Multi-Layer Ensemble Evolving Fuzzy Inference System

Xiaowei Gu

***Abstract*—In order to tackle high-dimensional, complex problems, learning models have to go deeper. In this paper, a novel multi-layer ensemble learning model with first-order evolving fuzzy systems as its building blocks is introduced. The proposed approach can effectively learn from streaming data on a sample-by-sample basis and self-organizes its multi-layered system structure and meta-parameters in a feed-forward, non-iterative manner. Benefiting from its multi-layered distributed representation learning ability, the ensemble system not only demonstrates the state-of-the-art performance on various problems, but also offers high level of system transparency and explainability. Theoretical justifications and experimental investigation show the validity and effectiveness of the proposed concept and general principles.**

***Index Terms*—ensemble model, evolving fuzzy system, multi-layered structure, transparency**

I. Introduction

DEEP neural networks (DNNs, or artificial neural networks, ANNs) are the best-known computational intelligence approaches with a multi-layered architecture. DNNs are powerful representation-learning methods and can automatically learn multiple levels of representations from raw data through a general-purpose learning procedure. In recent years, DNNs have gained enormous popularity among the academic circles and general public thanks to the breakthroughs they have brought in processing images, videos, texts and speeches [1]. A number of publications have reported very promising results using DNNs in various practical applications [2], [3], and some even suggest that DNNs can match the human performance on image recognition tasks [4].

Despite of the great success they have achieved, DNNs are still typical "black-box" systems, computationally cumbersome and data-hungry, and they lack theoretical proof of convergence to the optimal solution [5], [6]. In addition, DNNs are fragile to new patterns and they fail easily due to uncertainties. As DNNs are extremely complicated models with too many hyper-parameters, their reasoning processes are unexplainable to human, concerns on their trustability and reliability also have been raised frequently by research communities and industries [7].

Realizing that the power of DNNs comes from the multi-layered distributed representations, there have been few works [5], [8] published recently attempting to build alternative multi-layer ensemble learning models that can achieve high-level performance competitive to DNNs but with less aforementioned deficiencies. Nonetheless, the transparency and explainability of these models are still limited, and they lack the ability to handle streaming data.

Evolving fuzzy systems (EFSs), as another key branch of computational intelligence, are viewed as a powerful tool for handling complex problems with both measurement and motion uncertainties [9]. EFSs have demonstrated success in a wide variety of real-world applications [10]–[14] and is now an intensively studied area [15]–[19]. Compared with DNNs [1] and alternative mainstream approaches, e.g., support vector machine [20], random forest [21], EFSs perform human-like reasoning and decision-making [6], and their inner system structure is simpler and more transparent. In addition, the majority of EFSs are designed for processing streaming data in a "one pass" manner so that they can efficiently transform data into knowledge presented in a human-interpretable form. Currently, there have been a number of successful EFSs introduced, which include, but are not limited to, DENFIS [22], eTS [23], SAFIS [24], eClass [25], FLEXFIS [26], SOFMLS [15], PANFIS [27], GENEFIS [28], McIT2FIS [29], eT2Class [30] and CNFS [17]. Interested readers may refer to the recent survey [31] for more details regarding EFSs.

On the other hand, one might notice that EFSs usually could not reach the same level performance as DNNs for very complex, high-dimensional problems such as image recognition, due to the simpler system structure and operating mechanism. They also suffer from system obesity and lose transparency on these problems [6]. An effective solution to tackle these issues is to create an ensemble of EFSs to learn from data [32]. Current ensemble EFS models are composed of a number of zero-order EFSs [33]–[36] or first-order EFSs [37], [38] organized in parallel and a fusion module to fuse the processed information. The vast majority of ensemble EFS models are designed for classification purpose. The flat ensemble architecture effectively improves the overall computational efficiency by distributing computations between component learners. However, such architecture could not significantly enhance the system performance in terms of classification accuracy because only the system width is increased. To improve the representation learning ability, an ensemble model should also go deeper.

Inspiring by the multi-layered structure of DNNs, in this paper, a multi-layer ensemble evolving fuzzy inference system (MEEFIS) is introduced. The proposed approach is based on an ensemble of multiple-input multiple-output (MIMO) first-order evolving fuzzy inference systems (EFISs) organized in a multi-layered architecture. Each layer is based on a number of EFISs implemented in parallel, which simultaneously process feature information received from the preceding layer. Each EFIS only handles a selected subset of input features to

X. Gu is with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK, email: x.gu3@lancaster.ac.uk

Source code is available at: https://github.com/Gu-X.

Manuscript received XXXX XX, 2020; revised XXXX XX, 2020.

guarantee the diversity. Outputs from all EFISs at the same layer are cascaded together to form the inputs of the next layer. Unlike DNNs, MEEFIS is capable of self-developing and self-evolving its system structure and meta-parameters on the fly from streaming data to follow the possible shifts and/or drifts in the data patterns. More importantly, MEEFIS maintains the advantages of high transparency and human-interpretability that are typical for EFSs and, at the same time, it can achieve very high performance on various problems thanks to the multi-layered distributed representation learning ability.

Key features of the proposed approach include: (1) a self-evolving multi-layered parallel computation structure; (2) better representation learning ability benefited from its deeper structure, and; (3) stronger capability to handle large-scale, nonstationary, complex problems. Comparing with existing ensemble EFS models [33]–[38], the proposed approach has a deeper structure composed of multiple layers of interconnected EFISs, thus, it is capable of learning multiple levels of representations from data and has stronger approximation ability to nonlinear problems. In addition, it is more generic and applicable to both regression and classification tasks.

The remainder of this paper is organized as follows. Section II presents the technical details of the proposed ensemble approach. To be more specific, the ensemble structure is described by equations (1)-(4) in Section II.A, and the self-evolving process of each individual EFIS within the ensemble is described by equations (5)-(19) in Section II.B. Numerical experiments are presented in Section III as a proof of concept. This paper is concluded by Section IV.

II. Proposed Approach

In this section, the general architecture and learning process of MEEFIS are described in detail. Computational complexity analysis is provided in Supplementary Material.

A. General architecture

The general architecture of MEEFIS is depicted in Fig. 1. The proposed system is an ensemble of many EFISs that are connected and arranged in layers. Inputs are processed layer-by-layer within the ensemble system until the final layer generates outputs and, the structure and meta-parameters of each EFIS are dynamically self-updating on a sample-by-sample basis during the learning process from streaming data.

