Multilayer Evolving Fuzzy Neural Networks

Xiaowei Gu, Plamen Angelov, *Fellow, IEEE*, Jungong Han and Qiang Shen

***Abstract*—It is widely recognised that learning systems have to go deeper to exchange for more powerful representation learning capabilities in order to precisely approximate nonlinear complex problems. However, the best known computational intelligence approaches with such characteristics, namely, deep neural networks, are often criticised for lacking transparency. In this paper, a novel multilayer evolving fuzzy neural network (MEFNN) with a transparent system structure is proposed. The proposed MEFNN is a meta-level stacking ensemble learning system composed of multiple cascading evolving neuro-fuzzy inference systems (ENFISs), processing input data layer-by-layer to automatically learn multi-level nonlinear distributed representations from data. Each ENFIS is an evolving fuzzy system capable of learning from new data sample by sample to self-organise a set of human-interpretable IF-THEN fuzzy rules that facilitate approximate reasoning. Adopting ENFIS as its ensemble component, the multilayer system structure of MEFNN is flexible and transparent, and its internal reasoning and decision-making mechanism can be explained and interpreted to/by humans. To facilitate information exchange between different layers and attain stronger representation learning capability, MEFNN utilises error backpropagation to self-update the consequent parameters of the IF-THEN rules of each ensemble component based on the approximation error propagated backward. To enhance the capability of MEFNN to handle complex problems, a nonlinear activation function is introduced to modelling the consequent parts of the IF-THEN rules of ENFISs, thereby empowering both the representation and the reflection of nonlinearity in the resulting fuzzy outputs. Numerical examples on a wide variety of challenging (benchmark and real-world) classification and regression problems demonstrate the superior practical performance of MEFNN, revealing the effectiveness and validity of the proposed approach.**

***Index Terms*—evolving fuzzy system; fuzzy neural network; self-organised; stacking ensemble.**

I. Introduction

DEEP neural networks (DNNs, or artificial neural networks, ANNs) have demonstrated eye-catching successes on a range of practical problems concerning image, video, speech and text processing [1], [2]. Currently, DNNs are one of the most popular computational intelligence approaches, thanks to the state-of-the-art (SOTA) performances they have offered in terms of prediction accuracy, and have been implemented for many real-world applications such as autonomous driving [3], drug discovery [4], stock prediction [5], etc.

X. Gu is with the School of Computing, University of Kent, Canterbury, CT2 7NZ, UK. email: X.Gu@kent.ac.uk.

P. Angelov is with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK. email: p.angelov@lancaster.ac.uk.

J. Han is with the Department of Computer Science, the University of Sheffield, Sheffield, S10 2TN, UK. email: jungong.han@sheffield.ac.uk.

Q. Shen is with the Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK. email: qqs@aber.ac.uk.

Corresponding author: Xiaowei Gu

Manuscript received XXXX XX, 2023; revised XXXX XX, 2023.

The success of DNNs is built upon their capability to automatically learn multiple levels of nonlinear distributed representations from raw data through a general-purpose learning procedure [6]. It is widely recognised that this powerful representation learning ability of DNNs comes from their huge amounts (millions or sometimes billions) of parameters that are arranged in multiple layers and tuned to preserve abstract information learned from training data [7], [8]. However, DNNs are the typical type of "black box" models lack of transparency. They are extremely complicated models, and their parameters do not carry clear physical meanings that can be linked to the practical problems directly. Consequently, their internal reasoning processes are not interpretable by humans and their decisions are not explainable [9]. Another critical issue of DNNs is their vulnerability to real-world uncertainties. DNNs are fragile to new samples of unfamiliar data patterns, and they cannot be fixed easily because the training process is computationally expensive and usually requires huge amount of training data. These deficiencies have limited the wider deployment of DNNs in real-world applications, particularly, these high-stake ones [9]. It has also been shown by recent studies that the huge increase in interest towards DNNs starts to saturate [8].

Realising that the multi-level nonlinear distributed representations are the key to the success of DNNs, there have been a few works published in recent years attempting to build alternative multilayer stacking ensemble models that offer competitive performance to DNNs but with less deficiencies aforementioned [7], [10], [11]. Stacking is an ensemble learning scheme for minimising the prediction error of individual base models by arranging them in multiple layers, such that the outputs of the base models at a lower layer are used as the inputs of the base models at the layer above [12]. In this way, base models at upper layers learn to improve the predictions made by base models at lower layers and achieve improved prediction accuracy [13]. Unlike the alternative better-known parallel ensemble schemes such as bagging [14], [15] and boosting [16], [17], which attempt to increase the diversity between base models to complement each other, stacking constructs a meta-level learning model to learn multi-level distributed representations from data, exploiting a more sophisticated decision-making strategy than (weighted) majority voting. Thanks to the outstanding performance on many learning tasks, stacking has received increasing popularity in recent years [18], [19].

On the other hand, existing ensemble learning approaches (including the stacking ones) mainly focus on employing mainstream learning models, such as decision tree (DT), random forest (RF), k-nearest neighbour (KNN), support vector machine (SVM), multilayer perception (MLP), long-short term memory (LSTM), convolutional neural networks (CNN), etc.,

as base models to construct ensemble predictors [20]–[23]. Although such ensemble models have demonstrated great performance, these models are typically limited to offline scenarios and can only work with static data. In addition, many of these mainstream base models (e.g., SVM, MLP, LSTM, CNN) are often being criticised for lacking transparency. As a result, there is a strong demand for utilising alternative learning models in building ensemble models such that the created ensembles are capable of self-updating from new data whilst offering greater transparency and interpretability.

Evolving fuzzy systems (EFSs) are a special group of fuzzy systems and can be implemented in the form of fuzzy rule-based systems [24] or neuro-fuzzy systems [25]. EFSs are the prominent methodology for real-time non-stationary problem approximation [26]. They are capable of self-developing the transparent system structure and self-updating the parameters from data streams through a human-interpretable reasoning process to capture the changing data patterns. EFSs are well known for their strong capability to handle inherent uncertainties in data, and they have been widely implemented for many real-world applications concerning data stream processing [27], [28]. As a hot research topic, a variety of EFSs have been proposed in the literature since the underlying concepts being introduced around the beginning of this century [25], [29]. The most representative evolving fuzzy rule-based systems include, but are not limited to, evolving Takagi-Sugeno system (eTS) [29], sequential adaptive fuzzy inference system (SAFIS) [30], evolving fuzzy rule-based classifiers (eClass0 and eClass1) [24], extended sequential adaptive fuzzy inference system (ESAFIS) [31], generalized smart evolving fuzzy systems (Gen-Smart-EFS) [32], evolving fuzzy model (eFuMo) [33], self-evolving fuzzy system (SEFS) [34], autonomous learning multi-model system (ALMMo) [35], recursive maximum correntropy-based evolving fuzzy system (RMCEFS) [36], evolving fuzzy system with self-learning/adaptive thresholds (EFS-SLAT) [37], statistically evolving fuzzy inference system (SEFIS) [38], etc. The most popular examples of evolving neuro-fuzzy models include dynamic evolving neural-fuzzy inference system (DENFIS) [25], self-organising fuzzy neural network (SOFNN) [39], evolving granular neural network [40], generic evolving neuro-fuzzy inference system (GENEFIS) [41], parsimonious network based on fuzzy inference system (PANFIS) [42], correntropy-based evolving fuzzy neural system (CEFNS) [43], parsimonious learning machine (PALM) [44], etc.

EFSs have demonstrated significant successes in a wide variety of time-critical applications in dynamical environment thanks to their simpler, highly flexible system structure and transparent internal reasoning mechanism [45]. However, it is also recognised that EFSs usually cannot reach the same level of performance as DNNs on large-scale, high-dimensional, complex problems (e.g., image classification, image segmentation). Although there have been a number of EFS-based ensemble models proposed in the recent years reaching greater prediction performance on many challenging problems beyond individual single-model EFSs, the vast majority of existing works are focused on exploiting either parallel ensemble architectures to enhance the diversity between base models [14], [46]–[49] or sequential ensemble architectures to enhance the adaptability of the ensemble system towards data pattern drifting in non-stationary environments by constructing a new base model from each newly arrived data chunk [50]. There only exist very few works on exploring the potential of building stacking ensemble models with EFSs to facilitate multi-level distributed representation learning [11], [51]. On the other hand, EFSs dominantly use the standard recursive least squares (RLS) algorithm or its variants for consequent parameter learning [25], [29], [40], [52], [53], which requires the error function to be explicitly defined. As a result, individual EFS models in a standard stacking ensemble framework have to learn from input data separately to minimise the differences between their predictions and the ground truth in order to avoid any ambiguity in defining the error functions for different layers. The lack of interaction between base models greatly restricts the capability of stacking ensemble models to learn more informative and descriptive representations from data.

To learn more descriptive multi-level distributed representations with high-level transparency and interpretability, a novel stacking ensemble fuzzy model named multilayer evolving fuzzy neural network (MEFNN) is introduced in this paper. The proposed MEFNN is a meta-level learning model consisted of a number of evolving neuro-fuzzy inference systems (ENFISs) cascading in multiple layers. Each layer of MEFNN is based on a single ENFIS that takes the outputs from the previous layer as its inputs, and passes its outputs to the next layer as inputs. In this way, MEFNN processes input data layer-by-layer to learn multi-level distributed representations. Each ENFIS is a multiple-input multiple-output (MIMO) EFS that self-organises and self-develops its system structure and parameters from data streams on a sample-by-sample basis. To enhance the capability of handling nonlinear problems, a standard sigmoid function is introduced to the consequent parts of the IF-THEN fuzzy rules used by ENFISs to empower both the representation and the reflection of nonlinearity in the resulting fuzzy outputs, thereby enhancing the capability of MEFNN to handle complex problems. Unlike traditional EFSs, MEFNN employs the error backpropagation algorithm for updating the consequent parameters of its individual base models, which effectively avoids ambiguity in defining the error function for each individual layer by allowing the approximation error to be propagated backward from the last layer to the first layers. In this way, all the base models can interact and communicate with each other to learn more descriptive multi-level representations from data. Therefore, the proposed MEFNN can also be viewed as a special type of ANN with a highly flexible self-evolving multilayer structure (free from the requirement of specifying the numbers of nodes for each layer), human-interpretable internal reasoning mechanism and meaningful parameters that can be directly linked to the practical problems.

