A Chunk-Wise Learning Evolving Autonomous Fuzzy System for Intelligent Data Stream Prediction

Xiaowei Gu, Qiang Ni and Qiang Shen

***Abstract*—First-order evolving fuzzy systems (EFSs) typically self-organise and self-evolve their structure and parameters from non-stationary data streams on a sample-by-sample basis to approximate the changing data patterns in real time. However, processing individual data samples sequentially is computationally inefficient for large-scale data streams and makes EFSs more vulnerable to temporal fluctuations in data patterns, causing their prediction performances to deteriorate. To overcome this limitation, a novel chunk-wise learning evolving autonomous fuzzy system (CLEAF) is proposed in this paper. CLEAF identifies the antecedent parameters of fuzzy IF-THEN rules as prototypes from data streams via chunk-wise online clustering, and updates the associated consequent parameters using a modified chunk-wise fuzzily weighted recursive least squares algorithm that treats every data chunk as a single unit for calculation. Thanks to the novel chunk-wise learning mechanism, CLEAF attains greater computational efficiency and achieves better approximation with less fuzzy rules. Furthermore, to better approximate nonlinear, complex problems and further enhance the prediction performance, both the input and target output variables are leveraged in data density calculation for fuzzy rule identification, enabling CLEAF to more effectively capture the underlying data patterns and their overlaps. Numerical experiments on popular benchmark datasets demonstrate the superior computational efficiency and high-level prediction accuracy of CLEAF, outperforming state-of-the-art approaches.**

***Index Terms*—chunk-wise learning; data density; data stream; evolving fuzzy system.**

I. Introduction

WITH the rapid advancement of electronic and Internet technologies enormous volumes of digital data are produced from various aspects of daily life, including financial transactions, sensing networks, and social media platforms, often in the form of data streams [1]. Analysing and interpreting data streams in real time is crucial for a wide range of real-life applications, e.g., financial investment, business analysis, fraud detection and network intrusion detection [2].

Machine learning has revolutionised data processing and interpretation by revealing complex patterns and enabling automated decision-making [3]–[5]. However, learning and modelling non-stationary data streams remains a significant challenge. Most traditional machine learning techniques are designed for processing static data in offline setting [6]. Their computational complexity increases significantly on large-scale, high-dimensional data, and they lack the ability to continuously self-adapt and self-improve from new data after offline training is completed. To effectively handle streaming data with dynamically changing characteristics, machine learning models need flexible system structures that support incremental learning, and the associated learning algorithms should efficiently process new data, ideally in a "one pass" manner, while adhering to the time and memory constraints [7].

There are two major types of online learning algorithms, namely, 1) incremental [8] and 2) evolving [9], [10]. Evolving fuzzy systems (EFSs), as a key component of computational intelligence, are widely recognised as one of the predominant approaches for approximating real-time non-stationary problems [11]. EFSs are a special class of fuzzy systems designed to simultaneously self-organise and self-adapt the system structures and parameters online from data streams. They effectively capture the dynamical behaviours and underlying patterns of data streams and represent the learned knowledge in the form of human-interpretable IF-THEN fuzzy production rules [12]. Thanks to the rule-based system structures and human-like fuzzy reasoning mechanisms, EFSs provide greater transparency and interpretability, making them one of the most promising approaches toward explainable AI [13].

Since the initial concept of EFSs was firstly conceived around the beginning of the 21^{st} century [9], [10], it has received great attentions. To date, many EFSs have been developed and successfully implemented in real-life time-critical applications, demonstrating great success in dynamical environments [2], [12]. The most representative EFSs include, but are not limited to, dynamic evolving neural-fuzzy inference system [9], evolving Takagi-Sugeno system [10], sequential adaptive fuzzy inference system [14], online sequential fuzzy extreme learning machine [15], extended sequential adaptive fuzzy inference system [16], meta-cognitive neuro-fuzzy inference system [17], evolving granular neural network [18], parsimonious network based on fuzzy inference system [19], generic evolving neuro-fuzzy inference system [20], self-evolving fuzzy system [21], correntropy-based evolving fuzzy neural system [22], parsimonious learning machine [23], evolving Cauchy system [24], evolving fuzzy system with self-learning/adaptive thresholds [25], self-adaptive fuzzy learning system [26], statistically evolving fuzzy inference system [27], dynamic evolving fuzzy system [28] and X-Fuzz [29].

Different EFSs may employ different rule generation approaches to identify the antecedent parts of the IF-THEN rules from data streams when unfamiliar data patterns are

X. Gu is with the Department of Computer Science, University of Surrey, Guildford, GU2 7XH, UK. email: xiaowei.gu@surrey.ac.uk.

Q. Ni is with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK. email: q.ni@lancaster.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, 2025; revised XXXX XX, 2025.

observed, which include density/potential criterion [24], [26], [30], distance criterion [9], [22], [23], firing strength criterion [21], [31], statistical contribution criterion [16], [19] and error criterion [22], [27], leading to highly diverse behaviours and prediction performances [2]. Most first-order EFSs use recursive least squares (RLS)-based algorithms for consequent parameters learning [9], [10]. Among these, the fuzzily weighted RLS (FWRLS) algorithm has been the most popular one and is widely recognised as the golden standard for training the consequent parameters of EFSs [10], [30].

There are three inherent issues associated with the structure evolving and parameter learning mechanisms of conventional first-order EFSs that limit their performances on large-scale, high-dimensional, complex problems [32]. First, most EFSs identify fuzzy rules from data streams in a sample-wise manner. As data samples are processed individually, EFSs may not be able to adequately capture the complex multi-modal data distributions and become more sensitive towards the temporary fluctuations in data patterns. This potentially leads to more fuzzy rules being identified from data than necessary. Second, EFSs commonly use RLS-based algorithms to update the consequent parameters of fuzzy rules on a sample-by-sample basis. Such approach can be computationally expensive, particularly when the dimensionality of data is high or the fuzzy rule base is large, which is often necessary for handling data with complex structures. Third, EFSs primarily focus on the divergences between data samples in the input space only, often overlooking the divergences in the output space. This can reduce their ability to distinguish overlapping local data patterns that correspond to clearly different outputs. While many EFSs incorporate mechanisms to detect and prune redundant fuzzy rules [19], [20], [27], they often rely on conservative pruning criteria to mitigate the risk of catastrophic forgetting. Consequently, rule redundancy triggered by temporary fluctuations remains a persistent issue, leading to computational wastage and model degradation. Therefore, despite efforts to enhance the computational efficiency and prediction performance of first-order EFSs on large-scale, high-dimensional, complex problems, e.g., utilising compressive sensing techniques to reduce the dimensionality of consequent parts [11], [33], building ensemble models with parallel or sequential architectures to distribute the computation [33]–[35], the conventional sample-wise learning approach remains a significant bottleneck.

In short, conventional first-order EFSs provide strong adaptability and approximation capability for non-stationary data streams, together with computationally and memory-efficient structure evolving and parameter learning, and a transparent rule-based framework that supports human-interpretable reasoning. These strengths make them powerful tools for real-time data-stream processing. However, they also have notable limitations, particularly when handling large-scale, high-dimensional, or complex problems. These limitations include high sensitivity to temporary fluctuations in data patterns, reduced computational efficiency in consequent parameter update, and limited ability to distinguish overlapping input patterns associated with different outputs. These issues highlight the demand for more advanced learning schemes that facilitate learning from data streams and further improve the computational efficiency of first-order EFSs.

In this paper, a novel human-interpretable Chunk-wise Learning Evolving Autonomous Fuzzy system (CLEAF) that overcomes the aforementioned limitations is proposed for streaming data modelling and approximation. Different from conventional first-order EFSs [25]–[27] and multi-model ensembles [35]–[38] that rely on sample-wise learning, CLEAF is specifically designed to perform structure evolving and parameter updating from data streams on a chunk-by-chunk basis. While chunk-wise learning has demonstrated proven effectiveness in online data stream processing for prototype-based models [39], this paradigm remains largely unexplored in first-order EFSs. By treating each data chunk as a single processing unit, CLEAF processes and self-adapts to streaming data in a highly efficient manner with greater scalability. The chunk-wise learning also allows CLEAF to capture the underlying data distributions more effectively and acquire more comprehensive temporal and spatial information from data streams, enhancing its robustness and performance in dynamic environments.

Another distinctive feature of CLEAF is the utilisation of both input variables and target output variables in data density calculation for fuzzy rule identification. In practice, rule generation approaches relying on density/potential, distance, firing strength, and statistical contribution criteria aim to identify data samples that are spatially distant from prototypes of existing fuzzy rules. However, these approaches often overlook the boundary regions between different data patterns, where small shifts or changes in input values can lead to significantly different prediction outcomes. Error criterion-based approaches add new rules to the system when the prediction error exceeds a certain threshold, but such approaches are more sensitive towards outliers and lack robustness. In contrast, by considering the mutual distances between input variables as well as target output variables in data density calculation for fuzzy rule identification, CLEAF can accurately identify underlying data patterns and effectively detect overlapping areas within these patterns, thereby attaining greater prediction performance.

To summarise, the key contribution of this paper is CLEAF, a novel EFS that learns from streaming data in a chunk-wise manner while accounting for divergences in both input and output spaces. In particular, the novel features of the proposed CLEAF that are distinctive from alternative single-model first-order EFSs, such as SAFL [26], SEFIS [27], DEFS [28], X-Fuzz [29], include:

1) A chunk-wise prototype identification scheme to better capture the multi-modal distribution of data, mitigate temporary fluctuations in streaming environments, and achieve improved approximation with fewer fuzzy rules.
2) A chunk-wise FWRLS algorithm for consequent parameter updating with significantly higher computational efficiency than its sample-wise counterpart by updating the consequent parameters and associated covariance matrices once per data chunk only.
3) The utilisation of both input variables and target output variables in calculating data density for fuzzy rule

identification to effectively capture the dissimilarity in both input and output spaces, accurately disclosing the underlying data patterns and their boundary regions.

Together, these features enable CLEAF to learn efficiently from streaming data, preserving the strengths of conventional first-order EFSs whilst overcoming the three inherent limitations associated with their structure evolving and parameter learning mechanisms.