For each new input sample, a number of sub-samples are firstly created by sliding window scanning. Then, these sub-samples are fed to the respective EFISs in the input layer (the first layer) to perform system updating in parallel. The number of EFISs needed for a particular layer is determined by the sliding window scanning. This means that if the sliding window splits input features into M subsets, M EFISs will be implemented in this layer to process input information simultaneously. The outputs (namely, processed feature information) of these EFISs are integrated together and used as the input of the second layer. The same procedure repeats at the second layer and layers after. The final EFIS at the last layer plays as the decision-maker of the ensemble model and produces the final outcome based on the input features from the previous layer. Note that the ensemble architecture of the proposed approach needs to be determined at the beginning, which includes the number of layers and parameter settings of the sliding window used for each layer (window size and step size) .

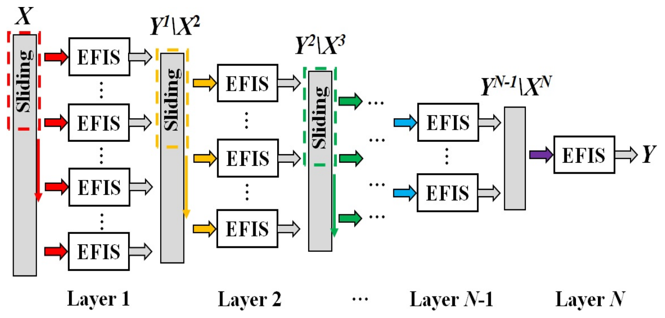

Fig. 1: General architecture of MEEFIS.

The inner structure of EFIS is given by Fig. 2. The system is composed of L linear models identified through its inherent learning process. These linear models are described by prototype-based fuzzy rules in the following form [16], [39]:

$$\boldsymbol{R}_i: \quad IF\ (\boldsymbol{x} \sim \boldsymbol{p}_i)\ THEN\ (\boldsymbol{y}_i = \bar{\boldsymbol{x}}^T \mathbf{a}_i); \tag{1}$$

where $i = 1, 2, ..., L$; L is the number of linear models (fuzzy rules) within the system; $\boldsymbol{x} = [x_1, x_2, ...x_K]^T$ is the K dimensional input; $\bar{\boldsymbol{x}} = [1, \boldsymbol{x}^T]^T$; $\boldsymbol{p}_i$ is the antecedent part (prototype) of $\boldsymbol{R}_i$; $\mathbf{a}_i = [\boldsymbol{a}_{i,1}, \boldsymbol{a}_{i,2}, ..., \boldsymbol{a}_{i,W}]$ is the consequent parameter matrix and there is $\boldsymbol{a}_{i,j} = [a_{i,j,0}, a_{i,j,1}, ..., a_{i,j,K}]^T$ $(j = 1, 2, ..., W)$; $\boldsymbol{y}_i = [y_1, y_2, ...y_W]$ is the $1 \times W$ dimensional output of $\boldsymbol{R}_i$.

The general mathematical model of EFIS is given by [16]:

$$\boldsymbol{y} = f(\boldsymbol{x}) = \sum_{i=1}^{L} \lambda_i \boldsymbol{y}_i, \tag{2}$$

where λ_i is the firing strength of the i^{th} fuzzy rule, $\boldsymbol{R}_i$.

Based on equation (2), the input-output relation of a particular layer (assuming the n^{th} layer) of MEEFIE is formulated as follows.

$$\boldsymbol{Y}^n = F^n(\boldsymbol{X}^n) = [f_1^n(\boldsymbol{x}_1^n), ..., f_M^n(\boldsymbol{x}_M^n)]^T, \tag{3}$$

where $\boldsymbol{X}^n$ and $\boldsymbol{Y}^n$ are the respective input and output vectors of the n^{th} layer; M is the number of EFISs implemented in this layer; the dimensionality of $\boldsymbol{Y}^n$ is $MW \times 1$; $\boldsymbol{x}_1^n, ..., \boldsymbol{x}_M^n$ are the M sub-samples split from $\boldsymbol{X}^n$ by sliding window, and; $f_j^n(\cdot)$ represents the j^{th} EFIS model at this layer $(j = 1, 2, ..., M)$.

Consider that the output of the n^{th} layer serves as the input of the $(n+1)^{th}$ layer (namely, $\boldsymbol{X}^{n+1} \leftarrow \boldsymbol{Y}^n$), the overall ensemble system can be mathematically modeled by the following composite function:

$$\boldsymbol{Y} = (F^N \circ F^{N-1} \circ ... \circ F^1)(\boldsymbol{X}), \tag{4}$$

where $\boldsymbol{X}$ is the $Q \times 1$ dimensional raw input of MEEFIS, and; $\boldsymbol{Y}$ is the $W \times 1$ dimensional final output (see Fig. 1).

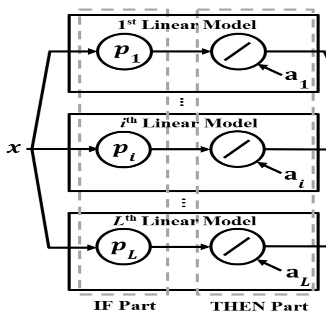

Fig. 2: Inner structure of EFIS [16].

B. Learning process

As stated in Section II.A, all EFISs within the ensemble system work collaboratively to process input feature information and generate the final output. Since every EFIS follows the exact same algorithmic procedure for system identification, only the learning process of a single EFIS is described below. The same principles can be applied to all EFISs within the ensemble system. Note that EFIS differs from its predecessors, eTS [23] and ALMMo [16] (which itself is a more advanced version of eTS) from the following aspects:

1) The structure updating mechanism is simplified to better preserve the learned knowledge, new fuzzy rules will not replace the previously identified ones;
2) The consequent parameter updating mechanism is optimized to reduce the computational complexity, only the more activated fuzzy rules are updated at each learning cycle.

Stage 1. System initialization

The system is initialized by the first input vector, $\boldsymbol{x}_k$ ($k=1$) with its global meta-parameters set as:

$$\boldsymbol{\mu} \leftarrow \boldsymbol{x}_k; \quad \chi \leftarrow ||\boldsymbol{x}_k||^2; \tag{5}$$

where $||\boldsymbol{x}_k||$ denotes the Euclidean norm of $\boldsymbol{x}_k$; $\boldsymbol{\mu}$ and χ are the global means of input vectors and their squared Euclidean norms, respectively.