To summarise, key features of the proposed MEFNN include:

1) A meta-level ensemble architecture composed of multiple cascading base models with self-evolving system structure and parameters to learn multi-level distributed representations from data on a sample-by-sample basis;

2) The use of error backpropagation to sequentially update the consequent parameters of the self-evolving base models based on the error of the existing approximation, facilitating information exchange between different layers to attain stronger representation learning capability;
3) The use of the sigmoid function in the consequent parts of IF-THEN fuzzy rules to empower both the representation and the reflection of nonlinearity in the fuzzy outputs of the base models, thereby enhancing the capability of the overall MEFNN to handle nonlinear, complex problems.

The remainder of this paper is organised as follows. Technical details of MEFNN are presented in Section II. Numerical examples on benchmark classification and regression problems are given in Section III as the proof of concept. This paper is concluded in Section IV.

II. Proposed MEFNN

A. General Architecture

The general architecture of MEFNN is depicted in Fig. 1. One can see from Fig. 1 that the proposed MEFNN is a stacking ensemble composed of multiple ENFISs arranged in layers (one ENFIS per layer). The inputs of MEFNN are processed layer-by-layer until the final outputs are produced. Each ENFIS takes the outputs of the previous layer as its inputs (except for the first ENFIS) and uses its outputs as the inputs of the next layer (except for the last ENFIS). Hence, different layers are fully connected.

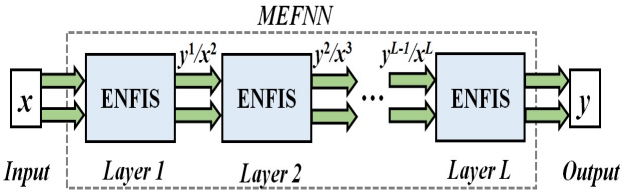

Fig. 1: General Architecture of MEFNN.

Assuming that MEFNN is composed of L cascading ENFISs, the input-output relationship of the ENFIS at the l^{th} layer can be defined by Eq. (1) ($l = 1, 2, \ldots, L$):

$$\boldsymbol{y}^l = f^l(\boldsymbol{x}^l) \tag{1}$$

where $\boldsymbol{x}^l = [x_1^l, x_2^l, \ldots, x_{M^l}^l]^T$ is the $M^l \times 1$ dimensional input vector; $\boldsymbol{y}^l = [y_1^l, y_2^l, \ldots, y_{W^l}^l]^T$ is the $W^l \times 1$ dimensional output.

Since the l^{th} layer takes the output of the $l-1^{th}$ layer as its input, namely, $\boldsymbol{x}^l = \boldsymbol{y}^{l-1}$ and $M^l = W^{l-1}$, the output of the l^{th} layer can also be defined by the following composite function with respect to the input of the $l-1^{th}$ layer, $\boldsymbol{x}^{l-1}$:

$$\boldsymbol{y}^l = (f^l \circ f^{l-1})(\boldsymbol{x}^{l-1}) = f^l(f^{l-1}(\boldsymbol{x}^{l-1})) \tag{2}$$

Similarly, the overall ensemble system with a L-layer structure can be formulated by the following composite function:

$$\boldsymbol{y} = (f^L \circ f^{L-1} \circ \cdots \circ f^2 \circ f^1)(\boldsymbol{x}) \tag{3}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$ and $\boldsymbol{y} = [y_1, y_2, \ldots, y_W]^T$ are the respective input and output vectors of MEFNN as depicted in Fig. 1. Note that the inputs to MEFNN are normalised to the range of $[0, 1]$ in advance [54].

Different from conventional ANNs, MEFNN does not require users to specify the numbers of nodes in each layer. Instead, users only need to set the output sizes of the cascading ENFISs (except for the final one) for MEFNN, namely, W^1, W^2,..., W^{L-1}. The inner structure of each individual base model will automatically self-evolve from the input data to incorporate the observed data patterns without relying on any prior assumptions concerning data distribution and other properties. The output sizes control the amounts of information flowing between layers, and can be predefined based on users' preferences in the absence of specialised expertise and prior knowledge. In practice, the output size of a base model should be no greater than its input size to avoid overfitting and also, should be no less than the output size of the final layer to avoid losing too much information during the transmission between the layers. A detailed discussion on the parameter settings of MEFNN, e.g., number of layers, output sizes of cascading base models, and their influence on the prediction performance of the system is presented in Section III.A. A recommended parameter setting is also given in Section III.A.

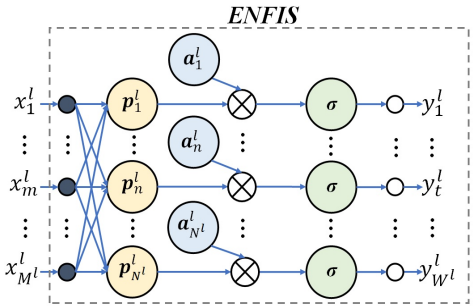

Fig. 2: Inner architecture of the l^{th} ENFIS

The inner architecture of the l^{th} ENFIS is presented in Fig. 2. ENFIS is a MIMO neuro-fuzzy system composed of N^l first-order IF-THEN fuzzy production rules in the following form ($l = 1, 2, \ldots, L$):

$$\boldsymbol{R}_n^l: \quad IF\ (\boldsymbol{x}^l \sim \boldsymbol{p}_n^l)\ \ THEN\ (\boldsymbol{y}_n^l = \sigma(\mathbf{A}_n^l \bar{\boldsymbol{x}}^l)) \tag{4}$$

where $\boldsymbol{R}_n^l$ denotes the n^{th} IF-THEN rule of l^{th} ENFIS in MEFNN; $n = 1, 2, \ldots, N^l$; N^l is the number of rules; $\boldsymbol{x}^l$ is the $M^l \times 1$ dimensional input vector; $\bar{\boldsymbol{x}}^l = [1, (\boldsymbol{x}^l)^T]^T$; "$\sim$" denotes similarity; $\boldsymbol{p}_n^l = [p_{n,1}^l, p_{n,2}^l, \ldots, p_{n,M^l}^l]^T$ is the $M^l \times 1$ dimensional prototype (antecedent parameters) of $\boldsymbol{R}_n^l$; $\mathbf{A}_n^l = [\boldsymbol{a}_{n,1}^l, \boldsymbol{a}_{n,2}^l, \ldots, \boldsymbol{a}_{n,W^l}^l]^T$ is the $W^l \times (M^l + 1)$ dimensional consequent parameter matrix and there is $\boldsymbol{a}_{n,w}^l = [a_{n,w,0}^l, a_{n,w,1}^l, \ldots, a_{n,w,M^l}^l]^T$ ($w = 1, 2, \ldots, W^l$); $\boldsymbol{y}_n^l = [y_{n,1}^l, y_{n,2}^l, \ldots, y_{n,W^l}^l]^T$ is the $W^l \times 1$ dimensional output of $\boldsymbol{R}_n^l$ and $\sigma(\cdot)$ denotes the standard activation function used by ANNs [55]. The activation function is utilised in the consequent part of the IF-THEN rules to add extra nonlinearity to ENFIS. Compared with the first-order IF-THEN rules used by conventional EFSs [24], [56], such modification (the utilisation of activation function) greatly enhances the capability of ENFIS to handle nonlinear, complex problems. In this

study, the classic sigmoid function, namely, $\sigma(x) = \frac{1}{1+e^{-x}}$ is employed. However, one may consider other types of activation functions, such as hyperbolic tangent, rectified linear unit.

Due to the utilisation of sigmoid function, each IF-THEN rule of ENFIS becomes a nonlinear model, and the output $\boldsymbol{y}^l$ of ENFIS in response to a particular input vector $\boldsymbol{x}^l$ is produced as a weighted sum of the outputs generated by its individual IF-THEN rules as Eq. (5):

$$\boldsymbol{y}^l = f^l(\boldsymbol{x}^l) = \sum_{n=1}^{N^l} \lambda_n^l \boldsymbol{y}_n^l = \sum_{n=1}^{N^l} \lambda_n^l \sigma(\mathbf{A}_n^l \bar{\boldsymbol{x}}^l) \tag{5}$$

where λ_n^l is the firing strength of $\boldsymbol{x}^l$ to $\boldsymbol{R}_n^l$, defined as follows [26].

$$\lambda_n^l = \frac{D_n(\boldsymbol{x}^l)}{\sum_{j=1}^{N^l} D_j(\boldsymbol{x}^l)} \tag{6}$$

and there is $D_n(\boldsymbol{x}^l) = e^{-\frac{||\boldsymbol{x}^l - \boldsymbol{p}_n^l||^2}{(\tau_n^l)^2}}$; $||\boldsymbol{x}^l - \boldsymbol{p}_n^l|| = \sqrt{(\boldsymbol{x}^l - \boldsymbol{p}_n^l)^T(\boldsymbol{x}^l - \boldsymbol{p}_n^l)}$; τ_n^l is the width of the exponential kernel associated with $\boldsymbol{R}_n^l$, and its value is derived directly based on the mutual distances of data (as to be specified later in Eq. (11)).