Numerical examples based on a range of popular benchmark regression and classification datasets demonstrate that CLEAF achieves high-level prediction accuracy across various tasks surpassing or, at least, on par with the best-performing comparative approaches whilst offering superior computational efficiency, particularly in large-scale non-stationary problems.

The remainder of this paper is organised as follows. Section II presents the preliminaries of this study. Technical details of CLEAF are described in Section III. Numerical examples on various regression and classification problems are given in Section IV for performance demonstration through comparisons with state-of-the-art (SOTA) approaches. This paper is concluded in Section V and directions for future work are also outlined in this section.

II. Problem Foundation

A conventional first-order evolving fuzzy system (EFS) is comprised of N IF-THEN fuzzy rules in the form of Eq. (1) that are constructed from streaming data in an adaptive and evolving manner [2], [26], [30]:

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

where $n = 1, 2, \ldots, N$; $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$ is a $M \times 1$ dimensional input vector to the EFS; $\bar{\boldsymbol{x}} = [1, \boldsymbol{x}^T]$; $\boldsymbol{y}_n = [y_{n,1}, y_{n,2}, \ldots, y_{n,C}]^T$ is the $C \times 1$ dimensional output vector of $\boldsymbol{R}_n$; $\boldsymbol{p}_n = [p_{n,1}, p_{n,2}, \ldots, p_{n,M}]^T$ is the prototype (antecedent parameters) of $\boldsymbol{R}_n$; $\mathbf{A}_n = [\boldsymbol{a}_{n,1}, \boldsymbol{a}_{n,2}, \ldots, \boldsymbol{a}_{n,C}]^T$ is the $C \times (M+1)$ dimensional consequent parameter matrix, and; $\boldsymbol{a}_{n,c} = [a_{n,c,0}, a_{n,c,1}, a_{n,c,2}, \ldots, a_{n,c,M}]^T$.

Compared to the linguistic terms used in conventional fuzzy rules, using prototypes as antecedent parameters significantly simplifies the learning process and eliminates the need for human expertise. Moreover, this simplification does not compromise the interpretability of the fuzzy rules, as these prototypes are actual data samples in the input space and can be described using meaningful linguistic terms.

The input-output relationship of this EFS is formulated as follows.

$$\boldsymbol{y} = f(\boldsymbol{x}) = \sum_{n=1}^{N} \lambda_n \boldsymbol{y}_n = \sum_{n=1}^{N} \lambda_n \mathbf{A}_n \bar{\boldsymbol{x}}^T \tag{2}$$

where λ_n is the normalised firing strength produced by $\boldsymbol{R}_n$ on $\boldsymbol{x}$, calculated by Eq. (3).

$$\lambda_n = \frac{\mu_n}{\sum_{j=1}^{N} \mu_j} \tag{3}$$

Here $\mu_n = e^{-\frac{\|\boldsymbol{x} - \boldsymbol{p}_n\|^2}{\sigma_n^2}}$ is the firing strength of $\boldsymbol{R}_n$; σ_n^2 is a scaling factor. Note that in this paper, the firing strengths are computed using the squared Euclidean distances between data samples and the prototypes of the fuzzy rules. The computational complexity of this distance-based approach increases slower with data dimensionality than the conventional product t-norm-based approaches for computing firing strength, which aggregate attribute-wise differences [40].

Typically, the antecedent parameters of the IF-THEN rules, namely, prototypes are identified from data streams by online clustering, and the consequent parameters are learned accordingly using recursive least squares (RLS)-based algorithms, both in a sample-wise manner [2]. Using the fuzzily weighted RLS (FWRLS) algorithm as an example [30], the consequent parameters of $\boldsymbol{R}_n$ are updated by Eqs. (4) and (5) in response to $\boldsymbol{x}$:

$$\boldsymbol{\Theta}_n \leftarrow \boldsymbol{\Theta}_n - \frac{\lambda_n \boldsymbol{\Theta}_n \bar{\boldsymbol{x}}^T \bar{\boldsymbol{x}} \boldsymbol{\Theta}_n}{1 + \lambda_n \bar{\boldsymbol{x}} \boldsymbol{\Theta}_n \bar{\boldsymbol{x}}^T} \tag{4}$$

$$\mathbf{A}_n \leftarrow \mathbf{A}_n - \lambda_n (\mathbf{A}_n \bar{\boldsymbol{x}}^T - \boldsymbol{t}) \bar{\boldsymbol{x}} \boldsymbol{\Theta}_n \tag{5}$$

where $\boldsymbol{t}$ is the target output corresponding to the input $\boldsymbol{x}$ that is used for guiding the consequent parameter adjustment.

However, as aforementioned, the sample-wise structure evolving and parameter learning scheme used by most first-order EFSs may not adequately capture the complexity and dynamics of the underlying distribution of streaming data. Such learning scheme is more sensitive towards the temporary fluctuations in the underlying data patterns and is less computationally efficient.

III. Proposed CLEAF

In this section, technical details of the chunk-wise learning evolving autonomous fuzzy system (CLEAF) are presented. The proposed CLEAF is a novel evolving fuzzy system (EFS) designed to learn and update both antecedent and consequent parameters of the IF-THEN rules from data streams on a chunk-by-chunk basis. The chunk-wise learning enables CLEAF to acquire richer temporal and spatial information from data streams with higher computational efficiency. In addition, CLEAF innovatively incorporates both the input and target output variables in data density calculation for fuzzy rule generation to more effectively discover the underlying patterns of data and their overlaps. These novel features enable CLEAF to achieve greater approximation of non-linear, non-stationary, complex problems. For clarity, a list of key notations and definitions used in this section is given in Table I.

Remark 1: Chunk-wise learning is related to mini-batch-wise learning, which is widely used by many machine learning algorithms, with artificial neural networks being the best example [3]–[5]. However, the two paradigms are fundamentally different as chunk-wise learning specifically describes an online learning process from streaming data arriving sequentially in chunks [39], [41]. A chunk-wise online learning system must quickly extract key information from each chunk, adapt to evolving data patterns, and discard the raw data afterward. In contrast, mini-batch-wise learning is an offline process in which a static dataset is divided into batches and used repeatedly to train a model until a stopping criterion is met.

Remark 2: Chunk-wise learning is closely related to sample-wise learning, as both paradigms are widely used for

TABLE I
LIST OF KEY NOTATIONS AND DEFINITIONS

Notation	Definition
$\mathbf{X}_l$	the l^{th} data chunk
K_l	the cardinality of $\mathbf{X}_l$
$\boldsymbol{x}_{l,k}$	the k^{th} sample of $\mathbf{X}_l$
$\mathbf{T}_l$	the target outputs of the l^{th} data chunk
$\boldsymbol{t}_{l,k}$	the target output corresponding to $\boldsymbol{x}_{l,k}$
$\mathbf{Y}_l$	the outputs produced in response to $\mathbf{X}_l$
$\boldsymbol{\chi}$	the mean of all input samples
X	the average squared Euclidean norm of all input samples
$\boldsymbol{\tau}$	the mean of all target outputs
T	the average squared Euclidean norm of all target outputs
$D(\boldsymbol{x}_{l,k})$	the data density of $\boldsymbol{x}_{l,k}$
N	the total number of fuzzy rules
$\boldsymbol{R}_j$	the j^{th} fuzzy rule
$\boldsymbol{p}_j$	the prototype of $\boldsymbol{R}_j$
$\boldsymbol{o}_j$	the target output corresponding to $\boldsymbol{p}_j$
$\mathbf{A}_j$	the consequent parameters of $\boldsymbol{R}_j$
$\boldsymbol{\Xi}_j$	the covariance matrix of $\boldsymbol{R}_j$
Λ_j	the average firing strength of $\boldsymbol{R}_j$
$\boldsymbol{z}_{l,k}$	the k^{th} prototype identified from $\mathbf{X}_l$ locally
$\boldsymbol{w}_{l,k}$	the target output corresponding to $\boldsymbol{z}_{l,k}$
$\mathbf{C}_{l,k}$	the cluster formed around $\boldsymbol{z}_{l,k}$
$D_j(\boldsymbol{z}_{l,k})$	the local density of $\boldsymbol{z}_{l,k}$ at the local model of $\boldsymbol{R}_j$
$\mu_{j,k}$	the firing strength of $\boldsymbol{R}_j$ on $\boldsymbol{x}_{l,k}$
$\boldsymbol{\Lambda}_j^*$	the normalised firing strength vector of $\boldsymbol{R}_j$ on $\mathbf{X}_l$
$\lambda_{j,k}^*$	the normalised firing strength of $\boldsymbol{R}_j$ on $\boldsymbol{x}_{l,k}$

online data stream processing [42]. A sample-wise learning system processes individual samples separately and prioritises immediate responses, but it is more sensitive to noise and short-term fluctuations commonly observed in highly non-stationary environments. In contrast, a chunk-wise learning system processes data streams chunk by chunk, where model updates are driven by information aggregated over multiple samples. This aggregation effectively reduces the variance of parameter updates, enabling the system to adapt more reliably to genuine changes in data patterns rather than reacting to noise or transient fluctuations. Thanks to the richer temporal and spatial information contained in each data chunk, which is particularly valuable in highly non-stationary environments, chunk-wise learning often yields higher approximation accuracy without accumulating an excessively large knowledge base or overfitting to temporary fluctuations.

A. System Architecture

The general architecture of the proposed CLEAF system is depicted in Fig. 1, where one can see that CLEAF is comprised of N IF-THEN fuzzy rules in the form of Eq. (1). CLEAF employs the activation control scheme proposed in [26] to identify the more activated fuzzy rules by individual input data samples for rule updating and output generation. The activation control scheme has been proven to be highly effective in reducing the computational complexity of EFSs and foster generalisation.