The first fuzzy rule, $\boldsymbol{R}_L$ ($L=1$) is initialized in the form of equation (1) with its premise and consequent parameters defined as:

$$\boldsymbol{p}_L \leftarrow \boldsymbol{x}_L; \quad \mathbf{a}_L \leftarrow \mathbf{0}_{(K+1)\times W}; \quad \boldsymbol{\Theta}_L \leftarrow \Omega_o \mathbf{I}_{(K+1)\times(K+1)}; \tag{6}$$

where $\boldsymbol{\Theta}_L$ is the covarience matrix of $\boldsymbol{R}_L$; $\mathbf{I}_{(K+1)\times(K+1)}$ is the $(K+1)\times(K+1)$ dimensional identity matrix. Note that, Ω_o is an externally controlled parameter for initializing $\boldsymbol{\Theta}_L$, which is standard for recursive least squares algorithms [40]. In this paper, $\Omega_o = 100$.

Meta-parameters of the cluster, $\mathbf{C}_L$ formed around $\boldsymbol{p}_L$ resembling Voronoi tessellation are initialized as:

$$\mathbf{C}_L \leftarrow \{\boldsymbol{x}_k\}; \quad X_L \leftarrow ||\boldsymbol{x}_k||^2; \quad S_L \leftarrow 1. \tag{7}$$

where S_L is the cardinality (number of members) of $\mathbf{C}_L$.

Stage 2. Output generation

Each time the system receives a new input vector, $\boldsymbol{x}_k$ ($k \leftarrow k+1$), the local density, $D_i(\boldsymbol{x}_k)$ of $\boldsymbol{x}_k$ at each cluster is evaluated firstly using equation (8) ($i=1,2,...,L$):

$$D_i(\boldsymbol{x}_k) = e^{-\frac{||\boldsymbol{x}_k - \hat{\boldsymbol{p}}_i||^2}{\hat{X}_i - ||\hat{\boldsymbol{p}}_i||^2}}. \tag{8}$$

where $\hat{\boldsymbol{p}}_i$ and $\hat{X}_i$ are calculated by equation (9), respectively [16]:

$$\hat{\boldsymbol{p}}_i = \frac{S_i \boldsymbol{p}_i + \boldsymbol{x}_k}{S_i+1}; \quad \hat{X}_i = \frac{S_i X_i + ||\boldsymbol{x}_k||^2}{S_i+1}. \tag{9}$$

The firing strength of each fuzzy rule, defined as the normalized local density, is calculated by equation (10) ($i=1,2,...,L$) [16]:

$$\lambda_{i,k} = \frac{D_i(\boldsymbol{x}_k)}{\sum_{j=1}^{L} D_j(\boldsymbol{x}_k)}. \tag{10}$$

Then, the system output, $\hat{\boldsymbol{y}}_k$ is produced by equation (2).

Stage 3. Structure updating

In this stage, the global meta-parameters, $\boldsymbol{\mu}$ and χ are updated as [16]:

$$\boldsymbol{\mu} \leftarrow \frac{(k-1)\boldsymbol{\mu} + \boldsymbol{x}_k}{k}; \quad \chi \leftarrow \frac{(k-1)\chi + ||\boldsymbol{x}_k||^2}{k}. \tag{11}$$

To testify the potential of $\boldsymbol{x}_k$ to initialize a new fuzzy rule, the global density values at $\boldsymbol{x}_k$ and $\boldsymbol{p}_i$ ($i=1,2,...,L$) are calculated firstly by equation (12):

$$D(\boldsymbol{z}) = e^{-\frac{||\boldsymbol{z}-\boldsymbol{\mu}||^2}{\chi - ||\boldsymbol{\mu}||^2}}; \quad \boldsymbol{z} \in \{\boldsymbol{x}_k, \boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_L\}. \tag{12}$$

Then, **Condition 1** is checked [16], [41]:

$$\begin{aligned} \textbf{Cond. 1}: \ & \textit{If } (D(\boldsymbol{x}_k) < \min_{i=1,2,\ldots,L}(D(\boldsymbol{p}_i))) \\ & \textit{Or } (D(\boldsymbol{x}_k) > \max_{i=1,2,\ldots,L}(D(\boldsymbol{p}_i))) \\ & \textit{And } (D_i(\boldsymbol{x}_k) \le D_0) \\ & \textit{Then } (\boldsymbol{x}_k \textit{ becomes a new prototype}) \end{aligned} \tag{13}$$

where $D_0 = e^{-\frac{1}{4}}$. If **Condition 1** is satisfied, a new fuzzy rule $\boldsymbol{R}_L$ ($L \leftarrow L+1$) is added to the rule base with $\boldsymbol{x}_k$ as its antecedent part, namely, $\boldsymbol{p}_L \leftarrow \boldsymbol{x}_k$. Its consequent parameters are initialized by equation (14) [16]:

$$\mathbf{a}_L \leftarrow \frac{1}{L-1}\sum_{i=1}^{L-1} \mathbf{a}_i; \quad \boldsymbol{\Theta}_L \leftarrow \Omega_o \mathbf{I}_{(W+1)\times(W+1)}. \tag{14}$$

The meta-parameter of the new cluster, $\mathbf{C}_L$, which is associated with $\boldsymbol{p}_L$, are set by equation (7).

Otherwise, $\boldsymbol{x}_k$ is used for updating the antecedent part of $\boldsymbol{R}_{n*}$ and meta-parameters of $\mathbf{C}_{n*}$ [16]:

$$\begin{aligned} & \mathbf{C}_{n*} \leftarrow \mathbf{C}_{n*} \cup \{\boldsymbol{x}_k\}; \quad \boldsymbol{p}_{n*} \leftarrow \frac{S_{n*}\boldsymbol{p}_{n*} + \boldsymbol{x}_k}{S_{n*}+1}; \\ & X_{n*} \leftarrow \frac{S_{n*}X_{n*} + ||\boldsymbol{x}_k||^2}{S_{n*}+1}; \quad S_{n*} \leftarrow S_{n*} + 1; \end{aligned} \tag{15}$$

where $n* = \underset{j=1,2,\ldots,L}{\operatorname{argmin}} (||\boldsymbol{p}_j - \boldsymbol{x}_k||^2)$.

For the fuzzy rules and the associated clusters that are not updated, their meta-parameters stay the same for the next processing cycle.

Stage 4. Fuzzy rule quality monitoring

To improve computational efficiency and reduce system complexity, the fuzzy rules and associated clusters that represent less dominant data patterns and contribute less to the overall system outputs are removed from the rule base. **Condition 2** is utilized to identify such fuzzy rules [16], [41]:

$$\textbf{Cond. 2}: \; \textit{If } (\eta_{i,k} < \eta_0) \textit{ Then } (\boldsymbol{R}_i \textit{ and } \mathbf{C}_i \textit{ are removed}) \tag{16}$$

where $\eta_{i,k}$ $(i = 1, 2, ..., L)$ is the utility of $\boldsymbol{R}_i$ $(i = 1, 2, ..., L)$ at the k^{th} time instance defined by equation (17); η_0 is another externally controlled parameter for quality monitoring, here $\eta_0 = 0.1$ following the setting of [16], [41].

$$\eta_{i,k} = \frac{1}{k - I_i} \sum_{j=I_i}^{k} \lambda_{i,j}; \tag{17}$$

where I_i denotes the time instance at which $\boldsymbol{R}_i$ was initialized; $\lambda_{i,j}$ is the firing strength of $\boldsymbol{R}_i$ calculated by equation (10) at the j^{th} time instance.

