Fast Training Algorithms for Deep Convolutional Fuzzy Systems With Application to Stock Index Prediction

Li-Xin Wang

Abstract—A deep convolutional fuzzy system (DCFS) on a high-dimensional input space is a multilayer connection of many low-dimensional fuzzy systems, where the input variables to the low-dimensional fuzzy systems are selected through a moving window across the input spaces of the layers. To design the DCFS based on an input–output data pairs, we propose a bottom–up layer-by-layer scheme. Specifically, by viewing each of the first-layer fuzzy systems as a weak estimator of the output based only on a very small portion of the input variables, we design these fuzzy systems using the Wang–Mendel method. After the first-layer fuzzy systems are designed, we pass the data through the first layer to form a new dataset and design the second-layer fuzzy systems based on this new dataset in the same way as designing the first-layer fuzzy systems. Repeating this process layer-by-layer, we design the whole DCFS. We also propose a DCFS with parameter sharing to save memory and computation. We apply the DCFS models to predict a synthetic dataset in the same way as designing the first-layer fuzzy systems. Repeating this process layer-by-layer, we design the whole DCFS. Specifically, by viewing each of the first-layer fuzzy systems as a weak estimator of the output based only on a very small portion of the input variables, we design these fuzzy systems using the Wang–Mendel method. After the first-layer fuzzy systems are designed, we pass the data through the first layer to form a new dataset and design the second-layer fuzzy systems based on this new dataset in the same way as designing the first-layer fuzzy systems. Repeating this process layer-by-layer, we design the whole DCFS. We also propose a DCFS with parameter sharing to save memory and computation. We apply the DCFS models to predict a synthetic chaotic plus random time-series and the real Hang Seng Index of the Hong Kong stock market.

Index Terms—Deep learning, hierarchical fuzzy systems, stock index prediction, Wang–Mendel (WM) method.

I. INTRODUCTION

THE GREAT success of deep convolutional neural networks (DCNN) [1], [2] in solving complex practical problems [3], [4] reveals a basic fact that multilevel structures are very powerful models in representing complex relationships. The main problems of DCNN are the huge computational load to train the tones of parameters of the DCNN and the lack of interpretability for these huge number of model parameters [5]. The goal of this article is to develop deep convolutional fuzzy systems (DCFS) and fast training algorithms for the DCFS to explore the power of multilevel rule-based representations and to overcome the computational and interpretability difficulties of the DCNN.

Hierarchical fuzzy systems were proposed by Raju et al. [6], roughly the same time when LeCun et al. [1] introduced the DCNNs in 1990. In the late 1990s, the basic properties of hierarchical fuzzy systems such as universal approximation were proved in [7] and a backpropagation (BP) algorithm was developed in [8] to train the hierarchical fuzzy systems based on input–output data. Then, a wave of research on the hierarchical fuzzy systems was conducted in the fuzzy community around the middle 2000s—the same period when Hinton et al. [2] proposed the celebrated new training algorithm for deep neural networks in 2006 which has led to the current AI boom. During this period, the structural and approximation properties of hierarchical fuzzy systems were studied in depth [9]–[12] and many new methods for designing hierarchical fuzzy systems were proposed [13]–[16]. Since then, the hierarchical fuzzy models have been applied to a wide variety of practical problems, such as environmental monitoring [17], educational assessment [18], video deinterlacing [19], price negotiation [20], mobile robots automation [21]–[23], self-nomintating in peer-to-peer networks [24], linguistic hierarchy [25], hotel location selection [26], smart structures [27], weapon target assignment [28], image description [29], nutrition evaluation [30], spacecraft control [31], photovoltaic management [32], and wastewater treatment [33]. More recently, the research on the hierarchical fuzzy systems has been advanced along many directions, such as fast implementation [34], adaptive control [35], multiobjective optimization [36], interpretability [37], classification [38], [39] and so on.

The research works summarized in the last paragraph demonstrate that the hierarchical fuzzy models are successful in solving many practical problems. However, the applications of the hierarchical fuzzy models have been generally restricted to the low-dimensional problems with small datasets. Furthermore, the current training algorithms for hierarchical fuzzy systems are of the same types as those for deep neural networks, which are computationally intensive when applied to the high-dimensional problems with big data. The heavy computational load is mainly due to the iterative nature of the training algorithms (multiple passes of the data) and it may take a long time to converge. Since the parameters of fuzzy systems have clear physical meanings (a clear connection to the input/output variables and the data) which the neural network parameters do not have, we can take advantage of these physical meanings to develop the fast training algorithms for the parameters. The Wang–Mendel (WM) method proposed in [40] and [41] is such a fast training algorithm that uses the training data only one-pass to determine the fuzzy system parameters.
Fig. 1. General structure of a DCFS.

The basic idea of this article is to use the WM method to design the low-dimensional fuzzy systems in a bottom–up layer-by-layer fashion so that a DCFS is eventually constructed, where the inputs to the low-dimensional fuzzy systems are selected through a convolutional operator (a moving window). These low-dimensional fuzzy systems may be viewed as the weak estimators [42] of the output variable. But, unlike the classical ensemble methods in machine learning [43] such as bagging [44], random forest [45], or boosting [46], the weak estimators (the low-dimensional fuzzy systems) in our DCFS models are constructed in a layer-by-layer fashion. Specifically, the first-level fuzzy systems may be viewed as the ordinary weak estimators, with each fuzzy system uses only a very small number of the input variables from the high-dimensional input space. After the first-level fuzzy systems are designed using the standard WM method, they are fixed and their outputs form the input space to the second-level fuzzy systems. By passing the training data through the fixed first-level fuzzy systems, a new dataset is generated, and the second-level fuzzy systems are designed based on this new dataset in the same way as the first-level fuzzy systems. This process continues, layer after layer, until the DCFS is constructed.

II. STRUCTURE OF DCFS

We begin with the definition of the general DCFS.

Definition 1: The general structure of a DCFS is illustrated in Fig. 1, where the input vector \((x_0^1, x_0^2, \ldots, x_0^n)\) to the DCFS is generally of very high dimension, and the output \(x^L\) is a scalar (a multioutput DCFS may be designed as multiple single-output DCFS). Level \(l (l = 1, 2, \ldots, L - 1)\) consists of \(n^l\) fuzzy systems \(FS^l_i (i = 1, 2, \ldots, n^l)\) whose outputs are denoted as \(x^l_i\), which are inputs to Level \(l + 1\). The top level, Level \(L\), has only one fuzzy system \(FS^L\) that combines the \(n^{L-1}\) outputs from Level \(L - 1\) to produce the final output \(x^L\). The input sets \(I_1^l, I_2^l, \ldots, I_{n^l}^l\) to the fuzzy systems \(FS^l_1, FS^l_2, \ldots, FS^l_{n^l} (l = 1, 2, \ldots, L - 1)\) are selected from the previous level’s outputs \(x_{n^l-1}^{l-1}, x_{n^l-2}^{l-1}, \ldots, x_{n^l-n^l}^{l-1}\) through a moving window of the length \(m\), where the window size \(m\) is usually a small number such as 3, 4, or 5.
The moving window may take a variety of moving schemes. For example, it may move one variable at a time starting from $x_i^{-1}$ until $x_{n_i}^{-1}$ is covered, and this gives

$$I_1 = (x_1^{-1}, \ldots, x_m^{-1})$$
$$I_2 = (x_2^{-1}, \ldots, x_m^{-1})$$
$$I_l = (x_l^{-1}, \ldots, x_m^{-1})$$
$$I_{n_l-1-m+1} = (x_{n_l-m+1}, \ldots, x_{n_l-1})$$

(1)

where $l = 1, 2, \ldots, L - 1$ (for Level $l = 1$ we have $n_0 = n$). For this one-variable-at-a-time moving scheme, we have

$$n_l = n^{l-1} - m + 1$$

(2)

for $l = 1, 2, \ldots, L - 1$ with $n_0 = n$, from which we get

$$n_l = n - l (m - 1).$$

(3)

If we do not want to use too many fuzzy systems $FS_l^i$ in the construction of the DCFS to improve the efficiency of each fuzzy system $FS_l^i$, we may move the window more than one variable at a time to cover the input variables in the levels. In the extreme case, we may move the window variables each time for all the fuzzy systems $FS_l^i$, so that a level DCFS can cover $m^l$ input variables. For a $L = 5$ level, DCFS with $m = 5$ inputs to each fuzzy system $FS_l^i$, for example, $m_L = 3125$ input variables can be covered. Also, the window size $m$ may be different for different fuzzy systems $FS_l^i$ to introduce more flexibility to the DCFS model.