Given an input data chunk in the form of a $M \times K_l$ dimensional matrix, $\mathbf{X}_l = [\boldsymbol{x}_{l,1}, \boldsymbol{x}_{l,2}, \ldots, \boldsymbol{x}_{l,K_l}]$ (assuming the l^{th} one; K_l is the number of samples within $\mathbf{X}_l$), the outputs

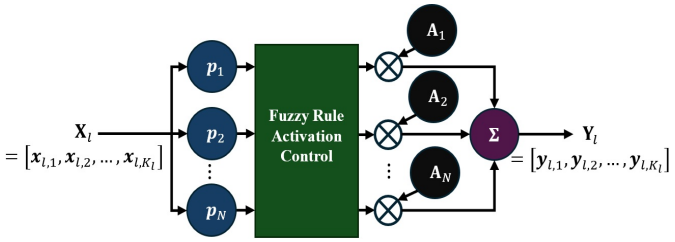


Fig. 1: General architecture of CLEAF system.

generated by CLEAF are computed using Eq. (6):

$$\mathbf{Y}_l = f(\mathbf{X}_l) = \sum_{n=1}^{N} \mathbf{A}_n(\boldsymbol{\Lambda}_n^* \odot \bar{\mathbf{X}}_l) \tag{6}$$

where $\mathbf{Y}_l = [\boldsymbol{y}_{l,1}, \boldsymbol{y}_{l,2}, \ldots, \boldsymbol{y}_{l,K_l}]$ is the $C \times K_l$ dimensional output matrix; $\boldsymbol{y}_{l,k}$ is the output produced by CLEAF given the input $\boldsymbol{x}_{l,k}$; $\bar{\mathbf{X}}_l = [\bar{\boldsymbol{x}}_{l,1}, \bar{\boldsymbol{x}}_{l,2}, \ldots, \bar{\boldsymbol{x}}_{l,K_l}]$; $\bar{\boldsymbol{x}}_{l,k} = [1, \boldsymbol{x}_{l,k}^T]^T$ ($k = 1, 2, \ldots, K_l$); $\boldsymbol{\Lambda}_n^* = [\lambda_{n,1}^*, \lambda_{n,2}^*, ..., \lambda_{n,K_l}^*]$ is the normalised firing strength vector generated by the n^{th} fuzzy rule $\boldsymbol{R}_n$ ($n = 1, 2, \ldots, N$) in response to $\mathbf{X}_l$; $\lambda_{n,k}^*$ is the normalised firing strength of $\boldsymbol{R}_n$ on $\boldsymbol{x}_{l,k}$ (computed by Eq. (27)); "$\odot$" stands for element-wise multiplication; $\mathbf{A}_n$ is the $C \times (M+1)$ dimensional consequent parameter matrix associated with $\boldsymbol{R}_n$. Note that, the cardinalities of different data chunks are not necessarily the same, namely, there exists $K_l \neq K_j$ for $l \neq j$.

B. System Learning

The chunk-wise learning process of CLEAF is composed of five repeating steps, with each data chunk treated as an individual processing unit. All the processed data chunks are discarded to maintain the computation- and memory-efficiency of CLEAF for online learning from data streams.

Step 1. Global meta-parameter initialisation/updating. The global meta-parameters of CLEAF include the mean and average squared Euclidean norm of all input samples presented to CLEAF, denoted by $\boldsymbol{\chi}$ and X, as well as the mean and average squared Euclidean norm of the corresponding target outputs for these input samples, denoted by $\boldsymbol{\tau}$ and T.

Given a new input data chunk $\mathbf{X}_l$ and the corresponding target output matrix $\mathbf{T}_l = [\boldsymbol{t}_{l,1}, \boldsymbol{t}_{l,2}, \ldots, \boldsymbol{t}_{l,K_l}]$ ($\boldsymbol{t}_{l,k}$ is the target output of $\boldsymbol{x}_{l,k}$, $\forall k = 1, 2, \ldots, K_l$), $\boldsymbol{\chi}$ and X are initialised by $\mathbf{X}_l$ using Eq. (7) if it is the first data chunk, namely, $l = 1$:

$$\boldsymbol{\chi} \leftarrow \frac{1}{K_l}\sum_{k=1}^{K_l} \boldsymbol{x}_{l,k}; \quad X \leftarrow \frac{1}{K_l}\sum_{k=1}^{K_l} \|\boldsymbol{x}_{l,k}\|^2 \tag{7}$$

Accordingly, $\boldsymbol{\tau}$ and T are initialised by $\mathbf{T}_l$ in the same way as $\boldsymbol{\chi}$ and X in Eq. (7).

Otherwise, namely, $l > 1$, $\boldsymbol{\chi}$ and X are updated with $\mathbf{X}_l$ using Eqs. (8) and (9):

$$\boldsymbol{\chi} \leftarrow \boldsymbol{\chi} + \frac{\sum_{k=1}^{K_l} \boldsymbol{x}_{l,k} - K_l\boldsymbol{\chi}}{\sum_{j=1}^{l} K_j} \tag{8}$$

$$X \leftarrow X + \frac{\sum_{k=1}^{K_l} \|\boldsymbol{x}_{l,k}\|^2 - K_l X}{\sum_{j=1}^{l} K_j} \tag{9}$$

and, $\boldsymbol{\tau}$ and T are updated with $\mathbf{T}_l$ in the same way as $\boldsymbol{\chi}$ and X in Eqs. (8) and (9).

After the global meta-parameters are updated/initialised with the current data chunk $\mathbf{X}_l$ and the corresponding target outputs $\mathbf{T}_l$, the current learning cycle enters the second step.

Step 2. Prototype identification. In this step, data samples representing local peaks of multi-modal distribution are identified from the input data chunk $\mathbf{X}_l$ as prototypes. To do so, data density values of individual data samples within the current input data chunk $\mathbf{X}_l$ are firstly calculated using Eq. (10) ($k = 1, 2, \ldots, K_l$).

$$D(\boldsymbol{x}_{l,k}) = \frac{\sum_{j=1}^{K_l} \alpha_{k,j} \cdot e^{-(\rho_o \frac{\|\boldsymbol{x}_{l,k} - \boldsymbol{x}_{l,j}\|^2}{\sigma_x^2} + (1-\rho_o)\frac{\|\boldsymbol{t}_{l,k} - \boldsymbol{t}_{l,j}\|^2}{\sigma_t^2})}}{\sum_{j=1}^{K_l} \alpha_{k,j}} \tag{10}$$

where $0 \leq \rho_o \leq 1$; $\sigma_x^2 = X - \|\boldsymbol{\chi}\|^2$; $\sigma_t^2 = T - \|\boldsymbol{\tau}\|^2$, and; $\alpha_{k,j}$ is obtained by Eq. (11).

$$\alpha_{k,j} = \begin{cases} 1, & if\ \|\boldsymbol{x}_{l,k} - \boldsymbol{x}_{l,j}\|^2 \leq \sigma_x^2\ \&\ \|\boldsymbol{t}_{l,k} - \boldsymbol{t}_{l,j}\|^2 \leq \sigma_t^2 \\ 0, & else \end{cases} \tag{11}$$

The coefficient ρ_o adjusts the respective weights assigned to the distances calculated based on the input variables and output variables when measuring dissimilarity between two data samples. By default, $\rho_o = 0.5$ is used, giving the balanced importance to the dissimilarity between data samples in both the input and output spaces.

Remark 3: The data density $D(\boldsymbol{x}_{l,k})$, computed using Eq. (10), is defined in a similar way as [39] but based on a weighted combination of the distances between $\boldsymbol{x}_{l,k}$ and other input data samples $\boldsymbol{x}_{l,j} \in \mathbf{X}_l$, and the distances between their corresponding target outputs $\boldsymbol{t}_{l,k}$ and $\boldsymbol{t}_{l,j} \in \mathbf{T}_l$, capturing dissimilarity in both input and output spaces. In general, an input sample will have a higher density value if it is located at a densely distributed region closer either to one of the local peaks of the multi-modal data distributions or to the boundaries of different data patterns.

Next, a subset of $\mathbf{X}_l$ with the highest density values locally are selected using Cond. 1. These input samples with the local maximum density values serve as the prototypes representing the local models of data distribution extracted from $\mathbf{X}_l$.

$$\begin{aligned} \text{Cond. } 1: \ &\text{if } \left(D(\boldsymbol{x}_{l,k}) = \max_{\forall j, \alpha_{k,j}=1} (D(\boldsymbol{x}_{l,j}))\right) \\ &\text{then } (\boldsymbol{x}_{l,j} \text{ is one of the local maxima}) \end{aligned} \tag{12}$$

The collection of prototypes is denoted by the $M \times N_l$ dimensional matrix $\mathbf{Z}_l = [\boldsymbol{z}_{l,1}, \boldsymbol{z}_{l,2}, \ldots, \boldsymbol{z}_{l,N_l}]$ ($\mathbf{Z}_l \subset \mathbf{X}_l$) with N_l being the number of prototypes identified by Cond. 1 from $\mathbf{X}_l$. The target outputs corresponding to $\mathbf{Z}_l$ are denoted as: $\mathbf{W}_l = [\boldsymbol{w}_{l,1}, \boldsymbol{w}_{l,2}, \ldots, \boldsymbol{w}_{l,N_l}]$ ($\mathbf{W}_l \subset \mathbf{T}_l$), where $\boldsymbol{w}_{l,j}$ is the target output of $\boldsymbol{z}_{l,j}$.

Then, individual data samples within $\mathbf{X}_l$ are assigned to the nearest prototype using Eq. (13) to form clusters resembling Voronoi tessellations.

$$\mathbf{C}_{l,n^*} \leftarrow \mathbf{C}_{l,n^*} \cup \{\boldsymbol{x}_{l,k}\}; \tag{13}$$

where $\mathbf{C}_{l,j}$ denotes the cluster formed around the prototype $\boldsymbol{z}_{l,j}$ ($j = 1, 2, \ldots, N_l$); $\boldsymbol{x}_{l,k} \in \mathbf{X}_l$; $n^* = \arg\min_{j=1,2,\ldots,N_l} (\rho_o \frac{\|\boldsymbol{z}_{l,j} - \boldsymbol{x}_{l,k}\|^2}{\sigma_x^2} + (1-\rho_o)\frac{\|\boldsymbol{w}_{l,j} - \boldsymbol{t}_{l,k}\|^2}{\sigma_t^2})$.

