}	1.001×10^{5}	9.652×10^{4}	6.545×10^{4}
	MAPE	5.022×10^{-6}	0	4.720×10^{-6}	5.631×10^{-6}	1.563×10^{-5}	1.426×10^{-5}	2.353×10^{-5}	1.485×10^{-5}
	MAE	1.720×10^{2}	0	2.497×10^{2}	6.651×10^{1}	2.015×10^{2}	1.147×10^{2}	2.526×10^{2}	1.102×10^{2}
COMP	MSE	8.674×10^{3}	0	5.045×10^{4}	8.317×10^{4}	5.737×10^{4}	6.404×10^{4}	5.607×10^{4}	2.917×10^{4}
	MAPE	1.301×10^{-5}	0	4.274×10^{-5}	4.502×10^{-5}	7.155×10^{-5}	4.251×10^{-5}	4.449×10^{-5}	3.349×10^{-5}
	MAE	6.901×10^{1}	0	1.484×10^{2}	1.117×10^{2}	1.952×10^{2}	1.118×10^{2}	1.777×10^{2}	5.778×10^{1}
NDX	MSE	8.178×10^{3}	0	5.759×10^{4}	1.562×10^{5}	8.285×10^{4}	1.132×10^{5}	2.397×10^{5}	1.523×10^{5}
	MAPE	3.822×10^{-6}	0	5.271×10^{-5}	4.857×10^{-5}	8.634×10^{-5}	6.299×10^{-5}	1.457×10^{-4}	4.932×10^{-5}
	MAE	6.590×10^{1}	0	1.551×10^{2}	1.218×10^{2}	2.245×10^{2}	1.548×10^{2}	3.885×10^{2}	1.172×10^{2}
N225	MSE	8.091×10^{4}	0	2.463×10^{5}	7.608×10^{4}	1.288×10^{5}	1.369×10^{5}	4.005×10^{5}	3.454×10^{5}
	MAPE	2.588×10^{-6}	0	1.574×10^{-5}	6.609×10^{-6}	2.031×10^{-5}	1.617×10^{-5}	2.217×10^{-5}	1.123×10^{-5}
	MAE	1.870×10^{2}	0	3.008×10^{2}	4.763×10^{1}	2.192×10^{2}	1.082×10^{2}	3.579×10^{2}	1.415×10^{2}
SPX	MSE	2.520×10^{3}	0	2.249×10^{4}	1.143×10^{4}	3.566×10^{4}	4.237×10^{4}	1.237×10^{5}	8.399×10^{3}
	MAPE	7.542×10^{-6}	0	6.331×10^{-5}	2.461×10^{-5}	9.251×10^{-5}	6.732×10^{-5}	1.977×10^{-4}	1.073×10^{-5}
	MAE	3.878×10^{1}	0	1.142×10^{2}	2.754×10^{1}	1.399×10^{2}	9.736×10^{1}	2.950×10^{2}	1.473×10^{1}
Rank		1.0667		2.4667		3.0667		4.2667	