The fuzzy systems $FS_l^i (i = 1, 2, \ldots, n_l, l = 1, 2, \ldots, L - 1)$ are standard fuzzy systems [58] constructed as follows. For each input variable $x_i^{-1}, \ldots, x_{m_i-1}^{-1} \in I_i^l$ to the fuzzy system $FS_l^i$ (note that $x_i^{-1}$ is the first input variable to $FS_l^i$, and may not be the $i$th input variable to Level $l$), define $q$ fuzzy sets $A^1, A^2, \ldots, A^q$, as shown in Fig. 2, where the centers of the $q$ fuzzy sets are equally spaced and the endpoints $\min x_j$ and $\max x_j$ are determined from the training data (the details will be given in Section III when we will develop the training algorithms for the DCFS). The fuzzy system $FS_l^i$: $(x_i^{-1}, \ldots, x_{m_i-1}^{-1}) \rightarrow x_i^l$ is

$$x_i^l = FS_l^i (x_i^{-1}, \ldots, x_{m_i-1}^{-1})$$

$$= \frac{\sum q \sum q \sum q c_{j_1, \ldots, j_m} A_{j_1} (x_i^{-1}) \ldots A_{j_m} (x_{m_i-1})}{\sum q c_{j_1, \ldots, j_m} A_{j_1} (x_i^{-1}) \ldots A_{j_m} (x_{m_i-1})}$$

(4)

which is constructed from the following $q^m$ fuzzy IF–THEN rules:

$$\text{IF } x_i^{-1}\text{ is } A_{j_1} \text{ and } \ldots \text{ and } x_{m_i-1}^{-1} \text{ is } A_{j_m} \text{, THEN } y \text{ is } B_{i_1, \ldots, i_l}$$

(5)

where the membership functions $A_{i_1}^1, \ldots, A_{i_m}^q$ are given in Fig. 2, and the parameters $c_{j_1, \ldots, j_m}$ are the centers of the fuzzy sets $B_{i_1, \ldots, i_l}$ and will be designed using the training algorithms in Section III.

For the membership functions in Fig. 2, we see that $\sum_{j_1=1}^q A_{j_1} (x_i^{-1}) = 1, \ldots, \sum_{j_m=1}^q A_{j_m} (x_{m_i-1}) = 1$, so the denominator of the fuzzy system $FS_l^i$ of (4) is equal to 1

$$\frac{q \sum_{j_1=1}^q \sum_{j_m=1}^q A_{j_1} (x_i^{-1}) \ldots A_{j_m} (x_{m_i-1})}{(\sum_{j_1=1}^q A_{j_1} (x_i^{-1})) \ldots (\sum_{j_m=1}^q A_{j_m} (x_{m_i-1}))}$$

$$= 1$$

(6)

and the fuzzy system $FS_l^i$ of (4) is simplified to

$$x_i^l = FS_l^i (x_i^{-1}, \ldots, x_{m_i-1})$$

$$= \frac{q \sum_{j_1=1}^q \sum_{j_m=1}^q c_{j_1, \ldots, j_m} A_{j_1} (x_i^{-1}) \ldots A_{j_m} (x_{m_i-1})}{(\sum_{j_1=1}^q A_{j_1} (x_i^{-1})) \ldots (\sum_{j_m=1}^q A_{j_m} (x_{m_i-1}))}$$

(7)

where $i = 1, 2, \ldots, n_l$ and $l = 1, 2, \ldots, L - 1$. The top level fuzzy system $FS_L$ is in the same form of (7) with $n L = n$ input variables $x_i^{-1}, \ldots, x_{n-1}$.

We claimed in Section I that the DCFS has better interpretability than the DCNN, and now we show the details of how to interpret the DCFS in terms of the fuzzy IF–THEN rules (5) and the parameters $c_{j_1, \ldots, j_m}$. First, notice that the fuzzy system $FS_l^i (x_i^{-1}, \ldots, x_{m_i-1})$ of (7) is constructed from the $q^m$ fuzzy IF–THEN rules (5) with each rule covering a cell $(j_1, \ldots, j_m)$ in the $m$-dimensional input space to $FS_l^i (x_i^{-1}, \ldots, x_{m_i-1})$. Fig. 3 illustrates the case of $m = 2$ and $q = 5$, where the two-dimensional (2-D) input space is partitioned into $q^m = 5^2 = 25$ cells with the cell $(i, j) (i, j = 1, \ldots, 5)$ covered by the fuzzy IF–THEN rule

$$\text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } A_j \text{, THEN } y \text{ is } B_{ij}$$

(8)
and the parameter $c^{ij}$ being the center of the fuzzy set $B_j$. For a given input point, say the point $(x_1^*, x_2^*)$ in Fig. 3 (left), the action of the fuzzy system $FS(x_1^*, x_2^*)$ can be represented by the fuzzy IF–THEN rule: IF $x_1$ is $A^2$ and $x_2$ is $A^3$, THEN $y$ is $B_2$, which covers the cell over $(x_1^*, x_2^*)$. That is, the fuzzy system takes local actions with one rule responsible mainly for one cell. Therefore, for a given point $(x_1^{l-1}, x_2^{l-1}, \ldots, x_{m-1}^{l-1})$, the action of the fuzzy system $FS_l(x_1^{l-1}, x_2^{l-1}, \ldots, x_{m-1}^{l-1})$ of (7) can be represented by a single parameter $c^{l1}$ that represents the fuzzy IF–THEN rule in the form of (5).

Because each of the fuzzy system $FS_l$ in the general DCFS of Fig. 1 can be represented by a single parameter $c^{l1, \ldots, l_m}$ (as we demonstrated in the last paragraph), the whole action of the DCFS on any given input point can be interpreted by a graph of connected $c^{l1, \ldots, l_m}$'s. Fig. 3 (the right part) shows an example of six-level DCFS with $n = 7$ inputs to the DCFS and $m = 2$ inputs to each fuzzy system $FS_l$ in the DCFS. For a given input $(x_1^*, \ldots, x_n^*)$ in Fig. 3, if the DCFS gives a bad output $y$ (such as causing an accident in an autocar application), then from the DCFS graph in the right part of Fig. 3 we can easily check what rules cause the bad output and take appropriate changes to these rules so that the mistake will not happen again. We will discuss this easy-error-correction property of the DCFS in more details after we develop the training algorithms in Section III.

Finally, we introduce a special DCFS that can greatly reduce the memory requirement and the computational cost of the training algorithms for the DCFS to be developed in Section III. We see from (7) that each fuzzy system $FS_l$ has $q^m$ free parameters $c^{l1, \ldots, l_m}$ to be designed and stored in the computer memory. The total computational and storage requirement is proportional to $q^m \sum_{l=1}^L n^l$, which may be a large number for the high-dimensional problems. To reduce the computational and storage requirement, we introduce a parameter sharing scheme where the fuzzy systems in the same level are identical (sharing the same $c^{l1, \ldots, l_m}$ parameters).

Definition 2: A DCFS with parameter sharing is a DCFS where the fuzzy systems $FS_l$ in the same level $l$ are identical, i.e., $FS_1 = \cdots = FS_L$ for $l = 1, 2, \ldots, L$, and we denote this identical fuzzy system in Level $l$ as $FS_l$. A DCFS with parameter sharing has $q^m L$ free parameters $c^{l1, \ldots, l_m}$ to be designed and stored.