After the clusters are formed, the parameters associated with the prototypes $\boldsymbol{z}_{l,j}$ ($j = 1, 2, ..., N_l$), such as the mean and average squared Euclidean norm of the input samples associated with $\mathbf{C}_{l,j}$, denoted by $\boldsymbol{q}_{l,j}$ and $Q_{l,j}$ are obtained using Eqs. (14) and (15), and the mean and average squared Euclidean norm of the corresponding target outputs, denoted by $\boldsymbol{r}_{l,j}$ and $R_{l,j}$ are obtained similarly.

$$\boldsymbol{q}_{l,j} = \frac{1}{Z_{l,j}} \sum_{\boldsymbol{x} \in \mathbf{C}_{l,j}} \boldsymbol{x} \tag{14}$$

$$Q_{l,j} = \frac{1}{Z_{l,j}} \sum_{\boldsymbol{x} \in \mathbf{C}_{l,j}} \|\boldsymbol{x}\|^2 \tag{15}$$

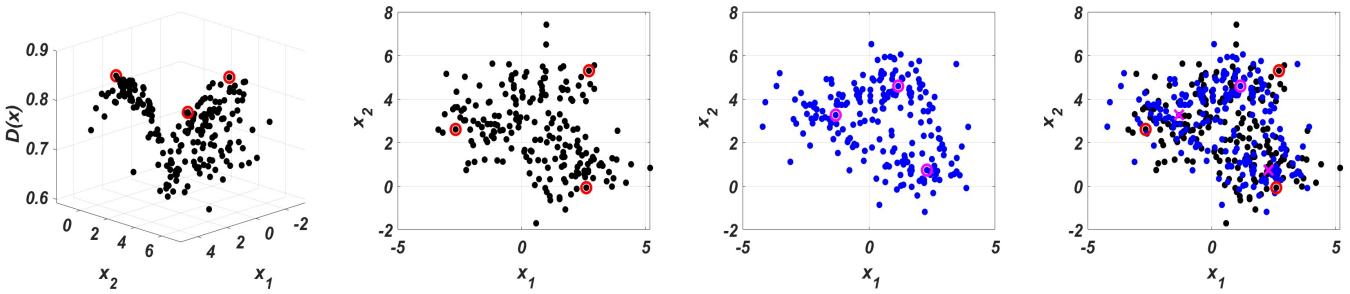
where $j = 1, 2, \ldots, N_l$; $Z_{l,j}$ is the support of $\mathbf{C}_{l,j}$ (the number of data samples associated with $\mathbf{C}_{l,j}$).

Remark 4: Unlike conventional EFSs that process individual samples in isolation, CLEAF calculates data density across all samples within each data chunk, allowing it to better capture the multi-modal nature of data distributions. This chunk-wise approach enables CLEAF to leverage richer temporal and spatial information persevered in individual data chunks to identify the underlying patterns in data streams, and represent them through prototypes in a flexible, objective manner, without relying on prior assumptions about data generation.

Remark 5: Conventional EFSs generally focus solely on the input space, overlooking possible divergences in the output space [24]. In contrast, CLEAF incorporates mutual distances between data samples in both the input and output spaces when calculating data density. This allows CLEAF to more effectively recognise and differentiate between various local patterns and overlapping regions, enhancing its ability to capture complex data relationships.

In this research, a two-dimensional synthetic dataset comprising three Gaussian clusters with 20,000 samples in total is generated for illustrative purposes. The first 400 samples are used to visualise the chunk-wise prototype identification and fusion process. CLEAF processes the 400 samples in two consecutive chunks, each containing 200 samples (e.g., $K_1 = K_2 = 200$). The density values of the samples in the first chunk are plotted in Fig. 2a, where blue dots represent individual data samples and red circles denote the identified prototypes, corresponding to samples with local maximum density values. A scatter plot is also provided in Fig. 2b to better visualise the spatial distribution of data samples and prototypes. Together, these prototypes sculpt the antecedent space by defining regions of influence around them.

Step 3. Structure evolving and antecedent parameter updating. In this step, the fuzzy rule base is initialised if $\mathbf{X}_l$ is the very first data chunk, namely, $l = 1$. In this case, there are a total of N ($N \leftarrow N_l$) new fuzzy rules created with the extracted prototypes $\mathbf{Z}_l$ in the form of Eq. (1). The



(a) Data density of the first data chunk (b) Prototypes from the first data chunk (c) Prototypes from the second data chunk (d) Prototypes after fusion and removal of redundant ones

Fig. 2: Illustration of prototype identification and fusion process across consecutive data chunks.

antecedent and consequent parameters of the new fuzzy rules are initialised by Eq. (16) ($j = 1, 2, \ldots, N$).

$$\boldsymbol{p}_j \leftarrow \boldsymbol{z}_{l,j}; \quad \mathbf{A}_j \leftarrow \mathbf{0}_{C\times(M+1)}; \quad \boldsymbol{\Xi}_j \leftarrow \Omega_o \mathbf{I}_{(M+1)\times(M+1)} \tag{16}$$

where $\mathbf{0}_{C\times(M+1)}$ is a $C \times (M+1)$ dimensional zero matrix; $\boldsymbol{\Xi}_j$ is the covariance matrix associated with the j^{th} fuzzy rule, $\boldsymbol{R}_j$; $\mathbf{I}_{(M+1)\times(M+1)}$ is a $(M+1) \times (M+1)$ dimensional identity matrix; Ω_o is a small constant for covariance matrix initialisation, standard for RLS. In this study, $\Omega_o = \frac{1}{10000}$.

The target output corresponding to $\boldsymbol{p}_j$ is set as $\boldsymbol{o}_j \leftarrow \boldsymbol{w}_{l,j}$, and other meta-parameters associated with the fuzzy rules are initialised as ($j = 1, 2, \ldots, N$):

$$\boldsymbol{\rho}_j \leftarrow \boldsymbol{q}_{l,j}; \quad P_j \leftarrow Q_{l,j}; \quad \boldsymbol{\gamma}_j \leftarrow \boldsymbol{r}_{l,j}; \quad \Gamma_j \leftarrow R_{l,j} \tag{17}$$

where $\boldsymbol{\rho}_j$ and $\boldsymbol{\gamma}_j$ are the means of input samples and corresponding target outputs associated with $\boldsymbol{R}_j$; P_j and Γ_j are the respective average squared Euclidean norms. In addition, the support of $\boldsymbol{R}_j$ is set as: $S_j \leftarrow Z_{l,j}$ and the time instance at which $\boldsymbol{R}_j$ is created is recorded as $I_j \leftarrow l$.

In addition, the average firing strength of each individual fuzzy rule, $\boldsymbol{R}_j$ by the current chunk $\mathbf{X}_l$ is computed as [26]:

$$\Lambda_j = \frac{1}{K_l} \sum_{k=1}^{K_l} \mu_{j,k} \tag{18}$$

where $\mu_{j,k}$ is the firing strength produced by $\boldsymbol{R}_j$ in response to $\boldsymbol{x}_{l,k}$, calculated by Eq. (19):

$$\mu_{j,k} = e^{-\frac{\|\boldsymbol{x}_{l,k} - \boldsymbol{p}_j\|^2}{\sigma_{j,x}^2}} \tag{19}$$

and there is $\sigma_{j,x}^2 = \frac{1}{2}(P_j - \|\boldsymbol{\rho}_j\|^2 + \sigma_x^2)$.

Otherwise, namely, $l > 1$, the local density values of every prototype, $\boldsymbol{z}_{l,k}$ ($\boldsymbol{z}_{l,k} \in \mathbf{Z}_l$) at the local models represented by the existing fuzzy rules identified from historical data chunks are firstly calculated using Eq. (20).

$$D_j(\boldsymbol{z}_{l,k}) = e^{-(\rho_o \frac{\|\boldsymbol{p}_j - \boldsymbol{z}_{l,k}\|^2}{\sigma_{j,x}^2} + (1-\rho_o)\frac{\|\boldsymbol{o}_j - \boldsymbol{w}_{l,k}\|^2}{\sigma_{j,t}^2})} \tag{20}$$

where $j = 1, 2, \ldots, N$; $k = 1, 2, \ldots, N_l$; $\sigma_{j,t}^2 = \frac{1}{2}(\Gamma_j - \|\boldsymbol{\gamma}_j\|^2 + \sigma_t^2)$.

The local density serves as an indicator showing how well the prototype fits the local model. The higher $D_j(\boldsymbol{z}_{l,k})$ is, the more similar $\boldsymbol{z}_{l,k}$ is to the prototype of the j^{th} local model represented by $\boldsymbol{p}_j$, and vice versa.

Then, Cond. 2 is utilised to identify a selected group of prototypes from $\mathbf{Z}_l$ that represent unseen data patterns from historical data chunks ($k = 1, 2, \ldots, N_l$).

$$\begin{aligned} \text{Cond. 2}: \ &\text{if } \big(\min_{j=1,2,\ldots,N}(D_j(\boldsymbol{z}_{l,k})) < D_o\big) \\ &\text{then } (\boldsymbol{z}_{l,k} \text{ represents an unseen pattern}) \end{aligned} \tag{21}$$

where D_o is a threshold externally controlled by users. In this study, D_o is set as $e^{-\frac{1}{6}}$.

If Cond. 2 is not satisfied by $\boldsymbol{z}_{l,k}$ ($\forall k$), $\boldsymbol{z}_{l,k}$ is used for updating the parameters of the fuzzy rule with the nearest prototype, denoted by $\boldsymbol{R}_{n^*}$, where $n^* = \arg\max_{j=1,2,\ldots,N}(D_j(\boldsymbol{z}_{l,k}))$. The mean, $\boldsymbol{\rho}_{n^*}$ and average squared Euclidean norm, P_{n^*} of input samples associated with $\boldsymbol{R}_{n^*}$ are updated using Eqs. (22) and (23). Similarly, the mean, $\boldsymbol{\gamma}_{n^*}$ and average squared Euclidean norm, Γ_{n^*} of the target outputs corresponding to these input samples are updated in the same manner as $\boldsymbol{\rho}_{n^*}$ and P_{n^*} in Eqs. (22) and (23). The support of $\boldsymbol{R}_{n^*}$ is also updated as $S_{n^*} \leftarrow S_{n^*} + Z_{l,k}$.

$$\boldsymbol{\rho}_{n^*} \leftarrow \boldsymbol{\rho}_{n^*} + \frac{Z_{l,k}(\boldsymbol{q}_{l,k} - \boldsymbol{\rho}_{n^*})}{S_{n^*} + Z_{l,k}} \tag{22}$$

$$P_{n^*} \leftarrow P_{n^*} + \frac{Z_{l,k}(Q_{l,k} - P_{n^*})}{S_{n^*} + Z_{l,k}} \tag{23}$$

If Cond.2 is met, a new fuzzy rule $\boldsymbol{R}_N$ ($N \leftarrow N+1$) is initialised with $\boldsymbol{z}_{l,k}$ being its prototype, namely, $\boldsymbol{p}_N \leftarrow \boldsymbol{z}_{l,k}$ (and $\boldsymbol{w}_{l,j}$ being the corresponding target output, $\boldsymbol{o}_N \leftarrow \boldsymbol{w}_{l,k}$). Consequent parameters and other associated parameters of $\boldsymbol{R}_N$ are initialised using Eqs. (16) and (17). The average firing strength, Λ_N of $\boldsymbol{R}_N$ is then calculated using Eq. (18).