Stage 5. Consequent parameter updating

After the structure updating, the consequent parameters of the system are going to be updated in this stage. The system only updates the fuzzy rules that satisfy **Condition 3**:

$$\textbf{Cond. 3}: \; \textit{If } (D_i(\boldsymbol{x}_k) \geq D_1) \textit{ Then } (\boldsymbol{R}_i \textit{ is updated}) \tag{18}$$

where $i = 1, 2, ..., L$; $D_1 = e^{-4}$. If $\mathbf{R}_i$ satisfies **Condition 3**, its prototype is closer to $\boldsymbol{x}_k$ and its corresponding consequent parameters ($\mathbf{a}_i$ and $\boldsymbol{\Theta}_i$) are updated by the fuzzily weighted recursive least square algorithm [23]:

$$\begin{aligned} \boldsymbol{\Theta}_i &\leftarrow \boldsymbol{\Theta}_i - \frac{\lambda_i \boldsymbol{\Theta}_i \bar{\boldsymbol{x}}_k \bar{\boldsymbol{x}}_k^T \boldsymbol{\Theta}_i}{1 + \lambda_{i,k} \bar{\boldsymbol{x}}_k^T \boldsymbol{\Theta}_i \bar{\boldsymbol{x}}_k}; \\ \mathbf{a}_i &\leftarrow \mathbf{a}_i + \lambda_i \boldsymbol{\Theta}_i \bar{\boldsymbol{x}}_k (\boldsymbol{y}_k - \bar{\boldsymbol{x}}_k^T \mathbf{a}_i). \end{aligned} \tag{19}$$

Instead of updating the whole rule base [16], which is computationally expensive and does not necessarily guarantee the optimal solution [41], updating the fuzzy rules that are more activated by the current input can significantly reduce the computational burden, especially, when the system identifies a large number of fuzzy rules from data. This effectively strengthens the capability of EFIS to handle large-scale, complex problems.

Once the system structure and meta-parameters are updated to the latest, the system goes back to **Stage 2** and processes the next input vector. To summarize, the main algorithmic procedure of the system identification process is presented in the form of pseudo-code given by **Algorithm 1**.

Algorithm 1 EFIS identification.

```
while (new input, $\boldsymbol{x}_k$ is available) do
  if ($k = 1$) then
    $L \leftarrow 1$;
    initialize $\boldsymbol{\mu}$, $\chi$, $\boldsymbol{R}_L$ and $\mathbf{C}_L$ by (5)-(7);
  else
    generate $\hat{\boldsymbol{y}}_k$ by (2);
    update $\boldsymbol{\mu}$ and $\chi$ by (11);
    $n* = \underset{j=1,2,...,L}{\text{argmin}} (||\boldsymbol{p}_j - \boldsymbol{x}_k||^2)$;
    if ($\boldsymbol{x}_k$ satisfies Condition 1) then
      $L \leftarrow L + 1$;
      initialize $\boldsymbol{R}_L$ and $\mathbf{C}_L$ by (14) and (7);
    else
      update $\boldsymbol{R}_{n*}$ and $\mathbf{C}_{n*}$ by (15);
    end if
    for $i = 1$ to $L$ do
      if ($\boldsymbol{R}_i$ satisfies Condition 2) then
        remove $\boldsymbol{R}_i$ and $\mathbf{C}_i$;
        $L \leftarrow L - 1$;
      end if
    end for
    for $i = 1$ to $L$ do
      if ($\boldsymbol{R}_i$ satisfies Condition 3) then
        update $\mathbf{a}_i$ and $\boldsymbol{\Theta}_i$ by (19);
      end if
    end for
  end if
end while
```

III. Experimental Investigation

A. Configuration

In this section, experimental studies are conducted for validating the proposed concept and general principles. The performance of MEEFIS is evaluated on a wide range of benchmark problems and compared with the state-of-the-art approaches. Details of comparative algorithms are tabulated in Supplementary Table. I. The algorithms were developed using MATLAB2018a, and numerical experiments were performed on a desktop with dual core i7 processor 3.60GHz×2 and 16.0GB RAM.

Since the goal is to demonstrate that a multi-layered ensemble of EFISs can outperform the state-of-the-art approaches with a general experimental setting across a variety of tasks, MEEFIS is using the same three-layered architecture in all numerical examples. A sliding window with window size of $\lceil \frac{Q}{2} \rceil$ and step size of $\lfloor \frac{Q}{4} \rfloor$ is employed by the first layer to split each input data sample, $\boldsymbol{X} = [x_1, x_2, ..., x_Q]^T$ into four sub-samples, namely, $\boldsymbol{x}_1^1 = [x_1, x_2, ..., x_{\lceil \frac{Q}{2} \rceil}]^T$, $\boldsymbol{x}_2^1 = [x_{1+\lfloor \frac{Q}{4} \rfloor}, x_{2+\lfloor \frac{Q}{4} \rfloor}, ..., x_{\lceil \frac{Q}{2} \rceil + \lfloor \frac{Q}{4} \rfloor}]^T$, $\boldsymbol{x}_3^1 = [x_{1+2\lfloor \frac{Q}{4} \rfloor}, x_{2+2\lfloor \frac{Q}{4} \rfloor}, ..., x_{\lceil \frac{Q}{2} \rceil + 2\lfloor \frac{Q}{4} \rfloor}]^T$ and $\boldsymbol{x}_4^1 = [x_{1+3\lfloor \frac{Q}{4} \rfloor}, ..., x_Q, x_1, ..., x_{Q-3\lfloor \frac{Q}{4} \rfloor}]^T$. The second layer uses a sliding window with window size of $3W$ and step size of W to divide the input features received from the previous layer, $\boldsymbol{Y}^1 = [\boldsymbol{y}_1^1, \boldsymbol{y}_2^1, \boldsymbol{y}_3^1, \boldsymbol{y}_4^1]^T$ into four subsets, namely, $\boldsymbol{x}_1^2 = [\boldsymbol{y}_1^1, \boldsymbol{y}_2^1, \boldsymbol{y}_3^1]^T$, $\boldsymbol{x}_2^2 = [\boldsymbol{y}_2^1, \boldsymbol{y}_3^1, \boldsymbol{y}_4^1]^T$, $\boldsymbol{x}_3^2 = [\boldsymbol{y}_3^1, \boldsymbol{y}_4^1, \boldsymbol{y}_1^1]^T$ and $\boldsymbol{x}_4^2 = [\boldsymbol{y}_4^1, \boldsymbol{y}_1^1, \boldsymbol{y}_2^1]^T$, where Q and W are the dimensionality of system inputs and outputs, respectively. Thus, the three-layer ensemble model is composed of nine EFISs in total, namely, four in the first layer, another four in the second layer and one in the last layer. The structure of MEEFIS for numerical studies is also given in Supplementary Fig. 1 for better illustration. Note that the model used in experiments is only for conceptual demonstration, the architecture of MEEFIS is, in fact, very flexible. In practice, one may explore different ensemble architectures to find the best-performing models, but this is out of the scope of this paper.