We now move to Section III to develop a number of fast training algorithms to determine the parameters $c^{l1, \ldots, l_m}$ of the fuzzy system (7) based on input–output data for the general DCFS and the DCFS with parameter sharing.

III. Fast Training Algorithms for DCFS

Task I (Offline training): Given $N$ input–output data pairs

$$ [x_1^0(k), x_2^0(k), \ldots, x_n^0(k); y^0(k)], \quad k = 1, 2, \ldots, N \quad (9) $$

where $x_1^0(k), x_2^0(k), \ldots, x_n^0(k)$ are the inputs and $y^0(k)$ is the output, our task is to design a DCFS in Fig. 1 to match these input–output data pairs.

First, we develop a training algorithm for the general DCFS in Definition 1 to match the input–output data pairs of (9) (Training Algorithm 1). Then, we show how to design the DCFS with parameter sharing in Definition 2 in Training Algorithm 2. Finally, in Training Algorithms 3 and 4, we show how to do an online training for the general DCFS and the DCFS with parameter sharing, respectively.
Training Algorithm 1 (For general DCFS): Given the input–output data pairs of (9), we design the general DCFS in Definition 1 with the fuzzy systems $FS_2^i$ in the form of (7) through the following steps.

Step 1: Choose the moving window size $m$ and the moving scheme (such as moving one-variable-at-a-time or other schemes).

Step 2: Design the Level 1 fuzzy systems $FS_2^i$ in the form of (7) with $m$ input $x_1^1$, $x_2^1$, $x_{m+1}^1$, $i = 1, 2, \ldots, n^1$, using the WM method [40], [41], where the input–output data pairs used to design the $FS_2^i$ are

$$[x_0^0(k), \ldots, x_{m+i-1}^0(k); y_0^0(k)], \quad k = 1, 2, \ldots, N.$$

(10)

2.1: For each cell $(j_1, j_2, \ldots, j_m)$ with $j_1, j_2, \ldots, j_m = 1, 2, \ldots, q$, set the initial values of the weight parameter $w_{j_1} \cdots j_m$ and the weight-output parameter $w_{j_1} \cdots j_m$ equal to zero.

2.2: For each input $x_0^1, \ldots, x_{m+i-1}^1$ to $FS_2^i$, consider the q fuzzy sets $A_1, A_2, \ldots, A_q$ in Fig. 2 and choose the endpoints $0$ and the mov-

2.3: For each input–output data pairs of (10) starting from $k = 1$, determine the fuzzy sets $A_1, A_2, \ldots, A_q$ at $x_0^1(k), \ldots, x_0^1(k)$, respectively, i.e., determine

$$j_*^1 = \arg\max_{j \in (1,2,\ldots,q)} (A_j(x_0^1(k)))$$

(11)

$$j_*^1 = \arg\max_{j \in (1,2,\ldots,q)} (A_j(x_0^1(k))).$$

(12)

2.4: Update the weight and weight-output parameters for cell $(j_1^*, \ldots, j_m^*)$, $w_{j_1^*} \cdots j_m^*$, and $w_{j_1^*} \cdots j_m^*$ through

$$w_{j_1^*} \cdots j_m^* = w_{j_1^*} \cdots j_m^*$$

$$+ A_1^j(x_0^1(k)) \cdots A_m^j(x_{m+i-1}^1(k))$$(13)

$$w_{j_1^*} \cdots j_m^* = w_{j_1^*} \cdots j_m^*$$

$$+ (A_1^j(x_0^1(k)) \cdots A_m^j(x_{m+i-1}^1(k))) y_0^1(k).$$

(14)

2.5: Repeat 2.3 and 2.4 for $k = 1, 2, \ldots, N$. For the cells $(j_1, j_2, \ldots, j_m)$ with $w_{j_1^*} \cdots j_m^* \neq 0$, determine the parameters $c_{j_1^*} \cdots j_m^*$ in the fuzzy system $FS_1^i$ of (7) as

$$c_{j_1^*} \cdots j_m^* = w_{j_1^*} \cdots j_m^*/w_{j_1^*} \cdots j_m^*.$$ (15)

We call the cells $(j_1, \ldots, j_m)$ with $w_{j_1^*} \cdots j_m^* \neq 0$ covered by data, and define

$$C(0) = \{(j_1, \ldots, j_m) | (j_1, \ldots, j_m) \text{ is covered by data}\}.\quad (16)$$

2.6: For each cell $(j_1, \ldots, j_m)$ not in $C(0)$, search its neighbors to see whether they are in $C(0)$, where two cells $(j_1, \ldots, j_m)$ and $(j'_1, \ldots, j'_m)$ are neighbors to each other if $j_i = j'_i$ for all $i = 1, \ldots, m$ except at one location $r$, such that $j_r = j'_r + 1$ or $j_r = j'_r - 1$. For the cells $(j_1, \ldots, j_m)$ not in $C(0)$ that have at least one neighbor in $C(0)$, determine the $c_{j_1^*} \cdots j_m^*$ as the average of the $c_{j_1^*} \cdots j_m^*$'s of its neighbors in $C(0)$. Define

$$C(1) = C(0)$$

$$\oplus \{(j_1, \ldots, j_m) | c_{j_1^*} \cdots j_m^* \text{ determined in this step}\}.$$ (17)

2.7: Repeat 2.6 with $C(0)$ replaced by $C(1)$ and $C(1)$ replaced by $C(2)$, and continue this process to get $C(3)$, $C(4)$, ..., $C(p)$ that contains all the cells $(j_1, j_2, \ldots, j_m)$ with $j_1, j_2, \ldots, j_m = 1, 2, \ldots, q$. Fig. 4 illustrates this process. (More details of the WM method are given in [41].)

Step 3: Suppose the fuzzy systems in Level 1 to Level $l - 1$ have already been designed, we now design the fuzzy systems $FS_l^i$, $i = 1, 2, \ldots, n^l$, in Level $l$, starting from $l = 2$ (in this case, the Level $l - 1 = 1$ fuzzy systems $FS_1^i$, $i = 1, 2, \ldots, n^1$, have already been designed in Step 2).

3.1: For each input–output data pairs of (9) with $k = 1, 2, \ldots, N$, put $x_0^{l-1}(k), x_2^{l-1}(k), \ldots, x_{n-l}^{l-1}(k)$ as inputs to Level 1 and compute upwards along the DCFS to get the outputs of Level $l - 1$, denoted as $x_1^{l-1}(k), x_2^{l-1}(k), \ldots, x_{n-1}^{l-1}(k)$; view

$$[x_1^{l-1}(k), x_2^{l-1}(k), \ldots, x_{n-1}^{l-1}(k); y_0^{l}(k)]$$

(18)

as the new input–output data pairs for designing the Level $l$ fuzzy systems $FS_l^i$.

3.2: Use the same procedure in Step 2 to design the Level $l$ fuzzy systems $FS_l^i$ in the form of (7), $i = 1, 2, \ldots, n^l$, with the original input–output data pairs (9) replaced by the new input–output data pairs (18), and all the Level 1 variables in Step 2 replaced by the corresponding Level $l$ variables (for example, replace $n^1$ by $n^l$, $x_1^{l-1}(k), x_2^{l-1}(k), \ldots, x_{n-l}^{l-1}(k)$ by $x_1^{l-1}(k), x_2^{l-1}(k), \ldots, x_{n-l}^{l-1}(k)$, etc.).

3.3: Set $l = l + 1$ and repeat 3.1 and 3.2 until the Level $L$ fuzzy system $FS_L$ is designed.

Figs. 5–7 illustrate Training Algorithm 1 for the design of a three-level DCFS with 15 inputs and two-variable-at-a-time moving window scheme, where Figs. 5–7 are for the design of Levels 1–3, respectively. The MATLAB code of Training Algorithm 1 for a 11-input 5-level DCFS is given in the Supplemental Material, which can be easily modified for other applications. A few remarks about Training Algorithm 1 are now in order.