Finally, the average firing strengths of all fuzzy rules are updated using Eq. (24) ($n = 1, 2, \ldots, N$):

$$\Lambda_j \leftarrow \Lambda_j + \frac{\sum_{k=1}^{K_l} \mu_{j,k} - \Lambda_j}{\sum_{j=I_j}^{l} K_j} \tag{24}$$

Following the visual example given Fig. 2b, the scatter plot of data samples in the second data chunk and the identified prototypes from this data chunk is depicted in Fig. 2c, where green dots represent individual data samples in the second chunk; magenta circles denote the identified prototypes. After

fusion with the prototypes from the first data chunk, the remaining prototypes are shown in Fig. 2d, where magenta crosses indicate prototypes that are highly similar to those from the first data chunk and have therefore been removed; magenta circles are the prototypes preserved after fusion. The removal of highly similar prototypes effectively reduces the overlap between the regions of influence of remaining prototypes, resulting in a more compact fuzzy rule base.

Step 4. Rule quality monitoring. In this step, the fuzzy rules that contribute less to the outputs of CLEAF are identified using Cond. 3 ($j = 1, 2, \ldots, N$).

$$\begin{aligned} \text{Cond. 3}: \ & \text{if } \left(\Lambda_n < \Lambda_o\right) \\ & \text{then } (\boldsymbol{R}_j \text{ is a less activated fuzzy rule}) \end{aligned} \tag{25}$$

where Λ_o is a sensitivity threshold determined by users. A greater value of Λ_o enables CLEAF to be more sensitive to less activated fuzzy rules, and vice versa. The recommended setting for Λ_o is 0.01.

The fuzzy rules that satisfy Cond. 3 are removed from the fuzzy rule base, and then the current learning cycle proceeds to the final step.

Step 5. Consequent parameter updating. Finally, the consequent parts of the remaining fuzzy rules within the rule base are updated in this step using the modified FWRLS algorithm for chunk-wise consequent parameter updating. To do so, for every input sample $\boldsymbol{x}_{l,k} \in \mathbf{X}_l$, the firing strengths produced by fuzzy rules in response to $\boldsymbol{x}_{l,k}$ are firstly computed by Eq. (19), and ranked in descending order denoted as $\hat{\mu}_{1,k} \geq \hat{\mu}_{2,k} \geq \cdots \geq \hat{\mu}_{N,k}$. The number of fuzzy rules activated by $\boldsymbol{x}_{l,k}$ are estimated using Eq. (26) [26].

$$\hat{n}_k = \min_{n=1,2,\ldots,N} \left(\sum_{j=1}^{n} \hat{\mu}_{j,k} > \kappa_o \sum_{j=1}^{N} \mu_{j,k}\right) \tag{26}$$

where κ_o ($0 < \kappa_o \leq 1$) is a parameter determined by users to control the number of fuzzy rules within the rule base being activated by $\boldsymbol{x}_{l,k}$. The closer κ_o is to 1, the more rules will be considered as being activated by the given sample, $\boldsymbol{x}_{l,k}$. By default, $\kappa_o = 0.5$, but one may adjust the value of κ_o according to the nature of the given problem.

Then, the normalised firing strength of the fuzzy rules in response to $\boldsymbol{x}_{l,k}$ are computed using Eq. (27).

$$\lambda^*_{j,k} = \begin{cases} \frac{\mu_{j,k}}{\sum_{n=1}^{\hat{n}_k} \hat{\mu}_{n,k}}, & if \ \mu_{j,k} \geq \hat{\mu}_{\hat{n}_k,k} \\ 0, & else \end{cases} \tag{27}$$

Next, for each fuzzy rule of CLEAF, the associated covariance matrix is updated with $\mathbf{X}_l$ by using Eq. (28) [26], [43].

$$\boldsymbol{\Xi}_j \leftarrow \boldsymbol{\Xi}_j + (\boldsymbol{\Lambda}^*_j \odot \bar{\mathbf{X}}_l)\bar{\mathbf{X}}_l^T \tag{28}$$

where $j = 1, 2, \ldots, N$; $\bar{\boldsymbol{x}}_{l,k} = [1, \boldsymbol{x}^T_{l,k}]$, same as Eq. (6).

After $\boldsymbol{\Xi}_j$ is updated using Eq. (28), the consequent parameter matrix associated with $\boldsymbol{R}_j$ is updated using Eq. (29) ($j = 1, 2, \ldots, N$).

$$\mathbf{A}_j \leftarrow \mathbf{A}_j - (\mathbf{A}_j\bar{\mathbf{X}}_l - \mathbf{T}_l)(\boldsymbol{\Lambda}^*_n \odot \bar{\mathbf{X}}_l)^T \boldsymbol{\Xi}_j^{-1} \tag{29}$$

Remark 6: In contrast to the original fuzzily weighted recursive least squares (FWRLS) algorithm, which updates the consequent part of IF-THEN rules on a sample-by-sample basis (as given by Eqs. (4) and (5)), the proposed chunk-wise FWRLS algorithm only updates the consequent parameters and associated covariance matrix once per data chunk. This modification significantly improves the computational efficiency of CLEAF, as verified by the computational complexity analysis presented in Section III.C theoretically and the ablation analysis presented in Section IV.D empirically.

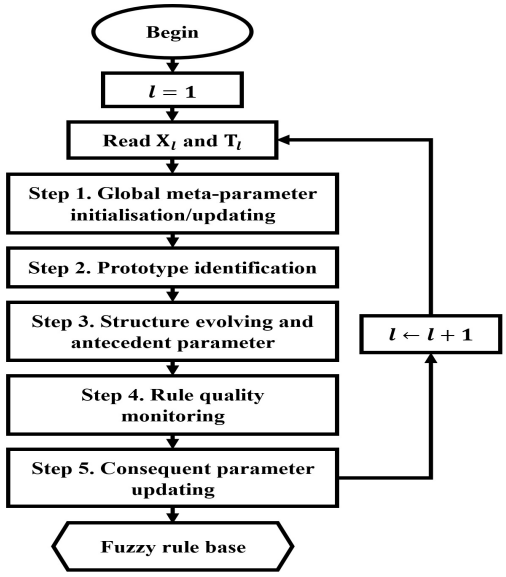


Fig. 3: Diagram of CLEAF chunk-wise learning process.

After the consequent parameter matrices of all the fuzzy rules have been updated using Eq. (29), the current learning cycle finishes. CLEAF starts a new learning cycle (going back to Step 1) with the next available data chunk $\mathbf{X}_l$ ($l \leftarrow l + 1$). The main procedure of the chunk-wise system learning process of CLEAF is summarised by Algorithm 1 in the form of pseudo code.

C. Computational Complexity Analysis

A brief analysis on the computational complexity of the system learning and decision-making processes of CLEAF is presented in this subsection. The detailed analysis is provided in Supplementary Section A.

1) System learning: Given a particular data chunk $\mathbf{X}_l$ and the corresponding target outputs $\mathbf{T}_l$, the computational complexity of CLEAF to self-evolve its system structure and self-update the parameters from $\mathbf{X}_l$ and $\mathbf{T}_l$ is $O(N(M+1)^3 + K_l N(M+1)^2 + (NN_l + K_l^2)(M+C))$. The overall computational complexity of the CLEAF system identification process given L data chunks is: $O(LN(M+1)^3 + KN(M+1)^2 + \sum_{l=1}^{L}(NN_l + K_l^2)(M+C))$.

In contrast, if a conventional RLS-based algorithm, e.g., FWRLS is used, the overall computational complexity becomes $O(KN(M+1)^3 + \sum_{l=1}^{L}(NN_l + K_l^2)(M+C))$, which would be significantly larger than the proposed chunk-wise approach, particularly when the chunk size is large.

A numerical example is conducted on the synthetic dataset introduced earlier to empirically illustrate the influence of different chunk sizes on the computational efficiency of CLEAF.