B. Results

1) Real-world regression problems: In the first example, five real-world benchmark problems, namely, (1) Autos, (2) Autompg, (3) Triazines, (4) Delta Ailerons and (5) California Housing, are used for performance evaluation. Details of the five datasets are given by Supplementary Table II. All the input and output features are normalized in advance to the range of $[0, 1]$ as the common practice. During each experiment, MEEFIS learns from the training set on a sample-by-sample basis and then is tested on the validation set. Experimental results in terms of root mean square error (*RMSE*) of the predictions and the total number of fuzzy rules within the ensemble system (*#rules*) obtained by MEEFIS are reported in Table I. The average number of fuzzy rules per component learner (EFIS) is presented in the bracket. The results obtained by its single-model version, namely, EFIS are also reported in the same table as baseline. It can be seen from Table I that the prediction error of EFIS is generally lower than MEFIS on the smaller-scale problems, namely, Autompg, Triazines and Delta Ailerons. This is because MEEFIS has more trainable parameters compared with EFIS and, thus, more training samples are needed for MEEFIS in order to fully exploit its strength.

To verify this, in the following experiments, MEEFIS is trained on the same training set for 5 and 10 epochs, respectively, before testing, and the obtained results are reported in Table I as well. Note that during the experiments, the structure of MEEFIS is fixed after the first training epoch and only the premise and consequent parameters are updated in the later training epochs to avoid overfitting. For comparison, the results obtained by EFIS following the same experimental protocols are also reported in the same table. It is shown by Table I that the prediction errors by MEEFIS on the five datasets reduced by 14.8%, 32.2%, 89.0%, 3.1% and 17.6%, respectively, after 5 epochs. In contrast, the prediction errors on the five datasets by EFIS only reduced by 4.7%, 16.7%, 14.3%, 2.5% and 13.3%, respectively. Moreover, the prediction performance of MEEFIS is much higher than EFIS on Autos, Autompg, Triazines and California Housing datasets after 10 training epochs, which shows the stronger learning ability of the ensemble system.

In the next numerical example, performances of the proposed approach with different ensemble architectures are investigated on the five benchmark problems. For clarity, the architecture for experiments as described in Section III.A is re-denoted as *Arch. 1*. *Arch. 2* and *Arch. 3* follow the same framework as *Arch. 1*. However, the window size of the sliding window used by the second layer of *Arch. 2* and *Arch. 3* is modified to $2W$ and W, respectively. Additionally, a two-layer ensemble architecture, denoted by *Arch. 4* and a four-layer ensemble architecture, denoted by *Arch. 5* are also involved for experimental investigation. Both *Arch. 4* and *Arch. 5* are given in Supplementary Figs. 2(a) and 2(b) for better demonstration. Specifically, the window size of both sliding windows employed by the second and third layers of *Arch. 5* is set as $3W$. The obtained numerical results by the five different architectures are reported in Supplementary Table III. It can be observed from this table that given the same ensemble architecture, in general, a larger sliding window size can lead to better prediction performance because each EFIS at the second layer is able to receive and utilize more information from the previous layer. Another interesting phenomenon worth to be noticed is that despite of higher system complexity, a deeper ensemble architecture generally performs better comparing with shallow ensemble architectures. This also validates the proposed concept and general principles.

The performance of MEEFIS is further compared with popular EFSs such as OS-Fuzzy-ELM [42], CEFNS [17], ESAFIS [43], eTS [23], ALMMo [16] and its optimized version, SB-ALMMo [41] on the five benchmark datasets. Performance comparison based on $RMSE$, training time (t_{exe}, in seconds) and $\#rules$ is presented in Table II. In addition, the proposed approach is tested on the widely-used chaotic Mackey–Glass time series problem (details of this problem are given in Supplementary Material). Numerical results obtained by MEEFIS as well as other comparative algorithms are reported in Table III in terms of non-dimensional error index ($NDEI$), t_{exe} and *#rules*. The definition of $NDEI$ is given by equation (20). Note that MEEFIS is trained for 10 epochs in the experiments. The average number of fuzzy rules and training time per component learner is presented in the bracket. It is clearly shown in Tables II and III that MEEFIS outperforms all the comparative EFSs in terms of prediction accuracy. Moreover, it is worth to be noticed that the computational efficiency and system complexity of each component learner is, in fact, on the same level with other comparative EFSs.

$$NDEI = \sqrt{\frac{\sum_{j=1}^{k}(\hat{y}_j - y_j)^2}{k\sigma^2}} \tag{20}$$

where $\hat{y}_j$ and y_j are the system output and the corresponding true value at the j^{th} time instance ($j = 1, 2, ..., k$), respectively; σ is the standard deviation of the true value.

2) Nonstationary regression problems: Nonstationary regression problems usually have much more frequent and intensive changes in data patterns and are very useful for evaluating the performance of EFSs in complex real-world application scenarios. In this paper, the following two real-world financial data prediction problems are considered for performance evaluation: (1) QuantQuote second resolution market dataset, and; (2) S&P 500 closing price prediction dataset. Details of the two datasets and experimental protocols are given in Supplementary Material. The step-by-step online perdition results by MEEFIS and alternative EFSs in terms of $NDEI$ and *#rules* are reported in Table IV and V. The average number of fuzzy rules per component learner (EFIS) of MEEFIS is presented in the bracket. Both tables clearly show that MEEFIS can efficiently and effectively react to the changing data patterns in the data stream and make accurate prediction surpassing the alternatives.