		DNM*		MLP		DNM-BP	
	metrics	**mean**	**std.**	**mean**	**std.**	**mean**	**std.**
DJIA	MSE	7.098×10^{4}	2.366×10^{4}	4.790×10^{5}	5.063×10^{5}	4.430×10^{6}	2.378×10^{7}
	MAPE	1.915×10^{-5}	4.776×10^{-6}	3.465×10^{-5}	2.669×10^{-5}	5.203×10^{-5}	2.295×10^{-4}
	MAE	2.221×10^{2}	2.829×10^{1}	5.199×10^{2}	2.568×10^{2}	4.760×10^{2}	2.042×10^{3}
COMP	MSE	2.330×10^{5}	7.877×10^{3}	2.510×10^{5}	2.802×10^{5}	9.697×10^{6}	2.457×10^{7}
	MAPE	1.569×10^{-4}	3.693×10^{-6}	7.617×10^{-5}	6.197×10^{-5}	4.766×10^{-4}	1.065×10^{-3}
	MAE	4.284×10^{2}	9.715E+00	3.194×10^{2}	1.814×10^{2}	1.284×10^{3}	2.829×10^{3}
NDX	MSE	5.203×10^{5}	9.770×10^{3}	8.762×10^{5}	1.329×10^{6}	1.211×10^{7}	3.032×10^{7}
	MAPE	2.645×10^{-4}	2.910×10^{-6}	2.174×10^{-4}	1.800×10^{-4}	6.326×10^{-4}	1.230×10^{-3}
	MAE	6.756×10^{2}	7.245E+00	5.950×10^{2}	4.597×10^{2}	1.597×10^{3}	3.079×10^{3}
N225	MSE	3.883×10^{5}	2.534×10^{4}	1.198×10^{6}	1.207×10^{6}	2.944×10^{6}	1.469×10^{7}
	MAPE	1.439×10^{-5}	3.392×10^{-6}	5.951×10^{-5}	4.202×10^{-5}	5.664×10^{-5}	2.735×10^{-4}
	MAE	3.509×10^{2}	2.108×10^{1}	7.471×10^{2}	3.087×10^{2}	5.393×10^{2}	1.558×10^{3}
SPX	MSE	1.473×10^{5}	6.209×10^{2}	4.226×10^{5}	4.979×10^{5}	1.235×10^{5}	4.593×10^{3}
	MAPE	2.284×10^{-4}	7.889×10^{-7}	3.252×10^{-4}	2.028×10^{-4}	1.988×10^{-4}	7.227×10^{-6}
	MAE	3.376×10^{2}	1.199×10^{0}	4.941×10^{2}	2.924×10^{2}	2.947×10^{2}	7.439×10^{0}
Rank		4.7333		6.1333		6.2667	

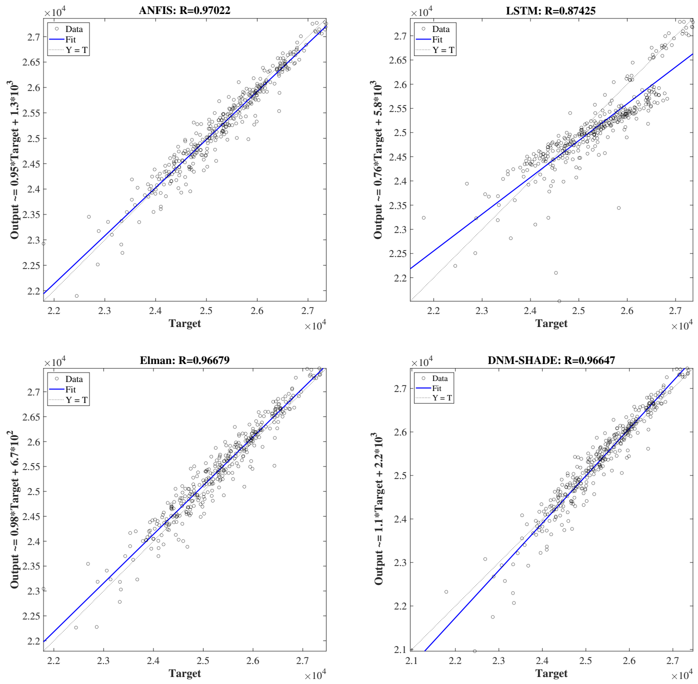


Figure 7. Correlation coefficient graphs of the top four models on DJIA.