Algorithm 1 CLEAF chunk-wise learning process

$l \leftarrow 1$
while ($\mathbf{X}_l$ and $\mathbf{T}_l$ are available) **do**
  if ($l = 1$) **then**
  # *Global meta-parameter initialisation (Step 1)* #
    initialise $\boldsymbol{\chi}$, X, $\boldsymbol{\tau}$ and T with $\mathbf{X}_l$ and $\mathbf{T}_l$
  else
  # *Global meta-parameter updating (Step 1)* #
    update $\boldsymbol{\chi}$, X, $\boldsymbol{\tau}$ and T with $\mathbf{X}_l$ and $\mathbf{T}_l$
  # *Prototype identification (Step 2)* #
  calculate $D(\boldsymbol{x}_{l,k})$ ($\forall k = 1, 2, ...K_l$) by (10)
  identify $\mathbf{Z}_l$ and $\mathbf{W}_l$ using Cond. 1
  form clusters around $\mathbf{Z}_l$ by (13)
  for $j = 1$ to N_l **do**
    obtain $Z_{l,j}$, $\boldsymbol{q}_{l,j}$, $Q_{l,j}$, $\boldsymbol{r}_{l,j}$ and $R_{l,j}$ from $\mathbf{C}_{l,j}$
  if ($l = 1$) **then**
  # *Fuzzy rule base initialisation (Step 3)* #
    $N \leftarrow N_l$
    for $j = 1$ to N **do**
      initialise $\boldsymbol{R}_j$ by (16)
      initialise $\boldsymbol{\rho}_j$, P_j, $\boldsymbol{\gamma}_j$ and Γ_j by (17)
      initialise S_j and I_j
      calculate Λ_j by (18)
  else
    for $k = 1$ to N_l **do**
      calculate $D_j(\boldsymbol{z}_{l,k})$ ($\forall j = 1, 2, ...N$) by (20)
      if (Cond. 2 is satisfied) **then**
  # *Structure evolving (Step 3)* #
        $N \leftarrow N + 1$
        initialise $\boldsymbol{R}_N$ by (16)
        initialise $\boldsymbol{\rho}_N$, P_N, $\boldsymbol{\gamma}_n$ and Γ_n by (17)
        initialise S_N and I_N
        calculate Λ_N by (18)
      else
  # *Antecedent parameter updating (Step 3)* #
        $n^* = \arg\max_{j=1,2,...,N}(D_j(\boldsymbol{z}_{l,k}))$
        update $\boldsymbol{\rho}_{n^*}$, P_{n^*}, $\boldsymbol{\gamma}_{n^*}$, Γ_{n^*} and S_{n^*}
    for $j = 1$ to N **do**
      update Λ_j by (24)
  # *Rule quality monitoring (Step 4)* #
  for $j = 1$ to N **do**
    if (Cond. 3 is satisfied) **then**
      remove $\boldsymbol{R}_j$
  # *Consequent parameter updating (Step 5)* #
  for $k = 1$ to N_l **do**
    calculate $\mu_{j,k}$ ($\forall j = 1, 2, ...N$) by (19)
    obtain $\hat{n}_k$ by (26)
    calculate $\lambda^*_{j,k}$ ($\forall j = 1, 2, ...N$) by (27)
  update $\boldsymbol{\Xi}_j$ by (28)
  update $\mathbf{A}_j$ by (29)

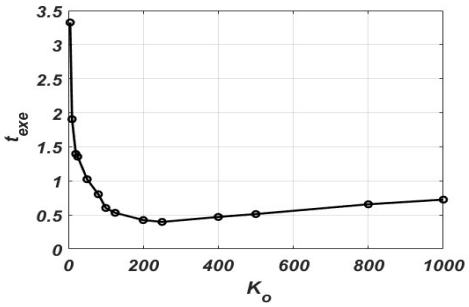


Fig. 4: Influence of chunk size on the computational efficiency of CLEAF.

During the experiment, the dataset is processed chunk-by-chunk, assuming a uniform chunk size, K_o (e.g., $K_l = K_o, \forall l$), which varies from 5 to 1000 with a nonlinear interval. The training time consumption (t_{exe}, in seconds) of CLEAF using different chunk sizes is plotted in Fig. 4. As shown in Fig. 4, the computational efficiency of CLEAF increases significantly with K_o increased from 5 to 250, followed by a slight decrease for larger K_o. This decrease in efficiency is due to the higher computational cost of prototype identification in Step 2.

2) Decision-making: Given a new unlabelled data chunk, $\mathbf{X}_l$, the computational complexity for CLEAF to generate the outputs in response to $\mathbf{X}_l$ is $O(NK_lC(M+1))$, which is same as conventional EFSs.

D. Stability Analysis

The stability of CLEAF is established in Theorem 1. The proof of this theorem is provided in Supplementary Section B due to space limitations.

Theorem 1. Under the standard persistent-excitation (PE) assumption, CLEAF is stable in the sense that both its consequent parameter matrices and outputs remain bounded for any bounded input data chunks.

Remark 7: Theorem 1 establishes that the consequent parameters and outputs of CLEAF remain bounded for any bounded input chunk, ensuring robust behaviour during online learning. A stronger result, namely, convergence of the consequent parameters would typically require a more detailed analysis, such as the construction of an explicit Lyapunov function [26]. A full convergence analysis is beyond the scope of this work and is left for future research.

IV. Experimental Investigation

In this section, numerical experiments are carried to demonstrate the performance of the proposed the chunk-wise learning evolving autonomous fuzzy system (CLEAF). The algorithms were developed using MATLAB2023b platform. The performance evaluation was carried out on a laptop with i7-12700H processor, 64GB RAM and RTX 3050 Ti GPU. Unless specifically declared otherwise, all the reported results are obtained as the average of 10 Monte Carlo experiments by default to allow a certain degree of randomness. It is important to note that all the reported numerical results of

CLEAF are obtained within simulated streaming environments, where data is partitioned into non-overlapping chunks and processed one-by-one sequentially. The source code of CLEAF is avaiable at: https://github.com/Gu-X/Chunk-Wise-Learning-Evolving-Autonomous-Fuzzy-System.

A. Configuration

1) Data description: In this paper, a total of 22 public benchmark datasets have been used for experimental investigation, which include 10 standard classification problems, eight large-scale non-stationary classification problems, one California housing problem, one S&P500 closing price prediction problem, and two visual classification problems. Key information of the 22 benchmark problems is provided in Supplementary Section C.

2) Parameter setting for CLEAF: The proposed CLEAF has four externally controlled parameters, namely, ρ_o, D_o, κ_o and Λ_o. As aforementioned, ρ_o is a coefficient for adjusting the weights assigned to the input samples and target outputs in the dissimilarity calculation; D_o is used by Cond. 2 for identifying novel data patterns; κ_o is used by Eq. (26) to control the number of fuzzy rules within the rule base being activated given a particular input sample, and; Λ_o is the threshold used by Cond. 3 to filter out these fuzzy rules that contribute less to the system outputs. Unless specifically declared otherwise, the values of the four externally controlled parameters are set as: $\rho_o = 0.5$; $D_o = e^{-\frac{1}{6}}$; $\kappa_o = 0.5$, and; $\Lambda_o = 0.01$ by default for all the numerical examples presented in this section.

For simplification, the sizes of data chunks are assumed to be uniformly the same, namely, $K_l = K_o, \forall l$. $K_o = 200$ is used across all the numerical examples. During the experiments, if the amount of remaining unprocessed data samples is less than K_o, all of the remaining samples will be included in the last chunk. Note that, the sizes of different data chunks do not necessarily need to be the same.

3) Comparative algorithms: In this paper, the following seven state-of-the-art (SOTA) offline classification algorithms are employed for performance comparison: 1) support vector machine (SVM) [6]; 2) k-nearest neighbour classifier (KNN) [44]; 3) random forest (RF) [45]; 4) eigenClass (EC) classifier [46]; 5) sequence (SEQ) classifier [47]; 6) sequence-dictionary-based KNN (SDKNN) classifier [47], and; 7) stage-wise additive modelling using a multi-class exponential loss function (SAMME) [48]. In addition, 12 popular single-model and ensemble evolving fuzzy system (EFS) models are used for comparison, which include: 8) sequential adaptive fuzzy inference system (SAFIS) [14]; 9) extended sequential adaptive fuzzy inference system (ESAFIS) [16]; 10) evolving fuzzy ensemble (eEnsemble) [34]; 11) parsimonious learning machine (PALM) [23]; 12) autonomous learning multimodel system (ALMMo) [30]; 13) self-adaptive fuzzy learning system (SAFL) [26]; 14) self-organizing fuzzy inference ensemble system (SOFE) [41]; 15) leaky-ReLU-based evolving classifier (LREC) [49]; 16) statistically evolving fuzzy inference system (SEFIS) [27]; 17) fuzzily weighted adaptive boosting system (FWADB) [50]; 18) self-adaptive fuzzy learning ensemble system (SAFLE) [33], and; 19) multilayer stacked evolving fuzzy system (MS-EFS) [11]. Parameter settings of the comparative algorithms are detailed in Supplementary Section D.

B. Sensitivity analysis

A sensitivity analysis based on three real-life large-scale datasets, namely, electricity pricing (EP), skin segmentation (SS) and weather forecasting (WF) is carried out to investigate the influences of the four externally controlled parameters (ρ_o, D_o, κ_o, Λ_o) and chunk size, K_o on the prediction performances of CLEAF. During the experiments, 80% of data samples are randomly selected and used for training the prediction model and the remaining 20% are used for performance evaluation. The detailed results in terms of classification error rate (err), number of identified fuzzy rules (N) and training time consumption (t_{exe}) are presented in Supplementary Section E.

Supplementary Table S5 indicates that choosing a very small ρ_o (e.g., 0.1) would give excessive importance to the divergence in output space. This will lead to lower sensitivity to the dissimilarity in the input space, leading to fewer fuzzy rules, lower memory consumption and prediction accuracy. Conversely, if ρ_o is too large (e.g., 0.9), CLEAF becomes less capable of distinguishing overlapping local input patterns associated with different outputs, yielding lower prediction accuracy.

Supplementary Table S6 shows that a larger D_o (e.g., $e^{-\frac{1}{10}}$) increases the sensitivity of CLEAF towards new data patterns, resulting in a larger fuzzy rule base, higher memory consumption and decreased computational efficiency. In contrast, a smaller D_o (e.g., $e^{-\frac{1}{2}}$) decreases the computational and memory costs of CLEAF thanks to a smaller rule base identified from the streaming data, but may also limit its adaptability to the changing data patterns.

Supplementary Table S7 suggests that a larger κ_o (e.g., 0.7) activates more fuzzy rules in consequent parameter updating and output generation, increasing computational cost and risking overfitting. In contrast, a very small κ_o (e.g., 0.3) may cause a significant loss of information and decrease prediction accuracy as too few fuzzy rules are activated in response to each input sample.

Supplementary Table S8 shows that a larger Λ_o (e.g., 0.1) encourages CLEAF to drop more fuzzy rules that are less activated over time and improves its responsiveness to the immediate changes in data patterns, but may deteriorate the prediction accuracy if too few rules remaining. A very small Λ_o (e.g., 0.001), however, would decrease the adaptability and computational efficiency of CLEAF because outdated rules are not removed timely.

Finally, Supplementary Table S9 indicates that a larger K_o makes CLEAF less sensitive towards the temporary fluctuations in the underlying patterns of data streams, resulting in a smaller rule base with higher computational efficiency and lower memory cost. However, if K_o is too large (e.g., 800), CLEAF may fail to adequately capture the short-term dynamic behaviours of the streaming data, leading to decreased adaptability and yielding less accurate predictions.