3) Benchmark classification problems: This example studies the performance of MEEFIS for classification tasks. The following popular benchmark datasets are involved for experiments, namely, (1) Letter Recognition (LR), (2) Multiple Features (MF), (3) Optical Recognition of Handwritten Digits (ORHD), (4) Pen-based Recognition of Handwritten Digits (PRHD) and (5) Wilt (WI). Details of the five classification problems are summarized in Supplementary Table IV. For ORHD, PRHD and WI datasets, the order of training samples

TABLE I: PERFORMANCE DEMONSTRATION ON BENCHMARK REGRESSION PROBLEMS

Algorithm	Epoch	Autos		Autompg		Triazines		Delta Ailerons		California Housing	
		$RMSE$	$\#rules$	$RMSE$	$\#rules$	$RMSE$	$\#rules$	$RMSE$	$\#rules$	$RMSE$	$\#rules$
MEEFIS	1	0.0405	57 (6.3)	0.0932	60 (6.7)	0.0263	67 (7.4)	0.0516	58 (6.4)	0.0743	78 (8.7)
	5	0.0345		0.0632		0.0029		0.0500		0.0612	
	10	0.0334		0.0609		0.0018		0.0499		0.0607	
EFIS	1	0.0445	7	0.0831	6	0.0035	7	0.0513	8	0.0805	10
	5	0.0424		0.0692		0.0030		0.0500		0.0698	
	10	0.0430		0.0686		0.0026		0.0499		0.0696	

TABLE II: PERFORMANCE COMPARISON ON BENCHMARK REGRESSION PROBLEMS

Dataset	Algorithm	$RMSE$	t_{exe}	$\#rules$
Autos	MEEFIS	**0.0334**	1.24 (0.14)	57 (6.3)
	OS-Fuzzy-ELM	0.0595	0.03	2
	CEFNS	0.0666	0.02	2
	ESAFIS	0.0604	0.22	3
	eTS	0.0535	0.22	3
	ALMMo	0.0565	0.10	8
	SB-ALMMo	0.0456	0.28	4
Autompg	MEEFIS	**0.0609**	2.29 (0.26)	60 (6.7)
	OS-Fuzzy-ELM	0.0765	0.05	3
	CEFNS	0.0750	/	2
	ESAFIS	0.0731	0.05	33
	eTS	0.0864	2.01	6
	ALMMo	0.0842	0.22	9
	SB-ALMMo	0.0934	0.40	3
Triazines	MEEFIS	**0.0018**	1.56 (0.17)	67 (7.4)
	OS-Fuzzy-ELM	0.0100	2.46	6
	CEFNS	0.0452	0.02	6
	ESAFIS	0.0331	24.44	19
	eTS	0.0179	3.52	9
	ALMMo	0.0078	0.19	7
	SB-ALMMo	0.0022	2.34	3
Delta Ailerons	MEEFIS	**0.0499**	28.41 (3.16)	58 (6.4)
	OS-Fuzzy-ELM	0.0507	0.46	3
	CEFNS	0.0502	0.34	3
	ESAFIS	0.0506	12.39	13
	eTS	0.0513	2.14	4
	ALMMo	0.0513	0.58	10
	SB-ALMMo	0.0512	1.22	4
California Housing	MEEFIS	**0.0607**	97.22 (10.80)	78 (8.7)
	OS-Fuzzy-ELM	0.1302	6.73	5
	CEFNS	0.0878	1.54	2
	ESAFIS	0.0892	30.23	6
	eTS	0.0772	7.61	3
	ALMMo	0.0782	1.24	10
	SB-ALMMo	0.0771	4.70	5

TABLE III: PERFORMANCE COMPARISON ON MACKEY-GLASS TIME SERIES PROBLEM

Algorithm	$NDEI$	t_{exe}	$\#rules$
MEEFIS	**0.1392**	2.92	74 (8.2)
OS-Fuzzy-ELM	0.2991	0.93	5
CEFNS	0.2635	0.44	5
SAFIS [24]	0.38	/	6
ESAFIS	0.2955	5.83	6
eTS	0.3805	/	9
ALMMo	0.4437	0.46	7
SB-ALMMo	0.4402	0.86	4
GENEFIS (C) [28]	0.280	4.46	19
GENEFIS (B) [28]	0.339	3.02	9
LEOA [44]	0.2480	144.78	42

TABLE IV: PERFORMANCE COMPARISON ON QUANTQUOTE DATASET

Algorithm	$y = x_{k+8,2}$		$y = x_{k+24,2}$	
	$NDEI$	*#rules*	$NDEI$	*#rules*
MEEFIS	**0.129**	67 (7.4)	**0.165**	79 (8.8)
EFIS	0.140	6	0.198	6
DENFIS [22]	1.598	12	1.582	12
eTS	0.183	6	0.271	7
SAFIS	0.554	20	0.779	14
ESAFIS	0.235	3	0.292	3
ALMMo	0.146	6	0.204	6

TABLE V: PERFORMANCE COMPARISON ON S&P 500 DATASET

Algorithm	$NDEI$	*#rules*
MEEFIS	**0.0124**	85 (9.4)
EFIS	0.0147	15
ALMMo	0.0149	7
PANFIS [27]	0.09	4
GENEFIS [28]	0.07	2
LEOA [44]	0.1229	52
SEFS [19]	0.0182	2
EFS-SLAT [45]	0.0156	23

is randomly scrambled. For LR and MF datasets, all the data samples are firstly split into 10 folds evenly, and five of the 10 folds are randomly selected to train the algorithms and the remaining folds are used for testing. The statistical performance of the proposed MEEFIS (classification accuracy, Acc and t_{exe}) on MF, ORHD and PRHD datasets are tabulated in Table VI after 10 Monte-Carlo experiments, and the results on LR and WI datasets are given in Supplementary Table V. Note that in this example, MEEFIS is trained for 10 epochs. As for classification problems, fuzzy rules that represent less dominant data patterns may play a significant role in classification, the same experiments are repeated by setting $\eta_0 = 0$, which means that each EFIS within the ensemble system will not forget any knowledge gained during training. A wide variety of state-of-the-art classification algorithms including EFSs, ANNs, popular ensemble approaches such as AdaBoost (AdaBo) [46] and random forest (RanFor) [21] are used for comparison under the same experimental protocol, and their results are reported in the same tables.