In the next subsection, the identification process of MEFNN, which include structure evolving and parameter learning, is presented. It is worth noting that all the cascading ENFISs follow the exactly same identification and decision-making procedures.

B. Identification Process

As aforementioned, MEFNN self-organises its multilayer structure and parameters from the scratch by learning from data on a sample-by-sample basis. The identification process of MEFNN is composed of the following three stages repeated for every input.

Stage 0. System Initialization. Given a particular input $\boldsymbol{x}_k$, the system structures and parameters of the cascading ENFISs in MEFNN will be initialised one-by-one from the first ($l = 1$) layer to the last ($l = L$) layer if $\boldsymbol{x}_k$ is the very first input, namely, $k = 1$. Otherwise, the identification process enters Stage 1 instead.

Once the l^{th} ($l = 1, 2, \ldots, L$) ENFIS receives its first input, $\boldsymbol{x}_k^l$ ($\boldsymbol{x}_k^l = \boldsymbol{x}_k$ if $l = 1$ or $\boldsymbol{x}_k^l = \boldsymbol{y}_k^{l-1}$ if $l > 1$), its global parameters are set as [26]:

$$\boldsymbol{\mu}^l \leftarrow \boldsymbol{x}_k^l; \quad X^l \leftarrow ||\boldsymbol{x}_k^l||^2 \tag{7}$$

where $\boldsymbol{\mu}^l$ is the global mean of all the input samples to the l^{th} ENFIS; X^l is the global mean of the squared Euclidean norms of all these input samples.

The first IF-THEN rule, $\boldsymbol{R}_{N^l}^l$ ($N^l \leftarrow 1$) is initialised in the form of Eq. (4) with its antecedent (prototype) and consequent parameters set by Eq. (8):

$$\boldsymbol{p}_{N^l}^l \leftarrow \boldsymbol{x}_k^l; \quad \mathbf{A}_{N^l}^l \leftarrow \frac{1}{M^l + 1}\boldsymbol{\epsilon}_o \tag{8}$$

where $\boldsymbol{\epsilon}_o = [\epsilon_{o,j,k}]_{k=1:(M^l+1)}^{j=1:W^l}$ is a randomly generated $W^l \times (M^l + 1)$ dimensional matrix, whose elements equal to either 0 or 1, namely, $\epsilon_{o,j,k} \in \{0, 1\}\ \forall\ j, k$, following the symmetrical binomial distribution. This simple initialisation strategy prevents the derivatives from always being zeros.

Parameters of the shape-free cluster formed around the prototype $\boldsymbol{p}_{N^l}^l$ denoted by $\boldsymbol{C}_{N^l}^l$ are initialised as follows [26].

$$\boldsymbol{c}_{N^l}^l \leftarrow \boldsymbol{x}_k^l; \quad \chi_{N^l}^l \leftarrow ||\boldsymbol{x}_k^l||^2; \quad S_{N^l}^l \leftarrow 1 \tag{9}$$

where $\boldsymbol{c}_{N^l}^l$ is the mean of data samples associated with $\boldsymbol{C}_{N^l}^l$; $\chi_{N^l}^l$ is the mean of the squared Euclidean norms of these samples; $S_{N^l}^l$ is the support (number of members) of $\boldsymbol{C}_{N^l}^l$.

After this, the l^{th} ENFIS produces the output $\boldsymbol{y}_k^l$ using Eq. (5). The output $\boldsymbol{y}_k^l$ is then used as the input of the ENFIS at the next layer ($\boldsymbol{x}_k^{l+1} \leftarrow \boldsymbol{y}_k^l$). The same process repeats for every individual ENFIS in the stacking ensemble until the last (L^{th}) ENFIS is initialised and the final output of MEFNN is produced ($\boldsymbol{y}_k \leftarrow \boldsymbol{y}_k^L$). Then, the identification process enters Stage 2 for consequent parameter learning.

Stage 1. Structure Evolving and Output Generation. If $k > 1$, namely, $\boldsymbol{x}_k$ is not the first input sample, the system structure and parameters of MEFNN will be updated by adjusting the cascading ENFISs layer-by-layer with respect to the input.

Given the corresponding input, $\boldsymbol{x}_k^l$ ($\boldsymbol{x}_k^l = \boldsymbol{x}_k$ if $l = 1$ or $\boldsymbol{x}_k^l = \boldsymbol{y}_k^{l-1}$ if $l > 1$), the global parameters of the l^{th} ENFIS are updated by Eq. (10) firstly [26]:

$$\begin{aligned} \boldsymbol{\mu}^l &\leftarrow \boldsymbol{\mu}^l + \frac{\boldsymbol{x}_k^l - \boldsymbol{\mu}^l}{k} \\ X^l &\leftarrow X^l + \frac{||\boldsymbol{x}_k^l||^2 - X^l}{k} \end{aligned} \tag{10}$$

Then, the local density of $\boldsymbol{x}_k^l$ at each cluster, denoted as $D_n(\boldsymbol{x}_k^l)$, is calculated by Eq. (11) ($n = 1, 2, \ldots, N^l$) [26]:

$$D_n(\boldsymbol{x}_k^l) = e^{-\frac{||\boldsymbol{x}_k^l - \boldsymbol{p}_n^l||^2}{(\tau_n^l)^2}} \tag{11}$$

where $\tau_n^l = \sqrt{\frac{X^l - ||\boldsymbol{\mu}^l||^2 + \chi_n^l - ||\boldsymbol{c}_n^l||^2}{2}}$.

Condition 1 is checked to see if $\boldsymbol{x}_k^l$ represents a novel data pattern unseen from the historical inputs [26]:

$$\begin{aligned} \text{Cond. 1}: \ & if\ \big(\max_{n=1,2,\ldots,N^l}(D_n(\boldsymbol{x}_k^l)) < \delta_o\big) \\ & then\ (\boldsymbol{x}_k^l\ becomes\ a\ new\ prototype) \end{aligned} \tag{12}$$

where δ_o is a threshold to determine whether $\boldsymbol{x}_k^l$ is sufficiently different from the data patterns represented by existing prototypes, and there is $0 < \delta_o < 1$. In this study, $\delta_o = e^{-3}$ is used. According to the Chebyshev rule, the chance for the Euclidean distance between $\boldsymbol{x}_k^l$ and $\boldsymbol{p}_n^l$ to be greater than $\sqrt{3}\tau_n^l$ is less than 33.3%.

If Condition 1 is satisfied, it suggests that $\boldsymbol{x}_k^l$ falls out of the areas of influence of existing prototypes, showing a significant departure from the existing local models in the l^{th} ENFIS. In other words, $\boldsymbol{x}_k^l$ is distinctive from existing prototypes and more likely to represent an unseen data pattern. Hence, a new IF-THEN rule ($N^l \leftarrow N^l + 1$) is introduced to the l^{th} ENFIS to incorporate this new data pattern represented by $\boldsymbol{x}_k^l$. The antecedent and consequent parameters of the new rule $\boldsymbol{R}_n^{N^l}$ are set by Eq. (8) and the parameters of the associated shape-free cluster are initialised by Eq. (9).

However, if $\boldsymbol{x}_k^l$ fails to meet Condition 1, $\boldsymbol{x}_k^l$ is used for updating the parameters of the cluster associated with the nearest prototype, denoted by $\boldsymbol{p}_{n^*}^l$ via Eq. (13) [26].

$$\begin{aligned}\boldsymbol{c}_{n^*}^l &\leftarrow \boldsymbol{c}_{n^*}^l + \frac{\boldsymbol{x}_k^l - \boldsymbol{c}_{n^*}^l}{S_{n^*}^l + 1}\\ \chi_{n^*}^l &\leftarrow \chi_{n^*}^l + \frac{||\boldsymbol{x}_k^l||^2 - \chi_{n^*}^l}{S_{n^*}^l + 1}\\ S_{n^*}^l &\leftarrow S_{n^*}^l + 1\end{aligned} \tag{13}$$

where $n^* = \arg\max_{n=1,2,\ldots,n^*}(D_n(\boldsymbol{x}_k^l))$; $\boldsymbol{c}_{n^*}^l$, $\chi_{n^*}^l$ and $S_{n^*}^l$ are the parameters of $\boldsymbol{C}_{n^*}^l$ associated with $\boldsymbol{p}_{n^*}^l$.

After the parameter updating and possible structure evolving, the l^{th} ENFIS produces the output $\boldsymbol{y}_k^l$ using Eq. (5). The output $\boldsymbol{y}_k^l$ is then used as the input of ENFIS at the next layer ($\boldsymbol{x}_k^{l+1} \leftarrow \boldsymbol{y}_k^l$). The same updating process repeats for each individual ENFIS until the last one (namely, the L^{th} one) has been updated. The identification process enters Stage 2 for consequent parameter learning once the final output, $\boldsymbol{y}_k$ of MEFNN in response to $\boldsymbol{x}_k$ has been produced ($\boldsymbol{y}_k \leftarrow \boldsymbol{y}_k^L$). Note that different from conventional EFSs, the outputs of individual base models in MEFNN are produced after structural evolving and antecedent parameter updating. In so doing, the derivative of the measured prediction errors with respect to the consequent parameters gives a better idea regarding how to adjust the consequent parameters, in order to minimise prediction errors given the current model structure and antecedent parameters.