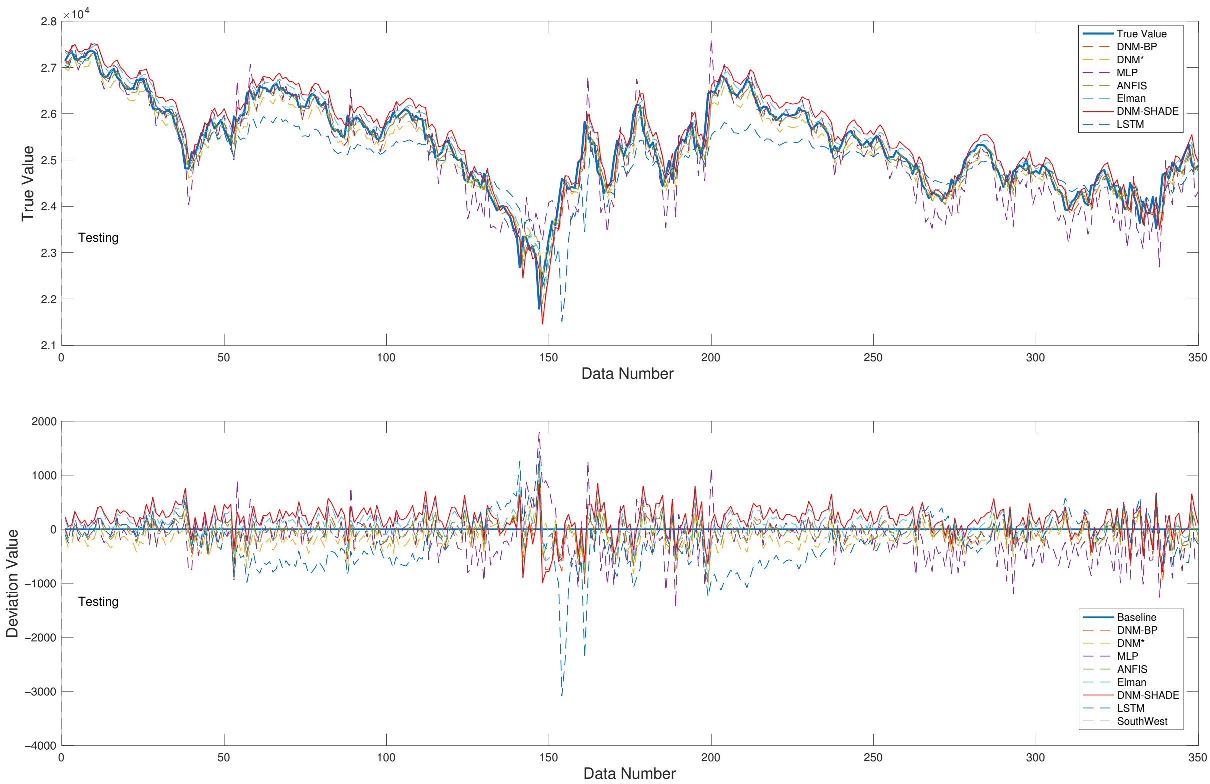


Figure 8. Prediction and deviation plots of seven types of time-series-forecasting models on DJIA.

4. Discussions and Conclusions

In this study, we focused on the optimization of DNMs. The DNM optimized by MHAs effectively avoided the problem of easily falling into local optima in predicting financial time-series problems relative to the BP-based DNM. Based on this, we further standardized the selection mechanism of MHAs to screen suitable MHAs for training DNMs based on

PIN theory. On the FTSF problem, theoretical analysis and experimental results showed that DNMs trained by MHAs with power-law-distributed PINs outperformed DNMs trained by MHAs with Poisson-distributed PINs. In addition, DNMs trained by MHAs with power-law-distributed PINs outperformed DNMs trained by swarm-intelligence algorithms on the FTSF problem.

These results open a new research perspective for optimizing neuronal models using MHAs. A large number of MHAs are divided into two categories: MHAs with power-law-distributed PINs and MHAs with Poisson-distributed PINs. Among the above two classes of MHAs, representative algorithms were selected, and their performance on the neuron model optimization problem was compared to determine which class of MHAs is more suitable for solving this problem. As a result, algorithms that are unsuitable for this problem were excluded from the MHA-selection process. This PIN-based screening method for MHAs greatly reduces the trial-and-error cost for researchers compared to empirical or even randomized screening methods for MHAs.

However, there are still shortcomings in this study. One is that PIN theory is not yet complete, and there are still many MHAs with network structures other than Poisson and power-law distributions as well as a large number of MHAs that have not been analyzed by PIN theory (e.g., group-intelligence algorithms). We will continue to refine our research on PIN theory in the future. Second, DNMs, as single neuron models, still have limitations, and the performance on the FTSF problem still falls short of mainstream neural network models, such as LSTM and ANFIS.