As it is clearly shown in Table VI and Supplementary Table V, MEEFIS outperforms alternative EFSs in terms of testing accuracy, and is on par with the best-performing classification algorithms on all five problems. Moreover, thanks to **Condition 3**, the computational complexity of MEEFIS is not

TABLE VI: PERFORMANCE COMPARISON ON BENCHMARK CLASSIFICATION PROBLEMS

Algorithm	**MF**		**ORHD**		**PRHD**	
	Acc	t_{exe}	Acc	t_{exe}	Acc	t_{exe}
MEEFIS ($\eta_0 = 0.1$)	**0.9817±0.0033**	246.06±16.86	0.9462±0.0031	83.31±1.21	0.9176±0.0037	137.43±7.10
MEEFIS ($\eta_0 = 0$)	0.9769±0.0041	287.77±32.91	**0.9604±0.0019**	108.34±5.34	**0.9567±0.0044**	159.72±9.52
AdaBo	0.9764±0.0064	7.46±0.45	0.9321±0.0000	1.49±0.12	0.9240±0.0007	2.54±0.82
RanFor	0.9690±0.0109	15.44±0.43	0.9331±0.0020	5.58±0.35	0.9495±0.0011	6.29±0.84
MLP	0.8648±0.0331	0.53±0.25	0.9243±0.0077	0.70±0.24	0.9177±0.0180	0.92±0.17
LSTM	0.6529±0.0116	64.46±2.90	0.7726±0.0544	5.61±0.91	0.8218±0.0280	34.29±7.82
ESAFIS	0.5938±0.1896	278.62±58.83	0.9538±0.0067	32.24±10.36	0.9197±0.0134	38.80±5.93
eClass0	0.7990±0.0113	2.22±0.27	0.8937±0.0000	0.75±0.07	0.7630±0.0001	0.73±0.08
ALMMo	0.9696±0.0076	108.74±13.92	0.9168±0.0089	4.67±0.23	0.8279±0.0073	2.26±0.29

TABLE VII: TESTING ACCURACY COMPARISON ON IMAGE RECOGNITION PROBLEMS

Algorithm	**MNIST (#Training Images)**						**Fashion MNIST (#Training Images)**					
	10000	**20000**	**30000**	**40000**	**50000**	**60000**	**10000**	**20000**	**30000**	**40000**	**50000**	**60000**
MEEFIS ($\eta_0 = 0.1$)	0.9742	0.9745	0.9746	0.9748	0.9751	0.9754	0.8765	0.8786	0.8797	0.8801	0.8799	0.8810
MEEFIS ($\eta_0 = 0$)	**0.9756**	**0.9765**	**0.9773**	**0.9775**	**0.9778**	**0.9788**	**0.8834**	**0.8881**	**0.8896**	**0.8905**	**0.8913**	**0.8916**
AdaBo	0.9446	0.9447	0.9446	0.9446	0.9454	0.9451	0.8178	0.8187	0.8190	0.8196	0.8199	0.8192
RanFor	0.9477	0.9559	0.9602	0.9606	0.9626	0.9631	0.8638	0.8733	0.8778	0.8816	0.8847	0.8872
MLP	0.4543	0.4717	0.6232	0.5848	0.6351	0.5650	0.6304	0.7451	0.8028	0.7739	0.8045	0.8318
eClass0	0.8470	0.8556	0.8512	0.8603	0.8551	0.8500	0.7289	0.7303	0.7147	0.7497	0.7233	0.7167
ALMMo	0.9676	0.9695	0.9707	0.9715	0.9722	0.9723	0.8666	0.8694	0.8696	0.8699	0.8702	0.8705

significantly increased without removing the fuzzy rules that contribute less to the outputs during the training stage.

4) Image recognition problems: In the final example, two widely-used image recognition datasets, namely, MNIST and Fashion MNIST, are used for performance evaluation. Details of both datasets are given in Supplementary Material. During the experiments, MEEFIS (with $\eta_0 = 0.1$ and $\eta_0 = 0$) and other comparative algorithms are trained on the training sets of different sizes (10000, 20000, 30000, 40000, 50000 and 60000 images), and then, tested on the validation sets. The average testing accuracy rates by the involved algorithms are reported in Table VII after 10 Monte-Carlo experiments. It can be seen from Table VII that MEEFIS outperforms all competitors in terms of average classification accuracy rates on both benchmark image sets.

C. Discussion

Numerical examples presented in this section demonstrate that MEEFIS outperforms the state-of-the-art learning approaches on various prediction and classification tasks. Thanks to its multi-layered ensemble architecture, MEEFIS can learn multi-layered distributed representations from data and achieve much better results than alternative single-model EFSs on large-scale, complex problems. In addition, it is also justified by numerical results that the performance of MEEFIS can be further improved by increasing the system depth.

However, as all the experiments were performed on a single desktop, the computational complexity of MEEFIS appears to be higher than its single-model counterparts. The proposed ensemble model, in fact, is highly parallelizable since all the component learners can be implemented in parallel on different computing nodes. The computational efficiency of MEEFIS can be significantly improved through parallelization.

On the other hand, it is also shown by numerical experiments that MEEFIS requires more training samples because of its deeper ensemble structure. In the future, the learning efficiency of MEEFIS needs to be improved. Ideally, MEEFIS should be able to learn a well-performing ensemble model with a smaller amount of training samples.

IV. CONCLUSION

This paper explores the possibility to build alternative multi-layered learning model using EFSs and introduces a novel ensemble fuzzy system named MEEFIS. The proposed system can learn multi-layered distributed representations from streaming data and continuously self-develop itself to follow the changing data patterns. Experimental investigation shows that MEEFIS is able to achieve high-level performance on a wide variety of benchmark problems, demonstrating the very promising future of this work.

Nevertheless, it has to be admitted that this paper only presents a seminal work and a few preliminary results on prediction and classification problems. There is much to explore in this direction. There are several considerations for future work. Firstly, the performance and learning efficiency of the component learner, namely, EFIS need to be further improved. One may also consider to replace it with more advanced EFSs. Secondly, this paper only explores a few types of ensemble architectures in numerical experiments, it is worth to try alternative ensemble architectures and see how they perform on different problems. Thirdly, it would be very interesting to build an ensemble system with different types of EFSs that utilize different algorithmic procedures for system identification. Such an ensemble learning model may demonstrate far better performance than the approach presented in this paper.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature Methods*, vol. 13, no. 1, pp. 35–35, 2015.

[2] D. Amodei et al., "Deep speech 2: endtoend speech recognition in english and mandarin," in *International conference on machine learning*, 2016, pp. 173–182.

[3] D. Ciresan, U. Meier, and J. Schmidhüber, "Multi-column deep neural networks for image classification," in *International Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3642–3649.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[5] Z. H. Zhou and J. Feng, "Deep forest: towards an alternative to deep neural networks," in *International Joint Conference on Artificial Intelligence*, 2017, pp. 3553–3559.