Stage 2. Consequent Parameter Updating. As aforementioned, MEFNN utilises error backpropagation for updating the consequent parameters of its individual ensemble components unlike conventional first-order EFSs. Backpropagation is commonly used by ANNs and neuro-fuzzy systems with prefixed system structure [57]–[59]. The main reason for choosing backpropagation rather than standard algorithms used by EFSs such as RLS and its variants or least squares [25], [60] is because only the last ENFIS in the stacking ensemble has the target output. This makes it practically impossible to define an error function for any layer except the final one. Backpropagation, however, allows the error to be propagated backwards from the final layer to the first layer by the use of chain rule, greatly facilitating the information exchanging between different layers. Due to the use of backpropagation, MEFNN also removes the need of co-variance matrices by conventional RLS-based algorithms for consequent parameter updating, which greatly reduces the computational complexity especially when the data dimensionality is high.

Once MEFNN generates the final output, $\boldsymbol{y}_k$ in response to the current input, $\boldsymbol{x}_k$, the corresponding loss function is defined as:

$$\boldsymbol{e}_k = \frac{1}{2}(\boldsymbol{y}_k - \boldsymbol{r}_k)^T(\boldsymbol{y}_k - \boldsymbol{r}_k) \tag{14}$$

where $\boldsymbol{r}_k = [r_{k,1}, r_{k,2}, \ldots, r_{k,W}]^T$ is the corresponding target output. Note that for classification problems, $\boldsymbol{r}_k$ is the vectorised class label of $\boldsymbol{x}_k$ via dummy coding.

The derivative of the prediction error with respect to $\boldsymbol{y}_k$ is obtained as Eq. (15):

$$\frac{\partial \boldsymbol{e}_k}{\partial \boldsymbol{y}_k} = \boldsymbol{y}_k - \boldsymbol{r}_k \tag{15}$$

Then, the consequent parameter matrices of the cascading ENFISs are updated one-by-one backwards from the L^{th} layer to the 1^{st} layer utilising the chain rule.

The derivative of the consequent parameter matrix of the l^{th} ENFIS is derived as follows ($n = 1, 2, \ldots, N^l$):

$$\frac{\partial \boldsymbol{e}_k}{\partial \mathbf{A}_n^l} = \frac{\partial \boldsymbol{e}_k}{\partial \boldsymbol{y}_k^L} \cdot \frac{\partial \boldsymbol{y}_k^L}{\partial \boldsymbol{y}_k^{L-1}} \cdot \cdots\cdot \frac{\partial \boldsymbol{y}_k^{l+1}}{\partial \boldsymbol{y}_k^l} \cdot \frac{\partial \boldsymbol{y}_k^l}{\partial \mathbf{A}_n^l} \tag{16}$$

where $l = 1, 2, \ldots, L$; $\boldsymbol{y}_k^L = \boldsymbol{y}_k$. Given $\boldsymbol{y}_k^j = \boldsymbol{x}_k^{j+1}$, $\forall j = 1, 2, \ldots, L-1$. Eq. (16) can be reformulated as Eq. (17) ($l = 1, 2, \ldots, L$). The detailed derivations are presented in Supplementary Section A.

$$\frac{\partial \boldsymbol{e}_k}{\partial \mathbf{A}_n^l} = \lambda_{n,k}^l \cdot (\boldsymbol{d}_k^l \otimes \sigma'(\mathbf{A}_n^l \bar{\boldsymbol{x}}_k^l)) \cdot (\bar{\boldsymbol{x}}_k^l)^T \tag{17}$$

where "$\otimes$" denotes element-wise multiplication; $\sigma'(\mathbf{A}_n^l \bar{\boldsymbol{x}}_k^l) = \sigma(\mathbf{A}_n^l \bar{\boldsymbol{x}}_k^l) \otimes (1 - \sigma(\mathbf{A}_n^l \bar{\boldsymbol{x}}_k^l))$; $\boldsymbol{d}_k^l$ is the derivate of the loss with respect to $\boldsymbol{y}_k^l$ ($l = 1, 2, \ldots, L-1$):

$$\begin{aligned}\boldsymbol{d}_k^l &= \frac{\partial \boldsymbol{e}_k}{\partial \boldsymbol{y}_k^L} \cdot \frac{\partial \boldsymbol{y}_k^L}{\partial \boldsymbol{y}_k^{L-1}} \cdot \cdots\cdot \frac{\partial \boldsymbol{y}_k^{l+1}}{\partial \boldsymbol{y}_k^l}\\ &= \sum_{n=1}^{N^{l+1}} \Big(\frac{\partial \lambda_{n,k}^{l+1}}{\partial \boldsymbol{x}_k^{l+1}} \cdot \sigma^T(\mathbf{A}_n^{l+1} \bar{\boldsymbol{x}}_k^{l+1}) \cdot \boldsymbol{d}_k^{l+1}\\ &\quad + \lambda_{n,k}^{l+1} \cdot (\tilde{\mathbf{A}}_n^{l+1})^T \cdot (\boldsymbol{d}_k^{l+1} \otimes \sigma'(\mathbf{A}_n^{l+1} \bar{\boldsymbol{x}}_k^{l+1})) \Big)\end{aligned} \tag{18}$$

Here, $\boldsymbol{d}_k^L = \frac{\partial \boldsymbol{e}_k}{\partial \boldsymbol{y}_k^L} = \boldsymbol{y}_k^L - \boldsymbol{r}_k$; $\tilde{\mathbf{A}}_n^l = [a_{n,w,m}^l]_{m=1:M^l}^{w=1:W^l}$ is the $W^l \times M^l$ dimensional consequent parameter matrix obtained by removing the first column of $\mathbf{A}_n^l$; and there is:

$$\frac{\partial \lambda_{n,k}^l}{\partial \boldsymbol{x}_k^l} = \lambda_{n,k}^l \cdot \Big(\frac{2(\boldsymbol{p}_n^l - \boldsymbol{x}_k^l)}{(\tau_n^l)^2} - \sum_{i=1}^{N^l} (\lambda_{i,k}^l \cdot \frac{2(\boldsymbol{p}_i^l - \boldsymbol{x}_k^l)}{(\tau_i^l)^2}) \Big) \tag{19}$$

Based on the derivative $\frac{\partial \boldsymbol{e}_k}{\partial \mathbf{A}_n^l}$, the consequent parameter matrix, $\mathbf{A}_n^l$ can be updated by Eq. (20):

$$\mathbf{A}_n^l \leftarrow \mathbf{A}_n^l - \gamma_o \cdot \frac{\partial \boldsymbol{e}_k}{\partial \mathbf{A}_n^l} \tag{20}$$

where γ_o is the learning rate. In this study, $\gamma_o = 1$ is used by default. However, the learning rate of MEFNN can be adjusted in the same way as the learning rate used in ANNs. Experienced users may further specify a particular learning rate for each layer of MEFNN.

Once the consequent parameter matrices of all the ENFISs in the stacking ensemble have been updated based on the loss calculated with the current output (namely, Eq. (14)), MEFNN can proceed to process the next input ($k \leftarrow k + 1$) and a new learning cycle starts. The system identification process of the proposed MEFNN is summarised by Supplementary Algorithm S1 for clarity.

It has to be stressed that the main purpose of this study is to demonstrate the proposed concept and general principles.

Therefore, the operation mechanism of ENFIS is kept simple with only the essential rule adding scheme being used. However, the proposed MEFNN and its base model, ENFIS are, in fact, highly flexible. Alternative schemes such as rule merging, pruning and splitting can be added to ENFIS to help the model construct and maintain a more compact rule base [37], thereby enabling MEFNN to achieve greater prediction performance. More advanced MIMO EFSs may be employed by MEFNN as base models as well.

C. Computational Complexity Analysis

A detailed analysis on the computational complexity of MEFNN is presented as follows. Since MEFNN processes the data on a sample-by-sample basis, it is assumed that the analysis is performed at the k^{th} time instance at which $\boldsymbol{x}_k$ is given as the input of MEFNN.

Stage 0 is for system initialisation and will not be repeated again after the system has been initialised. Hence, the computational complexity of Stage 0 is negligible within the overall learning process.

Stage 1 is for updating the system structure and producing the output. For the l^{th} ENFIS ($l = 1, 2, \ldots, L$), the computational complexity of updating global parameters $\boldsymbol{\mu}^l$ and X^l in response to $\boldsymbol{x}_k^l$ is $O(M^l)$, and that of calculating the local density value of $\boldsymbol{x}_k^l$ at each cluster is $O(N^l M^l)$. The complexity of adding or updating a cluster is $O(N^l M^l)$, and that of output generation is $O(N^l M^l W^l)$. Therefore, the computational complexity of the l^{th} ENFIS at Stage 1 is $O(N^l M^l W^l)$ and the overall computational complexity of Stage 1 of MEFNN given the input $\boldsymbol{x}_k^l$ is $O(\sum_{l=1}^{L} N^l M^l W^l)$.

The consequent parameters of the individual ensemble components are updated in response to $\boldsymbol{x}_k$ at Stage 2. The complexity of calculating $\frac{\partial \boldsymbol{e}_k}{\partial \mathbf{A}_n^l}$ for the l^{th} ENFIS ($l = 1, 2, \ldots, L$) is $O(N^l M^l W^l)$ and that of deriving $\boldsymbol{d}_k^l$ is $O(N^{l+1} M^{l+1} W^{l+1})$ for the l^{th} ENFIS ($l = 1, 2, \ldots, L-1$). Thus, the overall computational complexity of Stage 2 of MEFNN is $O(\sum_{l=1}^{L} N^l M^l W^l)$ as well.

Together, the computational complexity of the overall system identification process of MEFNN given K input samples is $O(K \sum_{l=1}^{L} N^l M^l W^l)$.

In contrast, for a representative, conventional M-input W-output EFS that uses the RLS-based algorithm for consequent parameter learning, the overall computational complexity of its system identification process given K input samples is typically $O(KNM^2W)$ if the local learning approach is used and $O(KN^2M^2W)$ if the global learning approach is used [24], where N denotes the number of rules in the model.