Remark 1 (Layer-by-layer construction): As discussed in Section I, the DCFS with Training Algorithm 1 may be viewed as a layered combination of weak estimators. Specifically, among the high-dimensional input variables $x_1, x_2, \ldots, x_n$ to the DCFS, each fuzzy system $FS_1^i(i = 1, 2, \ldots, n^1)$ in Level 1 selects only a very small number of $m$ variables as its inputs, so each fuzzy system $FS_1^i$ may be viewed as a weak estimator for the output $y_0^{l}$. After these $n^1$ Level 1 weak estimators $FS_1^i$ are designed, they are fixed and their outputs $x_1^1, x_2^1, \ldots, x_{n^1}^1$ form the $n^1$-dimensional input space to the Level 2 fuzzy systems $FS_2^i(i = 1, 2, \ldots, n^2)$. Here again as in Level 1, each Level 2
Fig. 4. Illustration of the WM Method to expand the cells from those covered by data (red) to the whole input space, where $C(0) = \text{red}$, $C(1) = C(0) + \text{green}$, $C(2) = C(1) + \text{blue}$, $C(3) = C(2) + \text{yellow}$, ..., and $C(8) = \text{all the cells}$.

Fig. 5. Illustration of Training Algorithm 1 for general DCFS in the design of the Level 1 fuzzy systems $FS_l^1$.

Remark 2 (Physical meaning of the parameter design): From (13)–(15), we see that the parameters $c_{j1}^{\cdots,jm}$ in the fuzzy system (7) are designed as the weighted average of the outputs $y^0(k)$ whose corresponding inputs $x_{l-1}^{i}(k), \ldots, x_{m+1-i}^{i}(k)$ fall into the cell $(j_1, \ldots, j_m)$, with the weight equal to the membership value $A_{j_1}(x_{l-1}^{i}(k)) \cdots A_{j_m}(x_{m+1-i}^{i}(k))$. If no data fall into a cell $(j_1, \ldots, j_m)$, then the $c_{j1}^{\cdots,jm}$ of the cell is determined through the extrapolation scheme of Steps 2.6 and 2.7 (illustrated in Fig. 4). We can use this simple scheme to design the parameters $c_{j1}^{\cdots,jm}$ because of the clear physical meaning of the $c_{j1}^{\cdots,jm}$. Specifically, the $c_{j1}^{\cdots,jm}$ is the center of the then part membership function of the fuzzy IF–THEN rule (5) for the cell $(j_1, \ldots, j_m)$, so the $c_{j1}^{\cdots,jm}$ may be viewed as the estimate of the desired output $y^0$ based on the fuzzy IF–THEN rule (5) at cell $(j_1, \ldots, j_m)$. Therefore, a good way to design the $c_{j1}^{\cdots,jm}$ is to put it as the weighted average of the $y^0(k)$’s whose corresponding inputs $x_{l-1}^{i}(k), \ldots, x_{m+1-i}^{i}(k)$ fall into the cell $(j_1, \ldots, j_m)$. The optimality of designing the parameters in this way is studied in [59].

Remark 3 (Fast training with low computational cost): From 2.3 to 2.5 in Training Algorithm 1, we see that to design each of the fuzzy systems $FS_l^1$ in the DCFS, the $N$ input–output data
Training Algorithm 2 (For the DCFS with parameter sharing): Given the input–output data pairs of (9), the DCFS with parameter sharing in Definition 2 are designed as follows:

Step 1: The same as Step 1 in Training Algorithm 1.

Step 2: Design the first Level 1 fuzzy system $FS_1^1 = FS_1^1$ in the form of (7) with the following $n^1 N$ input–output data pairs:

$$[x_0^0(k), \ldots, x_{m}^0(k); y_0^0(k)], \quad k = 1, 2, \ldots, N$$

$$\vdots$$

$$[x_{n^1}^0(k), \ldots, x_{m+n^1-1}^0(k); y_0^0(k)], \quad k = 1, 2, \ldots, N$$

(19)

using the same procedure 2.1–2.7 in Training Algorithm 1. Then, copy this $FS_1^1 = FS_1^1$ to all the rest $FS_i^1 (i = 2, \ldots, n^1)$ in Level 1.

Step 3: The same as Step 3 of Training Algorithm 1 except that the Step 2 there is replaced by the Step 2 here.

Fig. 8 illustrates the Training Algorithm 2 (for the DCFS with parameter sharing) for the design of the Level 1 fuzzy systems.
Fig. 8. Illustration of Training Algorithm 2 for the DCFS with parameter sharing in the design of the Level 1 fuzzy systems $FS^1_1 = \cdots = FS^1_q = FS^1$.

$FS^1_1 = \cdots = FS^1_q = FS^1$ (comparing Fig. 8 with Fig. 5 to see the difference between Training Algorithms 1 and 2). The design of the fuzzy systems $FS^1_1 = \cdots = FS^1_q = FS^1$ in other levels ($l = 2, \ldots, L$) can be illustrated in a similar way as in Figs. 6 and 7. A remark about the Training Algorithm 2 is as follows.

Remark 4 (Low memory requirement and fast training): The computational load of Training Algorithm 2 (for the DCFS with parameter sharing) is approximately $O(N \sum_{l=1}^{L} n^l + q^m L)$, where $O(N \sum_{l=1}^{L} n^l)$ accounts for the one-pass of the data (19) in the computation (13)-(15) for the data covered $c^{j_1 \cdots j_m}$, and $O(q^m L)$ accounts for the computation of the parameters $c^{j_1 \cdots j_m}$ of the L fuzzy systems $FS^l_i (l = 2, \ldots, L)$ in Steps 2.5–2.7 (illustrated in Fig. 4). Comparing with the computational load $O(N + q^m \sum_{l=1}^{L} n^l)$ of Training Algorithm 1 (for general DCFS), we see that a computational reduction of $O(q^m) (\sum_{l=1}^{L} n^l - L)$ is achieved with Training Algorithm 2. For memory requirement (mainly for the storage of the parameters $c^{j_1 \cdots j_m}$), it is reduced from $O(q^m \sum_{l=1}^{L} n^l)$ (for general DCFS) to $O(q^m L)$ (for DCFS with parameter sharing), a reduction of $O(q^m) (\sum_{l=1}^{L} n^l - L)$, which is in the same order as the computational reduction. For the DCFS in Fig. 3, for example, we have $L = 6$ and $\sum_{l=1}^{L} n^l = 21$, so the memory requirement is reduced from $21 O(q^m)$ to $6 O(q^m)$, a reduction of $\frac{21 - 6}{21} = 71.4\%$.

In Training Algorithms 1 and 2, the $N$ input–output data pairs (9) are processed in a batch fashion. In many real-world situations, the data are collected in real time, so it is interesting to design the DCFS in a recursive way with the new data coming in real time. This gives Task 2 as follows:

Task 2 (Online training): Let the input–output data pairs

$$[x^0(k), x^1(k), \ldots, x^n(k); y^0(k)], \quad k = 1, 2, 3, \ldots$$

be given online with $k$ being the time index. Suppose a DCFS has already been designed up to $k-1$, and we denote this DCFS as the DCFS$(k-1)$. Let the initial DCFS(0) be designed either by Training Algorithm 1 for general DCFS or by Training Algorithm 2 for the DCFS with parameter sharing, and our task is to update the DCFS$(k-1)$ based on the new data pair $[x^0(k), x^1(k), \ldots, x^n(k); y^0(k)]$ to get DCFS$(k)$.

We now show how to do Task 2 for the general DCFS and the DCFS with parameter sharing in Training Algorithms 3 and 4, respectively.