However, it is worth emphasizing that DNMs, as single-neuron models, already outperform certain neural network models, such as MLP, which indicates great potential for development. Indeed, enhancements to biological neurons have been much discussed; for example, the synaptic integration of dendritic branches can be conceptualized as pattern matching from a set of spatiotemporal templates, providing a unified characterization of the computational complexity of a single neuron and suggesting that a biological neuron can be equivalent to a deep artificial neural network [45].

Based on the properties of dendritic spikes, the dendrites of a single neuron may be able to generate active spikes to enhance its information-processing capabilities [46,47], implying that DNMs have much room for future progress as well. In the future, we will work on the further development of DNMs and eventually build more efficient neural network models based on DNMs to handle classification and prediction tasks.

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

Funding: This research received no external funding.

Data Availability Statement: Data will be made available upon request by contacting the corresponding author's email address.

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

Abbreviations

The following abbreviations are used in this manuscript:

DNM	Dendritic neuron model
MHA	Meta-heuristic algorithm
PIN	Population interaction network
FTSF	Financial time-series forecasting
ARIMA	Autoregressive integrated moving average
ANN	Artificial neural network
RNN	Recurrent neural network

MLP	Multi-layer perceptron
LSTM	Long short-term memory
ANFIS	Adaptive-network-based fuzzy inference system
BP	Backpropagation
DE	Differential evolution
JADE	Adaptive differential evolution with optional external archive
SHADE	Success-history based parameter adaptation for Differential Evolution
GLPSO	Genetic learning particle swarm optimization
SIS	Spatial information sampling algorithm
MSE	Mean square error
MAPE	Mean absolute percentage error
MAE	Mean absolute error

Appendix A

In SE, each individual can be thought of as a node, and the update between individuals denotes the generation of degrees. The degree is the sum of the connected edges between each individual and other individuals. It is also the number of edges. Utilizing the PIN, we can acquire the intrinsic communication of knowledge as well as the characteristics of the network formed by the populations. The cumulative distribution function of degrees of nodes obtained from the PIN on thirty IEEE CEC2017 benchmark functions [48] was fitted by the Poisson and power-law models, and each function was run 30 times.

We calculated the difference in fitting between the original data and two models using SSE and R^2 in order to determine which model fit the cumulative distribution function more accurately. SSE is the sum of squared errors between the original data and the fitted data. A lower SSE value indicates that the fitted data is more similar to the original data. SSE works with the following formula:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{A1}$$

where n represents the maximum degrees of nodes. y_i and $\hat{y}_i$ are the original data and fitted data, respectively. R^2 determines whether the fitted data accurately represent the original data. When the value of R^2 is close to 1, the fitted data can better reflect the original data. R^2 can be shown as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{A2}$$

where $\bar{y}$ is the mean of the original data. With the help of these two statistical techniques, we calculated the mean of the obtained results of SE and further list the data in Table A1. Table A1 shows that the cumulative distribution function of the PIN in SE fits the Poisson model best.

Table A1. Fitting results of SE.

	Poisson		Power Law	
	SSE	R^2	SSE	R^2
mean	$\mathbf{7.09 \times 10^{-2}}$	$\mathbf{9.84 \times 10^{-1}}$	8.32×10^{-2}	8.39×10^{-1}
std	1.86×10^{-2}	3.43×10^{-3}	3.64×10^{-2}	9.82×10^{-2}

References

1. Kilian, L.; Taylor, M.P. Why is it so difficult to beat the random walk forecast of exchange rates? *J. Int. Econ.* **2003**, *60*, 85–107. [CrossRef]
2. Krajbich, I.; Camerer, C.; Ledyard, J.; Rangel, A. Using neural measures of economic value to solve the public goods free-rider problem. *Science* **2009**, *326*, 596–599. [CrossRef]
3. Kim, H.J.; Shin, K.S. A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets. *Appl. Soft Comput.* **2007**, *7*, 569–576. [CrossRef]
4. Box, G.E.; Jenkins, G.M.; Reinsel, G.C.; Ljung, G.M. *Time Series Analysis: Forecasting and Control*; John Wiley & Sons: New York, NY, USA, 2015.