III. Experimental Investigation

In this section, numerical examples based on a variety of benchmark problems are presented for demonstrating the performance of the proposed MEFNN. The algorithms were developed on Matlab2021b platform and the performance was evaluated on a desktop with dual core i7 processor 3.80 GHz×2 and 32.0 GB RAM. Unless specifically declared otherwise, the reported numerical results were obtained as the average of 10 Monte Carlo experiments to allow a certain degree of randomness and hence, a fair comparison.

A. Configuration

1) Data Description: In this study, a total of 24 popular numerical benchmark datasets for classification from UCI machine learning repository[1] and KEEL-dataset repository[2] are involved in performance evaluation, which include: 1) cardiotocography (CA); 2) wall-following robot navigation (WF); 3) gesture phase segmentation (GP); 4) optical recognition of handwritten digits (OR); 5) australia (AU); 6) balance (BA); 7) liver (LI); 8) magic (MG); 9) monk (MO); 10) pageblocks (PB); 11) pima (PI); 12) seismic (SE); 13) sonar (SO); 14) spectfheart (SH); 15) occupancy detection (OD); 16) multiple features (MF); 17) pen-based recognition of handwritten digits (PR); 18) abalone (AB); 19) image segmentation (IS); 20) phishing websites (PW); 21) spambase (SP); 22) mammography (MA); 23) texture (TE), and; 24) steel plates faults (SPF).

To evaluate the performance of MEFNN on high-dimensional classification problems, the following four remote sensing image sets for land-use classification are used, namely, 1) OPTIMAL-31 (OPT)[3]; 2) WHU-RS19 (WHU)[4]; 3) UCMerced (UCM)[5] and 4) RSSCN7 (RSS)[6]. Following the same procedure as described in [61], in this study, three mainstream DCNN models, namely, ResNet50 [62], DenseNet121 [63] and InceptionV3 [64], are employed for feature extraction after being fine-tuned on the NWPU45 dataset[7]. In running the experiments, each fine-tuned DCNN model will extract a 1024×1 dimensional feature vector from each image. The arithmetic mean of the three feature vectors will be used as the representation of the image for model training and testing.

To demonstrate the performance of MEFNN on large-scale, nonstationary, complex problems, the following two popular benchmark datasets for network intrusion detection, namely, NSLKDD [65] and UNSWNB15 [66] are also involved in experimental investigation. As a common practice, the categorical attributes of the two datasets have been converted to numerical ones by one-hot mapping in advance.

Furthermore, six widely used benchmark datasets are employed to test the performance of MEFNN on regression problems, which include four real-world regression problems, one Mackey-Glass time series prediction problem and one S&P500 closing price prediction problem. The four real-world problems are: 1) autos; 2) autompg; 3) delta ailerons, and; 4) california housing.

Key information of the 24 benchmark numerical classification datasets, four image classification datasets, two network intrusion detection datasets and six regression problems used in the numerical examples presented in this study is summarised in Supplementary Section C.

2) Parameter Setting for MEFNN: A key feature of MEFNN is that its multi-layered system structure is highly flexible and is self-evolving with data. Users are required to predetermine the number of base models/layers in the stacking

[1] Available at: https://archive.ics.uci.edu/ml/index.php
[2] Available at: https://sci2s.ugr.es/keel/datasets.php
[3] Available at: https://1drv.ms/u/s!Ags4cxbCq3lUguxW3bq0D0wbm1zCDQ
[4] Available at: https://captain-whu.github.io/BED4RS/
[5] Available at: http://weegee.vision.ucmerced.edu/datasets/landuse.html
[6] Available at: https://github.com/palewithout/RSSCN7
[7] Available at: https://www.tensorflow.org/datasets/catalog/resisc45

ensemble, L, the output size for each individual base model (except for the final one), namely, W^1, W^2, ..., W^{L-1} and the threshold δ_o that enables the base models to identify data patterns different from previously observed ones. However, it has to be stressed that these externally controlled parameters can be determined based on the users' preferences without any prior knowledge of the problem. In this study, the output sizes of the 1^{th} to $L-1^{th}$ base models are set uniformly the same as $W^1 = W^2 = \ldots = W^{L-1} = W_o$ for simplicity.

To demonstrate the proposed concept and general principles, in running the experiments, the number of layers, L of MEFNN is set as 2 unless specifically declared otherwise, and the other two externally controlled parameters, W_o and δ_o are fixed as $3W$ and e^{-3}, respectively. Due to its multilayer structure, MEFNN has stronger multi-level representation learning ability and more trainable parameters. To ensure that the stacking ensemble model is trained sufficiently, MEFNN is trained on the same training sets with shuffling for 200 epochs during the experiments.

It is worth noting that the parameter setting of MEFNN in this study only serves as a feasible option for users' consideration. However, it will be demonstrated through numerical examples presented in the rest of this section that MEFNN can achieve superior prediction performance on a wide range of benchmark classification and regression problems using this set of parameters, outperforming the SOTA alternatives. In practice, one can adjust the three externally controlled parameters (namely, L, W_o and δ_o) to maximise the prediction performance of MEFNN according to the nature of data. More experienced users may further consider using different output sizes for individual base models at different layers. To understand the impacts of the three parameters on the performance of MEFNN, a sensitivity analysis is carried out on the following four benchmark datasets, namely, 1) CA; 2) WF; 3) GP, and; 4) OR. The detailed analysis results are presented in Supplementary Section D. In addition, an ablation analysis is performed and presented in Supplementary Section E to demonstrate the performance improvement brought forward by the sigmoid function used in the consequent part of IF-THEN rules of MEFNN. Note that, the four datasets will not be used for performance demonstration presented in the rest of this section.

3) SOTA Methods for Comparison: In this study, the following 17 single-model SOTA algorithms are used for performance comparison: 1) SVM [67]; 2) KNN [68]; 3) sequential classifier (SEQ) [69]; 4) sequence-dictionary-based KNN classifier (SDKNN) [69]; 5) extreme learning machine (ELM) [70]; 6) MLP; 7) LSTM [71]; 8) probabilistic neural network (PNN) [72]; 9) eigenvalue classifier (EIG) [73]; 10) spherical approximation classifier (SPA) [74]; 11) self-adaptive fuzzy learning system (SALF) [26]; 12) multi-objective optimised self-organising fuzzy inference system (MOOSOFIS) [75]; 13) SEFIS [38]; 14) ESAFIS [31]; 15) PALM [44]; 16) eClass0 classifier [24], and; 17) eClass1 classifier [24].

In addition, the following five ensemble learning approaches are involved for performance comparison, which include: 1) random forest (RF) [21]; 2) stagewise additive modelling using a multi-class exponential loss function (SAMME) [17]; 3) fuzzily weighted adaptive boosting (FWAdaBoost) [49]; 4) XGBoost [76], and; 5) eEnsemble [46]. In this study, XGBoost uses decision tree (DT) as its base classifiers; two ensemble models are created with SAMME by using DT and KNN as the base classifiers, denoted as SAMMED and SAMMEK, respectively. Hence, a total of six ensemble models are involved in this study.

The parameter settings of the 17 single-model predictors and six ensemble models used for numerical experiments are given in Supplementary Section F.

B. Performance Demonstration on Numerical Classification Problems

First, the performance of MEFNN is evaluated on 10 benchmark classification datasets from KEEL, which include 1) AU; 2) BA; 3) LI; 4) MG; 5) MO; 6) PB; 7) PI; 8) SE; 9) SO, and; 10) SH. Following the common practice [77], [78], the maximum accuracy, mean accuracy and standard deviation obtained by MEFNN from ten-fold cross-validation on each dataset are tabulated in Supplementary Table S8. The following six algorithms: 1) EIG; 2) SPA; 3) SAFL; 4) MOOSOFIS; 5) SEFIS, and; 6) PALM, are involved in performance comparison under the same experimental protocol and the results are given in Supplementary Table S8. For better comparison, the results obtained by the three state-of-the-art classification approaches in the literature, namely, 7) Chebyshev polynomial broad learning system (CPBLS) [77], 8) compact fuzzy broad learning system (CFBLS) [78] and 9) highly interpretable deep fuzzy classifier (HIDFC) [79], are also reported in the same table. In addition, the performances of MEFNN with a three-layer structure ($L = 3$), denoted as MEFNN_3, on the 10 benchmark datasets are reported in Supplementary Table S8 to better demonstrate the proposed concept. The best results per datasets in terms of maximum accuracy and mean accuracy are highlighted in Supplementary Table S8 for visual clarity.

It can be observed from Supplementary Table S8 that MEFNN outperforms the nine alternative classification approaches in terms of maximum accuracy on five out of the 10 benchmark problems (namely, BA, MG, MO, PI and SO) and its mean accuracy surpasses others on two out of the 10 problems (namely, MG and MO). The average classification accuracy of MEFNN over the 10 problems is 0.8684, which is ranked the first place among the 10 classification approaches involved in this numerical example. With a three-layer structure, MEFNN_3 achieves greater mean accuracy than MEFNN on three benchmark problems (namely, BA, MG and MO) and its maximum accuracy surpasses MEFNN on SH dataset. Its average classification accuracy over the 10 problems is 0.8596, which is slightly lower than the two-layer MEFNN but greater than the nine competitors. In contrast, CPBLS and CFBLS are the two best performing ones among the nine competitors, respectively, with the average classification accuracies of 0.8558 and 0.8544.