[6] H. Hagras, "Toward human-understandable, explainable AI," *Computer (Long. Beach. Calif).*, vol. 51, no. 9, pp. 28–36, 2018.

[7] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks,"*IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 1–1, 2019.

[8] J. Feng, Y. Yu, and Z. H. Zhou, "Multi-layered gradient boosting decision trees," in *Advances in neural information processing systems*, 2018, pp. 3551-3561.

[9] E. Lughofer, *Evolving fuzzy systems-methodologies, advanced concepts and applications*. Berlin: Springer, 2011.

[10] L. Maciel, R. Ballini, and F. Gomide, "Evolving possibilistic fuzzy modeling for realized volatility forecasting with jumps," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 2, pp. 302–314, 2017.

[11] R. E. Precup, T. A. Teban, A. Albu, A. I. Szedlak-Stinean, and C. A. Bojan-Dragos, "Experiments in incremental online identification of fuzzy models of finger dynamics," *Romanian Journal of Information Science and Technology*, vol. 21, no. 4, pp. 358–376, 2018.

[12] I. Škrjanc, G. Andonovski, A. Ledezma, O. Sipele, J. A. Iglesias, and A. Sanchis, "Evolving cloud-based system for the recognition of drivers' actions," *Expert Systems with Applications*, vol. 99, pp. 231–238, 2018.

[13] J. De Jesús Rubio, D. R. Cruz, I. Elias, G. Ochoa, R. Balcazarand, and A. Aguilar, "ANFIS system for classification of brain signals," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 3, pp. 4033–4041, 2019.

[14] J. De Jesús Rubio, et al., "Unscented Kalman filter for learning of a solar dryer and a greenhouse," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 5, pp. 6731–6747, 2019.

[15] J. De Jesús Rubio, "SOFMLS: online self-organizing fuzzy modified least-squares network," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[16] P. P. Angelov, X. Gu, and J. C. Principe, "Autonomous learning multi-model systems from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, pp. 2213–2224, 2018.

[17] R. Bao, H. Rong, P. P. Angelov, B. Chen, and P. K. Wong, "Correntropy-based evolving fuzzy neural system," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1324–1338, 2018.

[18] C. N. Giap, L. H. Son, and F. Chiclana, "Dynamic structural neural network," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 32, pp. 2479–2490, 2018.

[19] D. Ge and X. J. Zeng, "A self-evolving fuzzy system which learns dynamic threshold parameter by itself," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 8, pp. 1625–1637, 2019.

[20] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press, 2000.

[21] L. Breiman, "Random forests," *Machine Learning Proceedings*, vol. 45, no. 1, pp. 5–32, 2001.

[22] N. K. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[23] P. P. Angelov and D. P. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*: *Cybernetics*, vol. 34, no. 1, pp. 484–498, 2004.

[24] H. J. Rong, N. Sundararajan, G. Bin Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[25] P. Angelov and X. Zhou, "Evolving fuzzy-rule based classifiers from data streams," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1462–1474, 2008.

[26] E. D. Lughofer, "FLEXFIS: a robust incremental learning approach for evolving Takagi-Sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1393–1410, 2008.

[27] M. Pratama, S. G. Anavatti, P. P. Angelov, and E. Lughofer, "PANFIS: a novel incremental learning machine," *IEEE Transactions on Neural Network and Learning Systems*, vol. 25, no. 1, pp. 55–68, 2014.

[28] M. Pratama, S. G. Anavatti, and E. Lughofer,"Genefis: toward an effective localist network," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 547–562, 2014.

[29] K. Subramanian, A. K. Das, S. Sundaram, and S. Ramasamy, "A meta-cognitive interval type-2 fuzzy inference system and its projection based learning algorithm," *Evolving Systems*, vol. 5, no. 4, pp. 219–230, 2014.

[30] M. Pratama, J. Lu, and G. Zhang, "Evolving type-2 fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 3, pp. 574–589, 2016.

[31] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey," *Information Sciences*, vol. 490, pp. 344–368, 2019.

[32] R. Polikar,"Ensemble Based Systems in Decision Making," *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006.

[33] J. A. Iglesias, A. Ledezma, and A. Sanchis, "Ensemble method based on individual evolving classifiers," in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2013, pp. 56–61.

[34] P. Angelov and X. Gu, "MICE: multi-layer multi-model images classifier ensemble," in *IEEE International Conference on Cybernetics*, 2017, pp. 436–443.

[35] P. Angelov and X. Gu, "A cascade of deep learning fuzzy rule-based image classifier and SVM," in *IEEE International Conference on Systems, Man and Cybernetics*, 2017, pp. 1–8.

[36] X. Gu, P. P. Angelov, C. Zhang, and P. M. Atkinson, "A massively parallel deep rule-based ensemble classifier for remote sensing scenes," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 345–349, 2018.

[37] D. Kangin, P. Angelov, J. A. Iglesias, and A. Sanchis, "Evolving classifier TEDAClass for big data," *Procedia Computer Science*, vol. 53, no. 1, pp. 9–18, 2015.

[38] M. Pratama, W. Pedrycz, and E. Lughofer, "Evolving ensemble fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2552–2567, 2018.

[39] P. Angelov and R. Yager, "A new type of simplified fuzzy rule-based system," *International Journal of General Systems*, vol. 41, no. 2, pp. 163–185, 2012.

[40] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[41] X. Gu and P. Angelov, "Self-boosting first-order autonomous learning neuro-fuzzy systems," *Applied Soft Computing*, vol. 77, pp. 118–134, 2019.

[42] H. J. Rong, G. Bin Huang, N. Sundararajan and P. Saratchandran, "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Transactions on Systems, Man, and Cybernetics - Part B*: *Cybernetics*, vol. 39, no. 4, pp. 1067–1072, 2009.

[43] H. J. Rong, N. Sundararajan, G. B. Huang and G.S. Zhao "Extended sequential adaptive fuzzy inference system for classification problems," *Evolving Systems*, vol.2, no. 2, pp. 71-82, 2011.

[44] D. Ge and X. J. Zeng,"Learning evolving T-S fuzzy systems with both local and global accuracy - a local online optimization approach," *Applied Soft Computing*, vol. 86, pp. 795–810, 2018.

[45] D. Ge and X. J. Zeng,"Learning data streams online- an evolving fuzzy system approach with self-learning/adaptive thresholds," *Information Sciences*, vol. 507, pp. 172-184, 2020.

[46] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[47] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers," *Multiple Classifier Systems*, vol. 34, pp. 1–17, 2007.

[48] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of machine learning research*, vol. 3, no. 1, pp. 115-143, 2002.