5. Holt, C.C. Forecasting seasonals and trends by exponentially weighted moving averages. *Int. J. Forecast.* **2004**, *20*, 5–10. [CrossRef]
6. Winters, P.R. Forecasting sales by exponentially weighted moving averages. *Manag. Sci.* **1960**, *6*, 324–342. [CrossRef]
7. Sezer, O.B.; Gudelek, M.U.; Ozbayoglu, A.M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput.* **2020**, *90*, 106181. [CrossRef]
8. Qian, X.Y.; Gao, S. Financial series prediction: Comparison between precision of time series models and machine learning methods. *arXiv* **2017**, arXiv:1706.00948.
9. Tealab, A. Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Comput. Inform. J.* **2018**, *3*, 334–340. [CrossRef]
10. Zhang, G.P.; Qi, M. Neural network forecasting for seasonal and trend time series. *Eur. J. Oper. Res.* **2005**, *160*, 501–514. [CrossRef]
11. Chandra, R. Competition and Collaboration in Cooperative Coevolution of Elman Recurrent Neural Networks for Time-Series Prediction. *IEEE Trans. Neural Netw. Learn. Syst.* **2015**, *26*, 3123–3136. [CrossRef]
12. Pascanu, R.; Mikolov, T.; Bengio, Y. On the difficulty of training recurrent neural networks. In Proceedings of the International Conference on Machine Learning (PMLR), Atlanta, GA, USA, 17–19 June 2013; pp. 1310–1318.
13. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
14. Baek, Y.; Kim, H.Y. ModAugNet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Syst. Appl.* **2018**, *113*, 457–480. [CrossRef]
15. Reid, D.; Hussain, A.J.; Tawfik, H. Financial time series prediction using spiking neural networks. *PLoS ONE* **2014**, *9*, e103656. [CrossRef]
16. Karaboga, D.; Kaya, E. Adaptive network based fuzzy inference system (ANFIS) training approaches: A comprehensive survey. *Artif. Intell. Rev.* **2019**, *52*, 2263–2293. [CrossRef]
17. McCulloch, W.S.; Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
18. Stanley, K.O.; Clune, J.; Lehman, J.; Miikkulainen, R. Designing neural networks through neuroevolution. *Nat. Mach. Intell.* **2019**, *1*, 24–35. [CrossRef]
19. Townsend, J.; Chaton, T.; Monteiro, J.M. Extracting Relational Explanations From Deep Neural Networks: A Survey From a Neural-Symbolic Perspective. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 3456–3470. [CrossRef] [PubMed]
20. Ostojic, S.; Brunel, N. From spiking neuron models to linear-nonlinear models. *PLoS Comput. Biol.* **2011**, *7*, e1001056. [CrossRef] [PubMed]
21. Xu, Z.; Wang, Z.; Li, J.; Jin, T.; Meng, X.; Gao, S. Dendritic neuron model trained by information feedback-enhanced differential evolution algorithm for classification. *Knowl.-Based Syst.* **2021**, *233*, 107536. [CrossRef]
22. Todo, Y.; Tamura, H.; Yamashita, K.; Tang, Z. Unsupervised learnable neuron model with nonlinear interaction on dendrites. *Neural Netw.* **2014**, *60*, 96–103. [CrossRef]
23. He, H.; Gao, S.; Jin, T.; Sato, S.; Zhang, X. A seasonal-trend decomposition-based dendritic neuron model for financial time series prediction. *Appl. Soft Comput.* **2021**, *108*, 107488. [CrossRef]
24. Gao, S.; Zhou, M.; Wang, Y.; Cheng, J.; Yachi, H.; Wang, J. Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *30*, 601–614. [CrossRef]
25. Dauphin, Y.N.; Pascanu, R.; Gulcehre, C.; Cho, K.; Ganguli, S.; Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 1–9.
26. Todo, Y.; Tang, Z.; Todo, H.; Ji, J.; Yamashita, K. Neurons with multiplicative interactions of nonlinear synapses. *Int. J. Neural Syst.* **2019**, *29*, 1950012. [CrossRef] [PubMed]
27. Wang, Z.; Gao, S.; Wang, J.; Yang, H.; Todo, Y. A dendritic neuron model with adaptive synapses trained by differential evolution algorithm. *Comput. Intell. Neurosci.* **2020**, *2020*, 2710561. [CrossRef] [PubMed]
28. Dokeroglu, T.; Sevinc, E.; Kucukyilmaz, T.; Cosar, A. A survey on new generation metaheuristic algorithms. *Comput. Ind. Eng.* **2019**, *137*, 106040. [CrossRef]
29. Tang, Y.; Song, Z.; Zhu, Y.; Hou, M.; Tang, C.; Ji, J. Adopting a dendritic neural model for predicting stock price index movement. *Expert Syst. Appl.* **2022**, *205*, 117637. [CrossRef]
30. Yilmaz, A.; Yolcu, U. A robust training of dendritic neuron model neural network for time series prediction. *Neural Comput. Appl.* **2023**, *in press*. [CrossRef]
31. Hutson, M. Has artificial intelligence become alchemy? *Science* **2018**, *360*, 478. [CrossRef]
32. Li, X.; Li, J.; Yang, H.; Wang, Y.; Gao, S. Population interaction network in representative differential evolution algorithms: Power-law outperforms Poisson distribution. *Phys. A Stat. Mech. Appl.* **2022**, *603*, 127764. [CrossRef]
33. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
34. Zhang, J.; Sanderson, A.C. JADE: Adaptive Differential Evolution With Optional External Archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958. [CrossRef]
35. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for Differential Evolution. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 71–78.
36. Tang, D. Spherical evolution for solving continuous optimization problems. *Appl. Soft Comput.* **2019**, *81*, 105499. [CrossRef]