Next, the classification performance of MEFNN is evaluated on the following 10 benchmark problems from UCI: 1) OD; 2) MF; 3) PR; 4) AB; 5) IS; 6) PW; 7) SP; 8) MA; 9) TE,

TABLE I
OVERALL CLASSIFICATION PERFORMANCES AND THE RESPECTIVE RANKS OF THE 23 CLASSIFICATION APPROACHES ON 10 BENCHMARK CLASSIFICATION PROBLEMS FROM UCI

Algorithm	Acc		$Bacc$	
	$Mean$	$Rank$	$Mean$	$Rank$
MEFNN	0.8928	**5.00**	0.8686	**5.30**
$MEFNN_3$	0.8861	7.80	0.8543	8.90
SVM	0.8874	7.10	0.8525	9.40
KNN	0.8773	10.45	0.8570	9.65
SEQ	0.7902	14.40	0.8033	12.60
SDKNN	0.7930	16.60	0.8081	15.00
ELM	0.7170	15.40	0.7008	14.40
MLP	0.8554	13.80	0.8197	14.60
LSTM	0.8832	7.80	0.8559	8.60
PNN	0.8477	14.15	0.8616	10.00
EIG	0.8413	15.70	0.8150	16.00
SAFL	0.8882	6.50	0.8561	7.65
MOOSOFIS	0.8731	12.70	0.8526	11.00
SEFIS	0.6711	21.70	0.6223	22.00
ESAFIS	0.8828	9.10	0.8501	9.75
PALM	0.8580	12.50	0.8053	13.90
eClass0	0.7350	20.90	0.7604	18.80
eClass1	0.8620	12.60	0.8111	13.90
RF	0.8679	9.00	0.8299	11.10
FWAdaBoost	0.8783	10.40	0.8312	11.00
SAMMED	0.8401	14.50	0.8070	14.90
SAMMEK	0.8754	11.25	0.8550	10.45
XGBoost	**0.9001**	6.65	**0.8773**	7.10

and; 10) SPF, and compared with the 21 single-model and multi-model classification approaches mentioned in Section III.A (SPA is not involved in this example due to its extremely low computational efficiency on high dimensional problems). In running the experiments, for OD, PR and IS datasets, their original training-testing splits are used. For the other seven datasets, 50% of data samples are randomly selected to construct the training sets and the remaining 50% are used as the validation sets [75]. The detailed classification results in terms of accuracy (Acc) and balanced accuracy ($BAcc$) are given in Supplementary Tables S9 and S10, respectively. Similarly, the results obtained by $MEFNN_3$ are also reported in Supplementary Tables S9 and S10. The mean accuracies and mean balanced accuracies of the 23 classification approaches (including MEFNN and $MEFNN_3$) and their respective ranks from the best to the worst over the 10 benchmark problems are given by Table I.

One can see from Table I that MEFNN outperforms 20 single-model and multi-model classification approaches on the 10 benchmark problems in terms of average accuracy and average balanced accuracy, and is only outperformed by XGBoost. Meanwhile, MEFNN is ranked the top place on both accuracy measures, suggesting that the predicted labels produced by MEFNN are more accurate than other approaches in the majority of cases. On the other hand, despite that $MEFNN_3$ outperforms MEFNN on a number of benchmark problems in terms of mean (balanced) accuracy and maximum accuracy, it can be seen from this numerical example that its average classification accuracy over the 20 benchmark problems is slightly lower than MEFNN. This suggests that, with the recommended parameter setting, the two-level distributed representations learned by MEFNN from data are sufficiently discriminative for accurately classifying the unlabelled testing data of the 20 benchmark datasets used for performance demonstration. However, it has to be stressed that the optimal parameter setting for MEFNN always varies from problem to problem depending on the nature of data.

To examine the statistical significance of the superior performance achieved by MEFNN, over the 21 comparative approaches involved in this example (excluding $MEFNN_3$), pairwise Wilcoxon rank tests [80] are conducted, and the outcomes in terms of p-value are reported in Supplementary Table S11, where the cascaded classification results by each classification approach across the 10 Monte-Carlo experiments are used. One can see from Supplementary Table S8 that 86.19% of the p-values returned by the pairwise Wilcoxon tests are below the level of significance specified by $\alpha = 0.05$. The statistical analysis demonstrates that the performance of MEFNN is, indeed, significantly better than the 21 alternative classification approaches.

C. Performance Demonstration on Image Classification Problems

Then, the performance of MEFNN on high-dimensional problems are tested on the four image classification problems. In running the experiments, the training-testing split ratio of OPT is set to 8:2; and for WHU, UCM and RSS datasets, two different split ratios are considered for each, namely, 4:6 and 6:6, 5:5 and 8:2, and, 2:8 and 5:5, respectively, by following the common practice in the literature [81]. The classification accuracies of MEFNN on the four visual benchmark problems under different split ratios are reported in Table II. The results obtained by MEFNN (in terms of average classification accuracy and standard deviation) and a selected group of the SOTA approaches from the literature are given in Table II for comparison. Note that the results by comparative approaches are obtained from [75], [82] directly. It can be seen from Table II that MEFNN is able to achieve great classification performance on these visual classification problems, outperforming many DNN-based approaches.

D. Performance Demonstration on Network Intrusion Detection Problems

Furthermore, the performance of MEFNN on large-scale, nonstationary and complex problems is evaluated on the two aforementioned network instruction detection datasets and compared with six single-model EFS models and two ensemble EFS models aforementioned. In running the experiments, the original training-testing splits of the two datasets are kept. To facilitate simulation, 10% of the training and testing samples are randomly selected and used in each experiment. The numerical results obtained by MEFNN and the eight competitors on the two datasets are reported in Table III in terms of Acc and $BAcc$. One can see from Table III that MEFNN outperforms all its EFS competitors by offering the greatest Acc and $BAcc$ on both datasets, showing its stronger capability to handle highly challenging problems.

TABLE II
PERFORMANCE COMPARISON ON FOUR VISUAL CLASSIFICATION PROBLEMS

Algorithm	OPT	WHU		UCM		RSS	
		4:6	6:4	5:5	8:2	2:8	5:5
MEFNN	1.0000	0.9713	0.9791	0.9650	0.9743	0.9169	0.9287
MOOSOFIS [75]	0.9989	0.9630	0.9749	0.9598	0.9693	0.8908	0.9157
CaffeNet [81]	-	0.9511	0.9624	0.9398	0.9502	0.8557	0.8826
VGG-VD-16 [81]	-	0.9544	0.9605	0.9414	0.9521	0.8398	0.8718
GoogLeNet [81]	-	0.9312	0.9471	0.9370	0.9431	0.8255	0.8584
SalM^{3}LBP-CLM [83]	-	0.9421	0.9575	0.9535	0.9638	-	
ARCNet-VGG16 [84]	0.9270	0.9750	0.9975	0.9681	0.9912	-	-
GBNet [85]	0.9328	0.9705	0.9857	0.9732	0.9925	-	-
EfficientNet-B3-Basic [86]	0.9476	0.9728	0.9768	0.9763	0.9873	0.9206	0.9439
EfficientNet-B3-Attn-2 [86]	0.9586	0.9860	0.9868	0.9790	0.9921	0.9330	0.9617
MSDS [87]	-	-	0.9761	-	0.9696	-	-
MLDS [87]	-	-	0.9829	-	0.9788	-	-
RANet [82]	0.9461	0.9798	0.9897	0.9780	0.9927	-	-

TABLE III
PERFORMANCE COMPARISON ON TWO BENCHMARK PROBLEMS FOR NETWORK INTRUSION DETECTION

Algorithm	NSLKDD		UNSWNB15	
	Acc	$BAcc$	Acc	$BAcc$
MEFNN	**0.7861**	**0.8029**	**0.8334**	**0.8163**
SAFL	0.7737	0.7918	0.8139	0.7932
SEFIS	0.7721	0.7874	0.7430	0.7234
ESAFIS	0.7421	0.7704	0.7714	0.7456
PALM	0.7644	0.7842	0.8068	0.7847
eClass0	0.7177	0.7470	0.7250	0.7129
eClass1	0.5668	0.5000	0.5511	0.5000
eEnsemble	0.7018	0.7321	0.7223	0.7090
FWAdaBoost	0.7741	0.7924	0.8301	0.8136

TABLE IV
PERFORMANCE COMPARISON ON MACKEY-GLASS TIME SERIES PROBLEM

Algorithm	$NDEI$	#(Rules)	t_{exe}
MEFNN	**0.0914**	(4.4, 13.1)	164.2392
PSO-ALMMo	0.1910	8	314.3214
SAFL	0.1048	20	0.1868
CEFNS	0.2635	5	0.4368
eTS	0.2141	4	21.9745
ESAFIS	0.2487	6	2.7656
SAFIS	0.2925	4	0.4375
RMCEFS	0.1172	5	0.3432
OS-Fuzzy-ELM	0.2991	5	0.9253
PANFIS	0.2847	33	4.8679
GENEFIS	0.1198	42	4.9694
PALM	0.1380	18	0.7771
eFuMo	0.1388	41	-
EFS-SLAT	0.1140	8	-
SEFS	0.1287	4	0.3510
LEOA [90]	0.2480	42	144.7818
eGAUSS+ [91]	0.2728	25	$\sim$5
HiPCA [92]	0.2495	25	$\sim$14
InFuR [93]	0.1545	22	-

E. Performance Demonstration on Regression Problems

In this section, the regression performance of MEFNN is examined on widely used benchmark problems and compared with a variety of mainstream EFSs.

The performances of MEFNN on the four real-world benchmark problems, namely, 1) autos; 2) autompg; 3) delta ailerons, and; 4) california housing, in terms of root mean squared error ($RMSE$) and number of rules in the rule base ($\#(Rules)$) are reported in Supplementary Table S12, where the results by comparative approaches, including 1) PSO-ALMMo [35]; 2) SB-ALMMo [88]; 3) CEFNS [43]; 4) DENFIS [25]; 5) eTS [29]; 6) ESAFIS [31]; 7) SAFIS [30]; 8) RMCEFS [36], and; 9) OS-Fuzzy-ELM [89], are obtained from the literature.