Based on these observations, practical guidelines for tuning CLEAF in real applications can be drawn. In highly non-

stationary environments with rapidly changing data patterns, one may choose a large D_o, a large Λ_o and a small K_o to increase the adaptability of CLEAF with improved prediction accuracy, but this may yield a larger rule base with higher computational and memory costs. When the data stream exhibits abundant temporary fluctuations, a large K_o enables CLEAF to suppress less informative temporary fluctuations and evolve reliably in response to genuine pattern changes, as noted in Remark 2. It should be emphasised that the parameter setting used in this paper, as well as the above guidelines, are provided for user reference only. All externally controlled parameters may be selected according to user preference and do not require prior knowledge of the problem; however, the optimal settings will inevitably vary across problems.

C. Performance Demonstration

In this subsection, numerical experiments are carried out using the aforementioned benchmark datasets for performance demonstration.

1) Examples on standard classification problems: In the first numerical example, the prediction performance of the proposed CLEAF is tested on the 10 standard classification benchmark datasets: abalone (AB), occupancy detection (OD), pen-based recognition of handwritten digits (PR), optical recognition of handwritten digits (OR), MAGIC gamma telescope (MG), phishing websites (PW), page-blocks (PB), spambase (SP), texture (TE), and mammography (MA). In this example, for OD, PR and OR datasets, the original train/test splits are kept and the training samples are randomly shuffled in each experiment. For the other seven datasets, half of the data samples are randomly selected out for model training, and the remaining ones are used as testing samples. The average error rate (err) of CLEAF on the testing sets of the 10 datasets is tabulated in Table II. The average classification error rates of 17 comparative algorithms, which include SVM, KNN, RF, EC, SEQ, SDKNN, SAMME, ESAFIS, eEnsemble, PALM, ALMMo, SAFL, SOFE, LREC, FWADB, SAFLE and MS-EFS, on the 10 datasets obtained under the same experimental protocol are reported in Table II for benchmark comparison. The average training time costs (t_{exe}, in seconds) of the 18 algorithms are presented in Table II as well. The detailed results are given in Supplementary Tables S10 and S11.

It can be observed from Table II that CLEAF outperforms all 17 single-model and multi-model competitors by attaining the lowest average classification error rate over the 10 benchmark datasets with different nature. It further shows that the computational efficiency of the proposed CLEAF is higher than the alternative first-order EFSs involved in the experiments, e.g., ESAFIS, PALM, ALMMo, SAFL, LREC, SAFLE and MS-EFS.

To examine the statistical significance, the Friedman test [51] is carried out using the classification error rates of the 18 algorithms during each experiment, where $p < 0.0001$ is returned from the test yielding a highly significant result. Next, pairwise Wilcoxon signed rank tests [52] are carried out to compare CLEAF against each of the 17 competitors. The p-values returned from the statistical tests are tabulated in Supplementary Table S12, where one can see that 12 out of the 17 p-values returned from the pairwise tests are below the level of significance specified by $\alpha = 0.05$.

TABLE II
PERFORMANCE COMPARISON ON 10 STANDARD BENCHMARK CLASSIFICATION DATASETS

Algo.	err	t_{exe}
CLEAF	**0.0965**	2.3682
SVM	0.1055	0.2905
KNN	0.1132	**0.0291**
RF	0.1210	0.1739
EC	0.1279	248.943
SEQ	0.1965	0.8272
SDKNN	0.1869	0.2631
SAMME	0.1492	1.2603
ESAFIS	0.0996	9.6550
eEnsemble	0.2396	0.2581
PALM	0.1387	3.7591
ALMMo	0.1375	3.1354
SAFL	0.0972	3.5349
SOFE	0.1373	0.4195
LREC	0.1264	17.4841
FWADAB	0.1095	3.4291
SAFLE	0.0982	19.5968
MS-EFS	0.1289	7.9220

2) Examples on real-life non-stationary classification problems: Next, the performance of CLEAF is evaluated on the three real-life non-stationary classification problems, namely, EP, SS and WF, and compared against nine SOTA first-order EFSs, which include SAFIS, ESAFIS, PALM, ALMMo, SAFL, LREC, SEFIS, SAFLE and MS-EFS. In running the experiments, for each dataset, 80% of the samples are randomly selected for online training and the remaining 20% are used for testing. The performances of the 10 first-order EFSs are reported in Table III in terms of average classification error rate (err), average number of rules (N) and average training time costs (t_{exe}). The detailed results are provided in Supplementary Table S13.

One can see from Table III that the classification error rate of CLEAF is on par with the best performing models, such as ESAFIS, SAFL and SAFLE on the three benchmark problems, and its training time cost is the lowest among the nine EFS models, despite that CLEAF has a greater fuzzy rule base than other single-model EFSs. In particular, the average training time cost of CLEAF is 1.81 seconds, which is nearly 40% less than the runner-up, ESAFIS (2.98 seconds). This numerical example demonstrates the advantages of the chunk-wise learning scheme used by CLEAF, allowing the proposed model to attain greater classification accuracy with significantly higher computational efficiency.

In addition, the robustness of CLEAF to temporary fluctuations is further examined on the three real-life datasets by adding 5 dB additive white Gaussian noise (AWGN) to 5% of randomly selected input samples. The prediction performances of CLEAF and its eight competitors in the moderately noisy environments are also reported in Table III for visual clarity, following the same experimental protocol. Detailed results are presented in Supplementary Table S13 as well. One can see from the numerical results that CLEAF outperforms all other

TABLE III
PERFORMANCE COMPARISON ON THREE REAL-LIFE NON-STATIONARY CLASSIFICATION DATASETS

Algo.	Original			AWGN		
	err	N	t_{exe}	err	N	t_{exe}
CLEAF	0.1422	49.8	**1.8059**	**0.1510**	137.2	8.9915
SAFIS	0.3194	36.6	27.5140	0.3097	40.9	58.4938
ESAFIS	0.1407	11.7	123.2599	0.1522	13.4	146.7172
PALM	0.1833	7.3	18.1751	0.1981	1.8	**4.1886**
ALMMo	0.1886	9.9	13.6555	0.2016	10.2	13.2158
SAFL	**0.1405**	31.8	2.9793	0.1515	54.9	13.2986
SEFIS	0.3111	25.0	20.4651	0.3268	25.0	11.4771
LREC	0.1889	**1.2**	57.6602	0.1998	**1.2**	35.6313
SAFLE	0.1464	364.8	77.7047	0.1559	1010.7	91.1525
MS-EFS	0.1975	107.1	98.5413	0.2042	105.1	98.4166

competitors in noisy environments by attaining the lowest classification error rate. Although the size of its fuzzy rule base increases significantly due to the existence of AWGN, the computational efficiency of CLEAF remains at the top level. This example demonstrates the robustness of CLEAF to temporal fluctuations in data patterns introduced by noise.

Furthermore, pairwise Wilcoxon signed rank tests [52] are carried out to compare CLEAF against each of the seven competitors (e.g., ESAFIS, PALM, ALMMo, SAFL, LREC, SAFLE and MS-EFS) involved in the two numerical examples reported in Tables II and III, based on their classification error rates in each experiment. The p-values returned from the statistical tests are tabulated in Supplementary Table S14, where one can see that six out of the seven p-values returned from the pairwise tests are below the level of significance specified by $\alpha = 0.05$. The results reported in Supplementary Tables S12 and S14 demonstrate that CLEAF achieves statistically significant performance improvements over the majority of the competitors.

3) Examples on standard regression problems: Subsequently, the prediction performance of CLEAF is tested on two popular regression problems, namely, California housing and S&P500 under the standard experimental protocols [11], [21]. For California housing dataset, CLEAF is firstly trained online on the training set in a chunk-wise manner and then its performance is evaluated on the testing set [11]. In the case of S&P500, the first half of the augmented dataset is used for online training and the second half is used for online testing under prequential test-then-train framework [21]. The performance of CLEAF is compared against the following SOTA approaches under the same experimental protocols: 1) dynamic evolving neural-fuzzy inference system (DENFIS) [9]; 2) SAFIS [14]; 3) online sequential fuzzy extreme learning machine (OS-Fuzzy-ELM) [15]; 4) ESAFIS [16]; 5) meta-cognitive neuro-fuzzy inference system (McFIS) [17]; 6) correntropy-based evolving fuzzy neural system (CENFS) [22]; 7) particle swarm optimised autonomous learning multimodel system (PSO-ALMMo*) [53]; 8) SAFL [26]; 9) MS-EFS [11]; 10) parsimonious network based on fuzzy inference system (PANFIS) [19]; 11) generic evolving neuro-fuzzy inference system (GENEFIS) [20]; 12) local error optimisation approach (LEOA) [54]; 13) self-evolving fuzzy system (SEFS) [21]; 14) evolving fuzzy system with self-learning/adaptive thresholds (EFS-SLAT) [25]; 15) SAFLE [33]; 16) evolving hierarchical fuzzy system based on variable selection and variable expected output (EHFS-VSVO) [55], and; 17) stacked encoded cascade error feedback deep extreme learning machine (SEC-E-DELM) [56]. The performance comparison results on California housing and S&P500 datasets are reported in Tables IV and V, respectively, using the standard performance measures. Note that the results of the comparative algorithm are obtained directly from the literature [11], [26], [33], [53], [55], [56].