37. Gong, Y.J.; Li, J.J.; Zhou, Y.; Li, Y.; Chung, H.S.H.; Shi, Y.H.; Zhang, J. Genetic Learning Particle Swarm Optimization. *IEEE Trans. Cybern.* **2016**, *46*, 2277–2290. [CrossRef]
38. Yang, H.; Yu, Y.; Cheng, J.; Lei, Z.; Cai, Z.; Zhang, Z.; Gao, S. An intelligent metaphor-free spatial information sampling algorithm for balancing exploitation and exploration. *Knowl.-Based Syst.* **2022**, *250*, 109081. [CrossRef]
39. Poirazi, P.; Brannon, T.; Mel, B.W. Pyramidal neuron as two-layer neural network. *Neuron* **2003**, *37*, 989–999. [CrossRef]
40. Reichardt, W. Evaluation of optical motion information by movement detectors. *J. Comp. Physiol. A* **1987**, *161*, 533–547. [CrossRef]
41. Koch, C.; Poggio, T. Multiplying with synapses and neurons. In *Single Neuron Computation*; Elsevier: Amsterdam, The Netherlands, 1992; pp. 315–345.
42. Whitacre, J.M.; Sarker, R.A.; Pham, Q.T. The Self-Organization of Interaction Networks for Nature-Inspired Optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 220–230. [CrossRef]
43. Dorronsoro, B.; Bouvry, P. Improving Classical and Decentralized Differential Evolution With New Mutation Operator and Population Topologies. *IEEE Trans. Evol. Comput.* **2011**, *15*, 67–98. [CrossRef]
44. Altland, H.W. Computer-Based Robust Engineering: Essentials for DFSS. *Technometrics* **2006**, *48*, 153. [CrossRef]
45. Beniaguev, D.; Segev, I.; London, M. Single cortical neurons as deep artificial neural networks. *Neuron* **2021**, *109*, 2727–2739. [CrossRef] [PubMed]
46. Zang, Y.; Dieudonné, S.; De Schutter, E. Voltage-and branch-specific climbing fiber responses in Purkinje cells. *Cell Rep.* **2018**, *24*, 1536–1549. [CrossRef] [PubMed]
47. Zang, Y.; De Schutter, E. The cellular electrophysiological properties underlying multiplexed coding in Purkinje cells. *J. Neurosci.* **2021**, *41*, 1850–1863. [CrossRef] [PubMed]
48. Awad, N.; Ali, M.; Liang, J.; Qu, B.; Suganthan, P. *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report; Nanyang Technological University: Singapore, 2016.