One can see from Supplementary Table S12 that MEFNN outperforms all nine mainstream EFSs on the automgp and california housing problems by producing the most accurate predictions (with lowest $RMSE$), and its performance on the other two datasets are also well above the average. This example demonstrates the great potential of MEFNN on real-world regression problems.

Next, the performance of MEFNN is evaluated on the widely-used Mackey-Glass chaotic time series problem following the standard experimental protocol [34], [36], [90]. The result obtained by MEFNN is reported in Table IV in terms of non-dimensional error index ($NDEI$), number of rules in the rule base ($\#(Rules)$) and execution time (t_{exe}). The results by the mainstream EFSs are obtained directly from the literature [26], [34], [36], [91]–[93] and reported in the same table. It is shown by Table IV that MEFNN surpasses the SOTA competitors on the Mackey-Glass problem with the lowest $NDEI$ (0.0914), which further demonstrates that its superior performance on regression tasks.

Finally, MEFNN is tested on the S&P500 problem. The prediction performance of MEFNN on the testing data following the standard test-then-train protocol is reported in Supplementary Table S13 (in terms of $NDEI$ and $\#(Rules)$). The prediction performance of MEFNN without online training (following the same setting used in other examples presented in this section) is also reported in the same table for comparison. To differentiate between the two results, MEFNN that follows the test-then-train protocol is re-denoted as MEFNN*. The prediction results by MEFNN and MEFNN are depicted in Supplementary Fig. S2 for better comparison. In addition, results by selected SOTA approaches on this problem follow-

ing the commonly used test-then-train protocol are tabulated in Table Supplementary Table S13 for comparison.

Supplementary Table S13 shows that MEFNN* demonstrates greater prediction performance on the S&P500 problem with a lower $NDEI$ value of 0.0162, which is ranked the third place among the regression approaches involved in this example. In addition, it can be seen from Supplementary Table S13 and Fig. S2 that MEFNN can effectively utilise new streaming data samples to self-update its knowledge base and self-improve its predictions with the $NDEI$ value improved from 0.0200 to 0.0162. This example demonstrates the potential of MEFNN as a powerful tool for handling data streams. The IF-THEN rules learned by MEFNN from S&P500 problem during one particular experiment are presented in Supplementary Table S14, where one can see that the learned meta-level knowledge base of MEFNN can be visualised in a meaningful, human-interpretable form thanks to the use of ENFISs as its base components. This is useful for developing transparent inference tools.

To summarise, the systematic experimental studies over a wide variety of benchmark classification and regression problems presented in this section collectively and consistently demonstrate the superior performance of MEFNN in comparison to the SOTA approaches in terms of prediction accuracy. Numerical examples based on 20 popular numerical problems presented in Section III.B demonstrate that MEFNN with the default parameter setting can achieve great classification accuracy surpassing, or at least on par with the SOTA competitors. Numerical examples on four visual classification problems and two network intrusion detection problems presented in Sections III.C and III.D show the strong capability of MEFNN to handle high-dimensional, large-scale, complex problems. Thanks to the multi-level distributed representation learned from data, MEFNN achieves greater classification accuracy on these challenging benchmark problems than alternative EFS-based comparative models involved in experimental comparison. Last, MEFNN is tested on six benchmark regression problems, and numerical results show the great potential of MEFNN in solving real-world regression problems whist offering high-level transparency and interpretability. On the other hand, as the main purpose of this paper is to demonstrate the concept and general principles of the proposed stacking ensemble model, all the numerical results of MEFNN are obtained with the same parameter setting, thereby ensuring a fair comparison. The structure of MEFNN is, in fact, highly flexible, one can easily increase the depth by adding more base models in the stacking ensemble, control the size of each layer by changing the threshold, δ_o and increase/reduce the amount of information exchanged between successive layers by adjusting W_o. Therefore, there is still a large space for performance improvement by adjusting the externally controlled parameters according to the nature of data.

IV. Conclusion

This paper has presented a new meta-level ensemble learning model, composed of multiple cascading simpler EFSs to learn multi-level nonlinear distributed representation from data. The resultant multilayer evolving fuzzy neural network (MEFNN) learns from data on a sample-by-sample basis to self-organise the underlying multilayer system structure and self-update the network parameters. Systematic comparative investigations have demonstrated the superior performance of MEFNN on various challenging classification and regression problems, outperforming the SOTA approaches.

A number of open issues remain that require further considerations. First, theoretical analysis on the stability and convergence of the proposed MEFNN is not conducted in this study. Although there have been a number of works analysing the convergence of backpropagation and the stability of EFSs, the use of backpropagation in EFS-based meta-level ensemble models has not been explored before. Thus, this will be an important direction for future research. Second, compared with single-model EFSs, MEFNN requires larger amounts of training data (or being trained repeatedly with the same data) to reach practically excellent performance. This is fundamentally due to the use of the error backpropagation algorithm for consequent parameter updating, a limitation shared with many SOTA methods. As the ambiguity in defining the error functions for individual layers poses a challenge to the use of RLS-based algorithms in MEFNN, it would be worth exploring different options to address this aspect, better facilitating the consequent parameter training. Third, numerical examples presented in this study demonstrate that MEFNN with the same fixed parameter setting outperforms the SOTA approaches on many classification and regression problems in terms of prediction accuracy. However, predetermining the externally controlled parameters for MEFNN may still be a challenging task, especially for these less experienced users. Therefore, it would be helpful to explore the possibilities of deriving some of the parameters directly from data, for example, MEFNN may self-determine to increase or decrease the number of layers based on the nature of data. Fourth, this study only explores the use of sigmoid function in the consequent part of IF-THEN rules of MEFNN and compared it with the classical linear function. It would be interesting to see how MEFNN might perform with other types of nonlinear activation function, particularly, rectified linear unit, which is computationally cheaper and converges faster than sigmoid. Fifth, the base model ENFIS employed in this research only possesses the essential rule adding scheme. It would be helpful to introduce other potentially useful schemes such as pruning, merging and splitting in an effort to learn a more compact, meaningful rule base from data, thereby improving the overall prediction performance of MEFNN. Last, the numerical results presented in this paper are focused on classification and regression problems. MEFNN, in fact, is a generic, flexible approach for multi-level representation learning and has great potential for solving more complex problems concerning visual and audio information. The utilisation of MEFNN in other types of applications, such as dimensionality reduction, image feature extraction, etc., will be further explored in future works.

References

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

[2] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Comput. Sci. Rev.*, vol. 40, p. 100379, 2021.

[3] G. Li et al.,"A deep learning based image enhancement approach for autonomous driving at night," *Knowledge-Based Syst.*, vol. 213, p. 106617, 2021.

[4] H. Chen et al., "The rise of deep learning in drug discovery," *Drug Discov. Today*, vol. 23, no. 6, pp. 1241–1250, 2018.

[5] X. Ding et al., "Deep learning for event-driven stock prediction," in *International Joint Conference on Artificial Intelligence*, 2015, pp. 2327–2333.

[6] G. Hinton, "Learning multiple layers of representation," *Trends Cogn. Sci.*, vol. 11, no. 10, pp. 428–434, 2007.

[7] Z. Zhou and J. Feng, "Deep forest," *Natl. Sci. Rev.*, vol. 6, no. 1, pp. 74–86, 2019.

[8] P. Angelov et al., "Explainable artificial intelligence: an analytical review," *WIREs Data Min. Knowl. Discov.*, pp. 1–13, 2021.

[9] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[10] J. Feng, Y. Yu, and Z. Zhou, "Multi-layered gradient boosting decision trees," in *Advances in Neural Information Processing Systems*, 2018, pp. 3551–3561.

[11] X. Gu, "Multilayer ensemble evolving fuzzy inference system," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 8, pp. 2425–2431, 2021.

[12] D. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 505, pp. 241–255, 1992.

[13] A. Pernia-Espinoza et al., "Stacking ensemble with parsimonious base models to improve generalization capability in the characterization of steel bolted components," *Appl. Soft Comput.*, vol. 70, pp. 737–750, 2018.

[14] E. Lughofer, M. Pratama, and I. Skrjanc, "Online bagging of evolving fuzzy systems," *Inf. Sci. (Ny).*, vol. 570, pp. 16–33, 2021.

[15] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 421, pp. 123–140, 1996.

[16] Y. Jung, J. Goetz, and A. Tewari, "Online multiclass boosting," in *Advances in Neural Information Processing Systems*, 2017, pp. 920–929.

[17] J. Zhu et al., "Multi-class AdaBoost," *Stat. Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[18] Y. Xia, K. Chen, and Y. Yang, "Multi-label classification with weighted classifier selection and stacked ensemble," *Inf. Sci. (Ny).*, vol. 557, pp. 421–442, 2021.

[19] P. Vincent et al., "Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[20] M. Akhtar, A. Ekbal, and E. Cambria, "How intense are you? predicting intensities of emotions and sentiments using stacked ensemble," *IEEE Comput. Intell. Mag.*, vol. 15, no. 1, pp. 64–75, 2020.

[21] L. Breiman, "Random forests," *Mach. Learn. Proc.*, vol. 45, no. 1, pp. 5–32, 2001.

[22] A. Taherkhani, G. Cosma, and T. McGinnity, "AdaBoost-CNN: an adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning," *Neurocomputing*, vol. 404, pp. 351–366, 2020.