Training Algorithm 3 (Online training for general DCFS): For Task 2, let the initial DCFS(0) be a general DCFS (Definition 1) designed by Training Algorithm 1 and $c^{j_1 \cdots j_m}(k-1)$ be the parameters of the fuzzy systems $FS^1_i$ in the form of (7) in the DCFS$(k-1)$. Then, the parameters $c^{j_1 \cdots j_m}(k)$ of the $FS^1_i$ in the DCFS$(k)$ are updated through the following steps:

Step 1: Update the $c^{j_1 \cdots j_m}(k)$ of the Level 1 fuzzy systems $FS^1_i(x^0_{m+i-1}(k); y^0(k))$. Specifically, determine

$$j_1^* = \arg\max_{j \in \{1, 2, \ldots, q\}} (A^{j_1}(x^0_1(k)))$$

$$j_{m}^* = \arg\max_{j \in \{1, 2, \ldots, q\}} (A^{j_{m}}(x^0_{m+i-1}(k))))$$

and update

$$c^{j_1 \cdots j_{m}}(k) = \alpha A^{j_1}(x^0_1(k)) \cdots A^{j_{m}}(x^0_{m+i-1}(k)) y^0(k) + [1 - \alpha A^{j_1}(x^0_1(k)) \cdots A^{j_{m}}(x^0_{m+i-1}(k))]} c^{j_1 \cdots j_{m}}(k-1)$$

where $\alpha \in [0, 1]$ is a weighting factor. For other cells $(j_1, \ldots, j_{m}) (j_1, \ldots, j_{m} = 1, 2, \ldots, q)$ not equal to $(j_1^*, \ldots, j_{m}^*)$, set

$$c^{j_1 \cdots j_{m}}(k) = c^{j_1 \cdots j_{m}}(k-1).$$
Step 2: Pass the new data \( x_0^0(k), x_0^1(k), \ldots, x_0^n(k) \) through Level 1 to get the new input–output data pair \([x_1^0(k), x_1^1(k), \ldots, x_m^1(k); y_0^1(k)]\) for the Level 2 fuzzy systems \( F_{S_1}^2 \). Update the \( c^{j_1,\ldots,j_m}(k) \) of the Level 2 fuzzy systems \( F_{S_1}^2(x_1^1, \ldots, x_{m+i-1}^1) \) in the form of (7) from \( c^{j_1,\ldots,j_m}(k-1) \) using the same (21)–(23) as in Step 1 with \( x_1^2(k), \ldots, x_{m+i-1}^2(k) \) replaced by \( x_1^1(k), \ldots, x_{m+i-1}^1(k) \), respectively.

Step 3: Repeat Step 2 for the Levels 3, 4, ..., until the top Level L.

Training Algorithm 4 (Online training for the DCFS with parameter sharing): For Task 2, let the initial DCFS(0) be a parameter sharing DCFS (Definition 2) designed by Training Algorithm 2 and the \( c^{j_1,\ldots,j_m}(k-1) \) be the parameters of the fuzzy systems \( F_{S_1}^2 \) in the form of (7) in the DCFS \( (k-1) \). Then, the parameters \( c^{j_1,\ldots,j_m}(k) \) of the FS\( S_1^2 \) in the DCFS(k) are updated through the following steps:

**Step 1:** Update the \( c^{j_1,\ldots,j_m}(k) \) of the first Level 1 fuzzy system \( F_{S_1}^1 = F_{S_1}^1 \) in the form of (7) with the following \( n^1 \) input–output data pairs:
\[
[x_1^0(k), \ldots, x_{m+i-1}^0(k); y_0^0(k)], \quad i = 1, \ldots, n^1. \tag{24}
\]

Specifically, for each of the \( n^1 \) input–output data pairs of (24), update the \( c^{j_1,\ldots,j_m}(k) \) using (21) and (22); for the rest \( c^{j_1,\ldots,j_m}(k) \), keep them the same using (23). Then, copy this \( F_{S_1}^1 = F_{S_1}^1 \) to all the rest \( F_{S_1}^i \) \((i = 2, \ldots, n^1)\) in Level 1.

Steps 2 and 3: The same as Steps 2 and 3 of Training Algorithm 3, with the Step 1 there replaced by the Step 1 here.

Two remarks about the Training Algorithms 3 and 4 are now in order.

**Remark 5 (Physical meaning and a very low computational cost of the updating law):** From (21)–(23), we see that for each new input–output data pair, only the \( c^{j_1,\ldots,j_m} \) (the first \( k \) data points) of the new input data point \( (x_1^0(k), \ldots, x_{m+i-1}^0(k)) \) is updated, and the rest \( c^{j_1,\ldots,j_m} \)'s remain unchanged. The new \( c^{j_1,\ldots,j_m}(k) \) (22) is a weighted average of the output \( y_0^0(k) \) and the previous \( c^{j_1,\ldots,j_m}(k-1) \) with weight \( \alpha A^{j_1}(x_1^0(k)) \ldots A^{j_m}(x_{m+i-1}^0(k)) \), which is a measure of the degree that the input data point \( (x_1^0(k), \ldots, x_{m+i-1}^0(k)) \) belongs to the cell \((j_1, \ldots, j_m)\). Since only one parameter \( c^{j_1,\ldots,j_m}(k) \) needs to be updated and the computational load of the updating law (21)–(22) is very low, Training Algorithms 3 and 4 may be implemented on the simple devices such as a mobile phone. Therefore, an initial DCFS designed with the offline Training Algorithm 1 or 2 may be downloaded to the customer’s simple device such as a mobile phone, then the customer may update the DCFS using the online Training Algorithm 3 or 4 with the new data that the customer observes in real life.

**Remark 6 (Easy error correction):** As we discussed in Section I that a main problem of the black-box DCNN is that if something goes wrong, we do not know which part of the DCNN should be corrected so that the same mistake will not take place again. The DCFS designed with the Training Algorithm 3 or 4 can easily solve this problem of the DCNN. Specifically, if a DCFS makes a mistake at an input point, say the point \((x_1^0, \ldots, x_7^0)\) to the DCFS in Fig. 3, then we can update the \( c^{j_1,\ldots,j_m} \)'s of the DCFS in Fig. 3 that are responsible for the mistake, using Training Algorithm 3 or 4 with the correct input–output data.

We now apply the DCFS designed with Training Algorithms 1–4 to the time-series prediction problems in Section IV.

### IV. APPLICATION TO HSI PREDICTION

Before we apply the DCFS with the training algorithms to predict the real HSI of the Hong Kong stock market in Examples 2, 4, and 5, we first try it for a synthetic chaotic time-series in Examples 1 and 3 to get some feeling about the performance of the method. Specifically, in Examples 1 and 2, we test the Training Algorithms 1 and 2 for predicting the Mackey–Glass chaotic time-series and the real HSI, respectively. In Examples 3 and 4, we apply the online Training Algorithms 3 and 4 to predict the Mackey–Glass chaotic time-series and the real HSI, respectively. In Example 5, we add more related stocks to the inputs to get better performance.

**Example 1:** Consider the Mackey–Glass chaotic time-series generated by the differential equation
\[
\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{25}
\]
with \( \tau = 50 \). Let \( r(t) \) be the return (relative change) of the chaotic time-series \( x(t) \) plus a white Gaussian noise \( n(t) \), i.e.,
\[
r(t) = \frac{x(t) - x(t-1)}{x(t-1)} + n(t) \tag{26}
\]
and we generate a synthetic chaotic plus random time-series \( y(t) \) whose return is \( r(t) \), i.e.,
\[
y(t) = y(t-1) [1 + r(t)] \tag{27}
\]
We view \( y(t) \) \((t = 0, 1, 2, \ldots)\) of (27) as the daily closing prices of a stock index. Fig. 9 plots a realization of 3000 points of \( y(t) \).