[23] M. Ribeiro and L. dos Santos Coelho, "Ensemble approach based on bagging, boosting and stacking for short-term prediction in agribusiness time series," *Appl. Soft Comput.*, vol. 86, p. 105837, 2020.

[24] P. Angelov, *Autonomous learning systems*: *from data streams to knowledge in real time*. John Wiley & Sons, Ltd., 2012.

[25] N. Kasabov, *Evolving connectionist systems*: *the knowledge engineering approach*. Springer Science & Business Media, 2007.

[26] X. Gu and Q. Shen, "A self-adaptive fuzzy learning system for streaming data prediction," *Inf. Sci. (Ny).*, vol. 579, pp. 623–647, 2021.

[27] I. Skrjanc et al., "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey," *Inf. Sci. (Ny).*, vol. 490, pp. 344–368, 2019.

[28] E. Lughofer, "Evolving fuzzy and neuro-fuzzy systems: fundamentals, stability, explainability, useability, and applications," in *HANDBOOK ON COMPUTER LEARNING AND INTELLIGENCE: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation*, 2022, pp. 133–234.

[29] P. Angelov and D. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst. Man, Cybern. - Part B Cybern.*, vol. 34, no. 1, pp. 484–498, 2004.

[30] H. Rong et al., "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets Syst.*, vol. 157, no. 9, pp. 1260–1275, 2006.

[31] H. Rong et al., "Extended sequential adaptive fuzzy inference system for classification problems," *Evol. Syst.*, vol. 2, no. 2, pp. 71–82, 2011.

[32] E. Lughofer et al., "Generalized smart evolving fuzzy systems,"*Evol. Syst.*, vol. 6, no. 4, pp. 269–292, 2015.

[33] D. Dovzan, V. Logar, and I. Skrjanc, "Implementation of an evolving fuzzy model (eFuMo) in a monitoring system for a waste-water treatment process," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1761–1776, 2015.

[34] D. Ge and X. Zeng, "A self-evolving fuzzy system which learns dynamic threshold parameter by itself," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 8, pp. 1625–1637, 2018.

[35] X. Gu, Q. Shen, and P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5352–5363, 2021.

[36] H. Rong, Z. Yang, and P. Wong, "Robust and noise-insensitive recursive maximum correntropy-based evolving fuzzy system," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 9, pp. 2277–2284, 2019.

[37] D. Ge and X. Zeng, "Learning data streams online - an evolving fuzzy system approach with self-learning/adaptive thresholds," *Inf. Sci. (Ny).*, vol. 507, pp. 172–184, 2020.

[38] Z. Yang et al., "Statistically evolving fuzzy inference system for non-Gaussian noises," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 4, pp. 2649–2664, 2022.

[39] G. Leng, T. McGinnity, and G. Prasad, "An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network," *Fuzzy Sets Syst.*, vol. 150, no. 2, pp. 211–243, 2005.

[40] D. Leite, P. Costa, and F. Gomide, "Evolving granular neural networks from fuzzy data streams," *Neural Networks*, vol. 38, pp. 1–16, 2013.

[41] M. Pratama, S. Anavatti, and E. Lughofer, "Genefis: toward an effective localist network," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 3, pp. 547–562, 2014.

[42] M. Pratama et al., "PANFIS: a novel incremental learning machine," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 55–68, 2014.

[43] R. Bao et al., "Correntropy-based evolving fuzzy neural system," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 3, pp. 1324–1338, 2018.

[44] M. Ferdaus et al., "PALM: an incremental construction of hyperplanes for data stream regression," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 11, pp. 2115–2129, 2019.

[45] X. Gu et al., "Autonomous learning for fuzzy systems: a review," *Artif. Intell. Rev.*, pp. 1–47, 2022.

[46] J. 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.

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

[48] M. Pratama et al., "Online tool condition monitoring based on parsimonious ensemble+," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 664–677, 2020.

[49] X. Gu and P. Angelov, "Multi-class fuzzily weighted adaptive boosting-based self-organising fuzzy inference ensemble systems for classification," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 9, pp. 3722–3735, 2022.

[50] E. Lughofer and M. Pratama, "Online sequential ensembling of predictive fuzzy systems," *Evol. Syst.*, vol. 13, no. 2, pp. 361–386, 2022.

[51] M. Pratama, W. Pedrycz, and G. Webb, "An incremental construction of deep neuro fuzzy system for continual learning of nonstationary data streams," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 7, pp. 1315–1328, 2020.

[52] F. Bordignon and F. Gomide, "Uninorm based evolving neural networks and approximation capabilities," *Neurocomputing*, vol. 127, pp. 13–20, 2014.

[53] H. Huang et al., "Recursive least mean dual p-power solution to the generalization of evolving fuzzy system under multiple noises," *Inf. Sci. (Ny).*, vol. 609, pp. 228–247, 2022.

[54] Z. Sun, K. Au, and T. Choi, "A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 37, no. 5, pp. 1321–1331, 2007.

[55] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *Chinese control and decision conference*, 2019, pp. 1836–1841.

[56] C. Garcia et al., "Evolvable fuzzy systems from data streams with missing values: with application to temporal pattern recognition and cryptocurrency prediction," *Pattern Recognit. Lett.*, vol. 128, pp. 278–282, 2019.

[57] J. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[58] C. Lin and Y. Lu, "A neural fuzzy system with linguistic teaching signals," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, pp. 169–189, 1995.

[59] R. Yin et al., "A rule-based deep fuzzy system with nonlinear fuzzy feature transform for data classification," *Inf. Sci. (Ny).*, vol. 633, pp. 431–452, 2023.

[60] P. de Campos Souza, E. Lughofer, and A. Guimaraes, "An interpretable evolving fuzzy neural network based on self-organized direction-aware data partitioning and fuzzy logic neurons," *Appl. Soft Comput.*, vol. 112, p. 107829, 2021.

[61] X. Gu et al., "A self-training hierarchical prototype-based ensemble framework for remote sensing scene classification," *Inf. Fusion*, vol. 80, pp. 179–204, 2022.

[62] K. He et al., "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[63] G. Huang et al., "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[64] C. Szegedy et al., "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[65] M. Tavallaee et al., "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[66] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J.*, vol. 25, no. 1–3, pp. 18–31, 2016.

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

[68] P. Cunningham and S. Delany, "K-nearest neighbour classifiers," *Mult. Classif. Syst.*, vol. 34, pp. 1–17, 2007.

[69] R. Patro et al., "Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data," Int. J. *Remote Sens.*, vol. 40, no. 13, pp. 4996–5024, 2019.

[70] G. Huang et al., "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 42, no. 2, pp. 513–529, 2012.

[71] S. Hochreiter and J. Urgen Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] D. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[73] U. Erkan, "A precise and stable machine learning algorithm: eigenvalue classification (EigenClass)," *Neural Comput. Appl.*, vol. 33, no. 10, pp. 5381–5392, 2021.

[74] D. Li and D. Dunson, "Classification via local manifold approximation," *Biometrika*, vol. 107, no. 4, pp. 1013–1020, 2020.

[75] X. Gu et al., "Multi-objective evolutionary optimisation for prototype-based fuzzy classifiers," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2022.3214241, 2022.

[76] T. Chen and C. Guestrin, "Xgboost: a scalable tree boosting system," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[77] S. Feng, B. Wang, and C. Chen, "Chebyshev polynomial broad learning system," in *IEEE International Conference on Information, Cybernetics, and Computational Social Systems*, 2021, pp. 1–6.

[78] S. Feng et al., "On the accuracy-complexity trade-off of fuzzy broad learning system," *IEEE Trans. Fuzzy Syst.*, vol. 29, no. 10, pp. 2963–2974, 2021.

[79] 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, 2018.

[80] F. Wilcoxon, "Individual comparisons of grouped data by ranking methods," *J. Econ. Entomol.*, vol. 39, no. 6, pp. 269–270, 1946.

[81] G. Xia et al., "AID: a benchmark dataset for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, 2017.

[82] X. Wang et al., "Relation-attention networks for remote sensing scene classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 15, pp. 422–439, 2022.

[83] X. Bian et al., "Fusing local and global features for high-resolution scene classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 10, no. 6, pp. 2889–2901, 2017.

[84] Q. Wang, S. Liu, and J. Chanussot, "Scene classification with recurrent attention of VHR remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1155–1167, 2019.

[85] H. Sun et al., "Remote sensing scene classification by gated bidirectional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 82–96, 2020.

[86] H. Alhichri et al., "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE Access*, vol. 9, pp. 14078–14094, 2021.

[87] F. Hu et al., "Mining deep semantic representations for scene classification of high-resolution remote sensing imagery," *IEEE Trans. Big Data*, vol. 6, no. 3, pp. 522–536, 2020.

[88] X. Gu and P. Angelov, "Self-boosting first-order autonomous learning neuro-fuzzy systems," *Appl. Soft Comput.*, vol. 77, 2019.

[89] H. Rong et al., "Online sequential fuzzy extreme learning machine for function approximation and classification problems," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 4, pp. 1067–1072, 2009.

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

[91] I. Skrjanc, "Cluster-volume-based merging approach for incrementally evolving fuzzy Gaussian clustering-eGAUSS+," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 9, pp. 2222–2231, 2020.

[92] D. Dovzan and I. Skrjanc, "Fuzzy space partitioning based on hyperplanes defined by eigenvectors for Takagi-Sugeno fuzzy model identification," *IEEE Trans. Ind. Electron.*, vol. 67, no. 6, pp. 5144–5153, 2020.

[93] S. Blazic and I. Skrjanc, "Incremental fuzzy c-regression clustering from streaming data for local-model-network identification," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 4, pp. 758–767, 2020.