We now use the DCFS of Fig. 1 with the Training Algorithms 1 and 2 to predict the return sequence \( r(t) \) of (26). Specifically, let the \( n \) past returns up to day \( t-1 \):
\[
r(t-1), r(t-2), \ldots, r(t-n) \]
be the inputs \( x_1^0, x_2^0, \ldots, x_n^0 \) to the DCFS and the output \( x_{t}^{L} \) of the DCFS be the prediction of Fig. 9. Index generated by the Mackey–Glass chaotic time-series plus noise, where the first 2000 points are the training data and the last 1000 points are the testing data.
the return at day $t$, i.e.
\[ \hat{r}(t) = DCFS(r(t-1), r(t-2), \ldots, r(t-n)) \]  
(28)
where $\hat{r}(t)$ is the prediction of $r(t)$. In this case, the input–output data pairs of (9) become
\[ [r(k-1), r(k-2), \ldots, r(k-n) ; r(k)] \]  
(29)
where $k = t-1, t-2, \ldots, t-N+n$ (the current day is $t-1$), that is, at day $t-1$, $N$ past returns $r(t-1), r(t-2), \ldots, r(t-N)$ constitute the $N-n$ input–output pairs in the form of (29).

With $n = 11$, $m = 3$, and one-variable-at-a-time moving scheme, a five-level DCFS is established, where Levels 1–5 have 9, 7, 5, 3, and 1 fuzzy systems $FS_l^i$, respectively. Using the first 2000 points in Fig. 9 as the training data and the last 1000 points as the testing data, we simulate the Training Algorithms 1 and 2 for different values of $q$ ($q$ is the number of fuzzy sets covering each input variable, see Fig. 2). Figs. 10 and 11 plot the training (blue) and testing (red) errors of Training Algorithm 1 (for general DCFS) and Training Algorithm 2 (for the DCFS with parameter sharing) as a function of $q$, respectively. We see from Figs. 10 and 11 that for both general DCFS (with Training Algorithm 1) and the DCFS with parameter sharing (with Training Algorithm 2), the training errors (blue curves) keep decreasing as more fuzzy sets are used to cover the input variables. For the testing errors (red curves in Figs. 10 and 11), we see that they are first decreasing as $q$ increases, but then begin to increase as the DCFS models overfit the data.

We also compare the DCFS predictor with the fuzzy model trained by the BP algorithm [60], [61]. For models with similar complexity (roughly the same number of free parameters), the training and testing errors of the DCFS predictor are about 20% less than that of the BP trained fuzzy model, with a training speed at least ten times faster. Specifically, for $q = 20$ (roughly $\sum_{l=1}^{L} ni^l = 20^3 25 = 200000$ free parameters) and the 3000 chaotic time-series data in Fig. 9, the training (first 2000 points) and testing (last 1000 points) errors of the DCFS predictor and the BP trained fuzzy model are plotted in Fig. 12, where the mean square training and testing errors of the DCFS predictor are 0.0051 and 0.0063, respectively, and those of the BP trained fuzzy model are 0.0061 and 0.0077, respectively.

Example 2: The same as Example 1 except that the index generated by the Mackey–Glass chaotic time-series plus noise (see Fig. 9) is replaced by the real HSI (see Fig. 13). Specifically, considering the HSI daily closing from 2010/7/6 to 2018/8/30 (2000 data points) plotted in Fig. 13, we use the first 1500 points (from 2010/7/6 to 2016/8/4) as the training data and the last 500 points (from 2016/8/5 to 2018/8/30) as the testing data. Fig. 14 plots the training (blue) and testing (red) errors of Training Algorithm 1 (for general DCFS) as a function of $q$. Comparing Fig. 14 with Fig. 10, we see that the real HSI is much more difficult to predict than the index generated by the the Mackey–Glass chaotic time-series.
Since we view $y(t)$ of (27) as the daily closing prices of a stock index, the value of an Index Fund with $y(t)$ at day $t$, denoted as ValueIndex$(t)$, is updated daily according to

$$\text{ValueIndex}(t) = \text{ValueIndex}(t-1) \cdot [1 + r(t)]$$  \hspace{1cm} (30)

where $t = 1, 2, \ldots$, and we assume the initial investment ValueIndex$(0)$ = 100.

We now propose a trading strategy based on the DCFS prediction $\hat{r}(t)$ of (28).

Trading Strategy Based on the DCFS Prediction: At day $t-1$ [the most recent return available is $r(t-1)$], use the Training Algorithm 3 or 4 in Section III to design a DCFS of (28) to get the prediction $\hat{r}(t)$ of the return at day $t$. If $\hat{r}(t) > 0$, meaning that we predict the index $y(t)$ will go up in day $t$, then long (buy) the index at day $t-1$; if $\hat{r}(t) < 0$, meaning that we predict the index $y(t)$ will go down in day $t$, then short (sell) the index at day $t-1$.

The value of the fund with this trading strategy at day $t$, denoted as ValueDCFS$(t)$, is

$$\text{ValueDCFS}(t) = \text{ValueDCFS}(t-1) \cdot [1 + I(t) |r(t)|]$$  \hspace{1cm} (31)

where the initial investment ValueDCFS$(0)$ = 100 and

$$I(t) = \begin{cases} 1, & \text{if sign}(\hat{r}(t)) = \text{sign}(r(t)) \\ -1, & \text{if sign}(\hat{r}(t)) \neq \text{sign}(r(t)). \end{cases}$$  \hspace{1cm} (32)

The meaning of (31) and (32) is that: If $\text{sign}(\hat{r}(t)) = \text{sign}(r(t))$ that means we made the right prediction at day $t-1$, then our fund value will increase by $|r(t)|$ no matter the index $y(t)$ goes up or down at day $t$; on the other hand, if $\text{sign}(\hat{r}(t)) \neq \text{sign}(r(t))$ that means we made a wrong prediction at day $t-1$, then our fund will decrease in value by $|r(t)|$ no matter the index $y(t)$ goes up or down at day $t$.

We now test this trading strategy for the chaotic plus random index of Fig. 9 and the real HSI of Fig. 13 in Examples 3 and 4, respectively.

Example 3: The same as in Example 1 except that Training Algorithm 3 or 4 is used to build the DCFS predictor (28). Figs. 15 and 16 plot the index fund value ValueIndex$(t)$ of (30) and the DCFS fund value ValueDCFS$(t)$ of (31) using Training Algorithms 3 and 4, respectively. We see from Figs. 15 and 16 that in both cases the DCFS fund performs much better than the index fund and the general DCFS with Training Algorithm 3 is better than the parameter sharing DCFS with Training Algorithm 4.
Example 4: The same as Example 3 except that the index generated by the Mackey–Glass chaotic time-series plus noise (Fig. 9) is replaced by the real HSI (Fig. 13). Figs. 17 and 18 plot the index fund value ValueIndex(t) and the DCFS fund value ValueDCFS(t) using Training Algorithms 3 and 4, respectively. We see from Figs. 17 and 18 that the DCFS fund performs slightly better than the index fund in both the cases, and the general DCFS with Training Algorithm 3 is better than the parameter sharing DCFS with Training Algorithm 4.

To improve the performance of the DCFS fund, we notice that there may be other factors that influence the real HSI returns r(t), in addition to the past returns r(t−1), r(t−2), . . . , r(t−n) used in the DCFS model (28). For example, the returns of the major stocks at the current day t − 1 may have a stronger influence on the tomorrow’s HSI return r(t) than the HSI returns r(t−n) long in the past (when n is large). Indeed, the long-past HSI returns r(t−n) may introduce more noise than useful information for the prediction of tomorrow’s HSI return r(t). Hence, we add the returns of four major stocks in the Hong Kong stock market to the input space of the DCFS model and use only the five most recent returns as the inputs to the prediction model in the next example.

Example 5: We use the two-level DCFS in Fig. 19 as the prediction model for the HSI return r(t), where the five fuzzy systems FS1, FS2, . . . , FS5 in Level-1 use the five past daily returns of the HSI, HK0005 (HSBC Holdings plc), HK0700 (Tencent Holdings Limited), HK0388 (Hong Kong Exchanges and Clearing Limited) and HK0016 (Sun Hung Kai Properties Limited), respectively, as the inputs to produce five weak estimators x1(t), x2(t), . . . , x5(t) for the HSI return r(t), i.e.,

\[
x_1(t) = FS_1(r_{HSI}(t − 1), . . . , r_{HSI}(t − 5)), \quad (33)
\]

\[
x_2(t) = FS_2(r_{HK0005}(t − 1), . . . , r_{HK0005}(t − 5)), \quad (34)
\]

Fig. 20 plots the HSI fund value ValueIndex(t) (blue line) and ValueDCFS(t) (red line) with Training Algorithm 3 in Example 5.

\[
x_3(t) = FS_3(r_{HK0700}(t − 1), . . . , r_{HK0700}(t − 5)) \quad (35)
\]

\[
x_4(t) = FS_4(r_{HK0388}(t − 1), . . . , r_{HK0388}(t − 5)) \quad (36)
\]

\[
x_5(t) = FS_5(r_{HK0016}(t − 1), . . . , r_{HK0016}(t − 5)) \quad (37)
\]

where r_{HSI}, r_{HK0005}, r_{HK0700}, r_{HK0388}, and r_{HK0016} are the returns of the HSI and the stocks HK0005, HK0700, HK0388, and HK0016, respectively, then these five weak estimators are combined by the Level-2 fuzzy system FS2 to produce the final prediction

\[
\hat{r}(t) = FS_2(x_1(t), x_2(t), . . . , x_5(t)). \quad (38)
\]

Fig. 20 plots the HSI fund value ValueIndex(t) of (30) and the DCFS fund value ValueDCFS(t) of (31) with this two-level DCFS predictor (33)–(38) using Training Algorithm 3 for the real HSI daily returns r(t) over the three years from August 31, 2015 to August 31, 2018. Comparing the DCFS fund values ValueDCFS(t) in Figs. 17 and 20 we see a clear performance improvement by adding these four stocks into the input space of the DCFS predictor.
V. CONCLUSION
The DCFS models and the training algorithms proposed in this article have the following advantages.

1) It is fast: The data are used only once in the design of the fuzzy systems in the DCFS and no iterative training is needed.

2) It is highly interpretable: The fuzzy systems in different levels of the DCFS are weak estimators of the output variable that are constructed in a layer-by-layer, bottom–up fashion.

3) It is very flexible: The size of the moving window, the steps of each move, the number of fuzzy sets to cover the input variables, and the number of layers can be easily adjusted for better performance.

4) It is easy to correct mistakes: Because of the clear physical meaning of the parameters, it is easy to redesign the parameters with new data so that the same mistake will not happen again.

5) It may be implemented on simple devices: Due to simple computation and low memory requirement, the online training algorithms may be run on the user-end simple devices such as a mobile phone.

6) It supports online learning: Users can continuously update the DCFS models with their own new data on their own simple devices so that a wide variety of user-specific intelligent systems would be created.

7) It provides a natural structure for parallel computing: All the fuzzy systems in the same level can be trained in parallel, making the fast training algorithms even faster through parallel computing.

8) It is suitable for the high-dimensional problems.

ACKNOWLEDGMENT
The author would like to thank the reviewers for their very insightful comments that helped to improve the article.

REFERENCES
[1] Y. LeCun et al., “Handwritten digit recognition with a back-propagation network,” in Proc. Adv. Neural Inf. Process. Syst., 1990, pp. 396–404.
[2] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” Neural Comput., vol. 18, pp. 1527–1554, 2006.
[3] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, pp. 436–444, 2015.
[4] D. Silver et al., “Mastering the game of Go with deep neural networks and tree search,” Nature, vol. 529, pp. 484–489, 2016.
[5] R. Sadeghi, T. Banerjee, and W. Romine, “Early hospital mortality prediction using vital signs,” Smart Health, vol. 9, pp. 265–274, 2018.
[6] G. V. S. Raju, J. Zhou, and R. A. Kisner, “Hierarchical fuzzy control,” Int. J. Contr., vol. 54, no. 5, pp. 1201–1216, 1991.
[7] L.-X. Wang, “Universal approximation by hierarchical fuzzy systems,” Fuzzy Sets Syst., vol. 93, pp. 223–230, 1998.
[8] L.-X. Wang, “Analysis and design of hierarchical fuzzy systems,” IEEE Trans. Fuzzy Syst., vol. 7, no. 5, pp. 617–624, Oct. 1999.
[9] R. J. G. B. Campello and W. C. D. Amaral, “Hierarchical fuzzy relational models: Linguistic interpretation and universal approximation,” IEEE Trans. Fuzzy Syst., vol. 14, no. 3, pp. 446–453, Jun. 2006.
[10] M. G. Joo and J. S. Lee, “Universal approximation by hierarchical fuzzy system with constraints on the fuzzy rule,” Fuzzy Sets Syst., vol. 130, pp. 175–188, 2002.
[11] X.-J. Zeng and J. A. Keane, “Approximation capabilities of hierarchical fuzzy systems,” IEEE Trans. Fuzzy Syst., vol. 13, no. 5, pp. 659–672, Oct. 2005.
[12] J. J. J. Zeng, J. Y. Goulermans, P. Liatsis, D. Wang, and J. A. Keane, “Hierarchical fuzzy systems for function approximation on discrete input spaces with application,” IEEE Trans. Fuzzy Syst., vol. 16, no. 5, pp. 1197–1215, Oct. 2008.
[13] Y. Chen, B. Yang, A. Abraham, and L. Peng, “Automatic design of hierarchical Takagi–Sugeno type fuzzy systems using evolutionary algorithms,” IEEE Trans. Fuzzy Syst., vol. 15, no. 3, pp. 385–397, Jul. 2007.
[14] J. Dong, Y. Chen, and B. Yang, “Approach for evolving hierarchical T-S fuzzy systems,” in Proc. Int. Conf. Signal Process., 2004, vol. 2, pp. 1530–1533.
[15] Y. Chen, H. H. Cheung, W. Rattasiri, and S. K. Halgamuge, “Self-generating hierarchical fuzzy systems,” in Proc. Int. Conf. Intel. Sens. Inf. Process., 2004, pp. 348–353.
[16] P. Salgado, “Rule generation for hierarchical collaborative fuzzy system,” Appl. Math. Model., vol. 32, pp. 1159–1178, 2008.
[17] G. Back, J. Cho, M. H. Choi, and S. Kim, “On-line monitoring of oil in water using transmitted-scattered lights and clustering-based hierarchical TSK fuzzy systems,” in Proc. IEEE Int. Conf. Fuzzy Syst., 2010, pp. 1–8.
[18] R. Beshes, R. Tourki, and P. Gorce, “Engineering analysis for educational concepts’ assessment,” in Proc. 3rd Interdisciplinary Eng. Des. Educ. Conf., 2013, pp. 121–128.
[19] P. Brox, F. Bartunov, and S. Sánchez-Solano, “Tuning of a hierarchical fuzzy system for video de-interlacing,” in Proc. IEEE Int. Conf. Fuzzy Syst., 2010, pp. 1–6.
[20] X. Fu, X. J. Zeng, D. Wang, D. Xu, and L. Yang, “Fuzzy system approaches to negotiation pricing decision support,” J. Intell. Fuzzy Syst., vol. 29, pp. 685–699, 2015.
[21] H. A. Hagras, “A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots,” IEEE Trans. Fuzzy Syst., vol. 12, no. 4, pp. 524–539, Aug. 2004.
[22] H. A. Hagras, R. Ramadan, M. Nawito, H. Gabr, M. Zaher, and H. Fahmy, “A fuzzy based hierarchical coordination and control system for a robotic agent team in the robot hockey competition,” in Proc. Int. Conf. Fuzzy Syst., 2010, pp. 1–8.
[23] H. M. Kammoun, I. Kallel, and A. M. Alimi, “Accuracy preserving interpretability by a hierarchical fuzzy multiagent design for mobile robot controller,” in Proc. 1st Int. Conf. Innov. Eng. Syst., 2012, pp. 190–195.
[24] Q. Han, H. Wen, G. Feng, B. Wu, and M. Ren, “Self-nominating trust model based on hierarchical fuzzy systems for peer-to-peer networks,” Peer-to-Peer Netw. Appl., vol. 9, pp. 1020–1030, 2015.
[25] H. He and J. Lawry, “The linguistic attribute hierarchy and its optimisation for classification,” Soft Comput., vol. 18, pp. 1967–1984, 2014.
[26] Y. Jiang, S. Jin, and J. Peng, “Hierarchical fuzzy rule interpolation and its application for hotels location selection,” Int. J. Cogn. Inform. Natural Intell., vol. 10, pp. 55–79, 2016.
[27] J. Lin, “A vibration absorber of smart structures using adaptive networks in hierarchical fuzzy control,” J. Sound Vib., vol. 287, pp. 683–705, 2005.
[28] M. A. Şahin and K. Leblebicikoglu, “A hierarchical fuzzy decision maker for the weapon target assignment,” IFAC Proc. Vol., vol. 44, pp. 8893–8898, 2011.
[29] P. Salgado and P. Garrido, “Image description by hierarchical prioritised fuzzy systems,” in Proc. IEEE Int. Conf. Comput. Cybern., 2009, pp. 173–178.
[30] C. S. Son and G. B. Jeong, “A nutrition evaluation system based on hierarchical fuzzy control,” Int. J. Fuzzy Logic Intell. Syst., vol. 8, pp. 87–93, 2008.
[31] L. Sun and W. Huo, “Adaptive fuzzy control of spacecraft proximity operations using hierarchical fuzzy systems,” IEEE/ASME Trans. Mechatron., vol. 21, no. 3, pp. 1629–1640, Jun. 2016.
[32] Y. Wang, H. Yang, A. Li, and Y. Wang, “Research of photovoltaic and PHEV hybrid management system based on hierarchical fuzzy control,” in Proc. 8th Int. Conf. Elect. Eng. Comput. Sci. Autom. Control, 2011, pp. 1–5.
[33] W. Yu, X. Li, and D. M. Carrillo, “Hierarchical dynamic neural networks for cascade system modeling with application to wastewater treatment,” in Proc. Int. Joint Conf. Neural Netw., 2016, pp. 483–488.
[34] C. W. Chang and C. W. Tao, “A simplified implementation of hierarchical fuzzy systems,” Soft Comput., vol. 23, pp. 4471–4481, 2018.
[35] Y.-C. Hsueh, S.-F. Su and M.-C. Chen, “Decomposed fuzzy systems and their application in direct adaptive fuzzy control,” IEEE Trans. Cybern., vol. 44, no. 10, pp. 1772–1783, Oct. 2014.
[36] V. K. Ojha, V. Snášel, and A. Abraham, “Multiobjective programming for type-2 hierarchical fuzzy inference trees,” IEEE Trans. Fuzzy Syst., vol. 26, no. 2, pp. 915–936, Apr. 2018.
[37] T. R. Razak, J. M. Garibaldi, C. Wagner, A. Pourabdollah, and D. Soria, “Interpretability indices for hierarchical fuzzy systems,” in Proc. IEEE Int. Conf. Fuzzy Syst., 2017, pp. 1–6.

[38] T. Zhou, H. Ishibuchi, and S. Wang, “Stacked-structure-based hierarchical Takagi–Sugeno–Kang fuzzy classification through feature augmentation,” IEEE Trans. Emerg. Topics Comput. Intell., vol. 1, no. 6, pp. 421–436, Dec. 2017.

[39] Y. Zhang, H. Ishibuchi, and S. Wang, “Deep Takagi–Sugeno–Kang fuzzy classifier with shared linguistic fuzzy rules,” IEEE Trans. Fuzzy Syst., vol. 26, no. 3, pp. 1535–1549, Jun. 2018.

[40] L. X. Wang and J. M. Mendel, “Generating fuzzy rules by learning from examples,” IEEE Trans. Syst., Man, Cybern., vol. 22, no. 6, pp. 1414–1427, Nov./Dec. 1992.

[41] L. X. Wang, “The WM method completed: A flexible fuzzy system approach to data mining,” IEEE Trans. Fuzzy Syst., vol. 11, no. 6, pp. 768–782, Dec. 2003.

[42] J. Surowiecki, The Wisdom of Crowds: Why the Many Are Smarter Than the Few. London, U.K.: Abacus, 2005.

[43] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd ed. New York, NY, USA: Springer, 2009.

[44] L. Breiman, “Bagging predictors,” Mach. Learn., vol. 26, pp. 123–140, 1996.

[45] L. Breiman, “Random forests,” Mach. Learn., vol. 45, pp. 5–32, 2001.

[46] Y. Freund and R. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” J. Comput. Syst. Sci., vol. 55, pp. 119–139, 1997.

[47] B. G. Malkiel, A Random Walk Down Wall Street, 10th ed. New York, NY, USA: Norton, 2012.

[48] E. Fama, “Efficient capital markets: A review of theory and empirical work,” J. Finance, vol. 25, pp. 383–417, 1970.

[49] A. W. Lo and A. C. Mackinlay, A Non-Random Walk Down Wall Street. Princeton, NJ, USA: Princeton Univ. Press, 2001.

[50] A. W. Lo and A. C. Mackinlay, “The stock prices do not follow random walks: Evidence from a simple specification test,” Rev. Financial Stud., vol. 1, no. 1, pp. 41–66, 1988.

[51] D. J. Leinweber and A. N. Madhavan, “Three hundred years of stock market manipulations,” J. Investing, vol. 10, no. 2, pp. 7–16, 2001.

[52] D. Rapach and G. Zhou, “Forecasting stock returns,” in Handbook of Economic Forecasting, vol. 2A, G. Elliott and A. Timmermann, Eds., Amsterdam, The Netherlands: Elsevier, 2013, pp. 327–384.

[53] G. Soros, The Alchemy of Finance. Hoboken, NJ, USA: Wiley, 2003.

[54] L. X. Wang, “Dynamical models of stock prices based on technical trading rules—Part III: Application to Hong Kong stocks,” IEEE Trans. Fuzzy Syst., vol. 23, no. 5, pp. 1680–1697, Oct. 2015.

[55] L. X. Wang, “Modeling stock price dynamics with fuzzy opinion networks,” IEEE Trans. Fuzzy Syst., vol. 25, no. 2, pp. 277–301, Apr. 2017.

[56] R. J. Shiller, Irrational Exuberance. New York, NY, USA: Broadway Books, 2006.

[57] L. X. Wang, “Dynamical models of stock prices based on technical trading rules—Part II: Analysis of the model,” IEEE Trans. Fuzzy Syst., vol. 23, no. 4, pp. 1127–1141, Aug. 2015.

[58] L. X. Wang, A Course in Fuzzy Systems and Control. Englewood Cliffs, NJ, USA: Prentice-Hall, 1997.

[59] L. X. Wang and C. Wei, “Approximation accuracy of some neuro-fuzzy approaches,” IEEE Trans. Fuzzy Syst., vol. 8, no. 4, pp. 470–478, Aug. 2000.

[60] L. X. Wang and J. M. Mendel, “Back-propagation fuzzy systems as nonlinear dynamic system identifiers,” in Proc. IEEE Int. Conf. Fuzzy Syst., 1992, pp. 1409–1418.

[61] S. D. Nguyen, S. B. Choi, and T. I. Seo, “Recurrent mechanism and impulse noise filter for establishing ANFIS,” IEEE Trans. Fuzzy Syst., vol. 26, no. 2, pp. 985–997, Apr. 2018.

Li-Xin Wang received the Ph.D. degree in electrical engineering from the Department of Electrical Engineering, University of Southern California, Los Angeles, CA, USA, in 1992. From 1992 to 1993, he was a Postdoctoral Fellow with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, USA. From 1993 to 2007, he was with the faculty of the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong. In 2007, he resigned from his tenured position at HKUST to become an independent researcher and investor in the stock and real estate markets in Hong Kong and China. In Fall 2013, he returned to academia and is now a Professor with the University of Chinese Academy of Sciences, Beijing, China. His research interests include stock dynamics, market microstructure, trading strategies, fuzzy systems, and opinion networks.