TABLE IV
PERFORMANCE COMPARISON ON CALIFORNIA HOUSING DATASET

Algo.	$rmse$	N	t_{exe}
CLEAF	**0.0674**	116	**0.6135**
DENFIS	0.0715	14	-
SAFIS	0.0988	12	21.9805
OS-Fuzzy-ELM	0.1320	5	6.7252
ESAFIS	0.0892	6	30.2330
McFIS	0.0822	15	-
ALMMo	0.0782	10	1.2423
CENFS	0.0878	**2**	1.5444
PSO-ALMMo*	0.0711	11	1340.5226
SAFL	0.0692	28	0.6316
MS-EFS	0.0704	28	-
EHFS-VSVO	0.0691	48	-
SEC-E-DELM	0.1191	-	5.63

TABLE V
PERFORMANCE COMPARISON ON S&P500 DATASET

Algo.	$ndei$	N	t_{exe}
CLEAF	**0.0120**	28	**0.5908**
PANFIS	0.09	4	55.3
GENEFIS	0.07	**2**	48.3
LEOA	0.1229	52	-
SEFS	0.0182	**2**	2.3467
ALMMo	0.0149	7	2.5766
EFS-SLAT	0.0156	23	-
PSO-ALMMo*	0.0146	7	-
SAFL	0.0121	19	1.8
SAFLE	0.0132	146	17

It can be observed from Tables IV and V that CLEAF offers the highest prediction accuracy with the lowest root mean square error ($rmse$) on California housing and lowest non-dimensional error index ($ndei$) on S&P500 datasets, respectively. During the experiments on the California housing dataset, CLEAF identifies a significantly larger number of rules than others as shown in Table IV. This is due to the involvement of target output variables in fuzzy rule generation, which effectively helps CLEAF capture the local patterns and the potential overlaps, resulting better prediction accuracy. Importantly, the computational efficiency of CLEAF is the highest among the 15 algorithms on the two datasets, thanks to its unique chunk-wise learning mechanism.

4) Examples on large-scale non-stationary classification problems: Then, the test-then-train performance of CLEAF is evaluated on five large-scale non-stationary classification

TABLE VI
PERFORMANCE COMPARISON ON FIVE LARGE-SCALE NON-STATIONARY CLASSIFICATION PROBLEMS UNDER PREQUENTIAL TEST-THEN-TRAIN FRAMEWORK

Algo.	HP		SE		PM		RM		SU	
	err	t_{exe}	err	t_{exe}	err	t_{exe}	err	t_{exe}	err	t_{exe}
CLEAF	**0.0163±0.0017**	**4**	**0.0210±0.0025**	**2**	0.0892±0.0007	**60**	0.0924±0.0015	**62**	**0.2059±0.0003**	**0.6K**
SAFLE	0.0194±0.0014	0.4K	0.0220±0.0013	0.1K	**0.0825±0.0002**	5.8K	**0.0860±0.0007**	10.5K	0.2083±0.0001	85.7K
DEFS	0.0713±0.0180	-	-	-	-	-	-	-	-	-
ProgNN	0.1493±0.0712	0.2K	0.1513±0.0652	0.2K	0.3558±0.0877	0.2K	0.4381±0.1094	0.1K	0.3106±0.0408	345K
DEN	0.0817±0.0417	0.2K	0.2005±0.1928	0.2K	0.4792±0.2260	0.4K	0.3852±0.2173	0.4K	0.3685±0.1006	8K
OMCBoosting	0.1382±0.0373	0.1K	0.1322±0.0385	77	0.6442±0.2051	5K	0.7393±0.0580	5K	0.2287±0.0143	14K
HAT	0.2210±0.1076	3.7K	0.2535±0.1010	0.3K	0.4036±0.1888	0.2K	0.3548±0.1133	0.2K	0.2615±0.0318	16K
pEnsemble	0.0820±0.0190	68	0.0800±0.0570	0.2K	-	-	-	-	0.2556±0.0240	14K
pEnsemble+	0.1240±0.0620	0.2K	0.0800±0.0600	0.2K	-	-	-	-	0.2301±0.0460	35K
ADL	0.0767±0.0263	22	0.0718±0.0579	18	0.3160±0.2417	0.2K	0.2710±0.0935	0.2K	0.2174±0.0280	2.5K
NADINE	-	-	0.0776±0.0640	15	0.2235±0.1509	0.2K	0.2549±0.0750	192	0.2197±0.0300	1.5K
DSSCN	0.0875±0.0179	-	0.0950±0.0409	-	-	-	-	-	0.2300±0.0140	-
DEVFNN	0.0843±0.0176	-	0.0830±0.0500	-	-	-	-	-	0.2200±0.0190	-
MUSE-RNN	0.0736±0.0215	0.3K	0.0763±0.0611	0.1K	0.1613±0.1342	0.4K	0.2373±0.0490	190	0.2186±0.0160	21K
LREC	0.0786±0.0179	-	0.0712±0.0558	-	-	-	-	-	0.2200±0.0140	-
X-Fuzz	0.0780±0.0175	-	0.0704±0.0556	-	-	-	-	-	02108±0.0134	-

problems, which include hyperplane (HP), SEA (SE), MNIST permutations (PM), MNIST rotations (RM) and SUSY (SU). To facilitate computation, principal component analysis is used to reduce the dimensionality of PM and RM from 784 to 28 prior to the experiments, and the value of the externally controlled parameter D_o is adjusted to $e^{-\frac{1}{2}}$ for CLEAF in the experiments on PM, RM and SU. The performance of CLEAF is compared with the following 15 SOTA approaches designed for non-stationary streaming data classification: 1) SAFLE [33]; 2) dynamic evolving fuzzy system (DEFS) [28]; 3) progressive neural network (ProgNN) [57]; 4) dynamically expandable network (DEN) [58]; 5) online multiclass boosting (OMCBoosting) [59]; 6) hard attention to the task (HAT) classifier [60]; 7) parsimonious ensemble (pEnsemble) [35]; 8) pEnsemble+ [61]; 9) autonomous deep learning (ADL) classifier [62]; 10) neural network with dynamically evolved capacity (NADINE) [63]; 11) deep stacked stochastic configuration networks (DSSCN) [64]; 12) deep evolving fuzzy neural network (DEVFNN) [65]; 13) multilayer self-evolving recurrent neural network (MUSE-RNN) [66]; 14) LREC [49], and; 15) X-Fuzz [29]. The comparison results are tabulated in Table VI in terms of classification error rate (err) and the execution time consumption (t_{exe}). Note that the results of the 15 comparative algorithm are obtained directly from the literature [29], [33], [63], [66]. Table VI illustrates that CLEAF consistently outperforms or matches the top-performing comparative models across five large-scale datasets, achieving the lowest error rates on HP, SE and SU, and the second lowest error rates on PM and RM. Additionally, CLEAF demonstrates the highest computational efficiency across the experiments on the five datasets, significantly higher than the SOTA alternatives.

5) Examples on visual classification problems: Finally, the classification performance of CLEAF is evaluated on two visual classification datasets, MNIST and Fashion MNIST (FMNIST). In each experiment, a ResNet20 model [67] with an input size of 28×28 pixels is trained for 100 epochs on the training images using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 32. The trained ResNet20 model is then used for feature extraction by removing the final softmax layer, yielding a 64×1 feature vector for each image. CLEAF is trained online using the feature vectors of the training images and tested on the feature vectors of the testing images. The average classification error rates of CLEAF on the testing sets are reported in Table VII. The baseline results of the ResNet20 model are also given in the same table. Additionally, the performance of CLEAF is compared against the following five deep neuro-fuzzy (hybrid) models: 1) multiple attentions-based multilevel hybrid-guided deep fuzzy convolutional neural network (MAMH-DFCNN) [68]; 2) deep image feature learning with fuzzy rules (DIFL-FR) [69]; 3) deep convolutional neuro-fuzzy inference system (DCNFIS) [70]; 4) deep convolutional fuzzy system (DCFS) [71], and; 5) highly interpretable deep Takagi-Sugeno-Kang fuzzy classifier (HID-TSK-FC) [72]. The results of the five models are obtained from [68], [69] and are presented in Table VII.

Table VII shows that CLEAF achieves higher classification accuracy on the MNIST and FMNIST testing images compared to the baseline ResNet20 model, performing on par with state-of-the-art deep neuro-fuzzy hybrid models such as MAMH-DFCNN and DIFL-FR. Notably, CLEAF learns from data in a chunk-wise, single-pass manner, whereas these deep neuro-fuzzy hybrid models require iterative training to achieve high classification accuracy, making them computationally expensive and time-consuming.

6) Demonstration of adaptability and interpretability: For better demonstration, the changes of the classification error rate (err) on the testing images and the number of rules (N) identified from the training images across chunks on the MNIST dataset during one particular experiment are plotted in Fig. 5. For visual clarity, only the results from the first 10,000 training images (namely, 50 data chunks) are shown in this figure. As shown in Fig. 5, CLEAF identifies more fuzzy rules over time by learning from consecutive data chunks with its classification accuracy increases gradually. As CLEAF

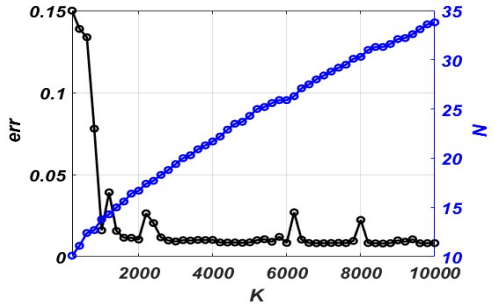


Fig. 5: Evolution of classification error rate and number of fuzzy rules on MNIST.

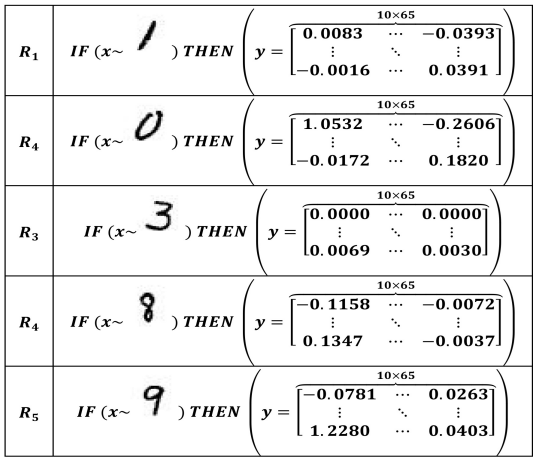


Fig. 6: Fuzzy rules learned from training images of MNIST.

processes one data chunk at one time, some mild fluctuations in performance are expected.

Five fuzzy rules learned from the training set of MNIST are presented in Fig. 6. As the learned prototypes are the embeddings of real images, the antecedent parts of the fuzzy rules are visualised using the corresponding images for clearer illustration. One can see that each of the five fuzzy rules represents a distinctive pattern of images of the corresponding class. Additionally, the respective normalised firing strengths and outputs of the five fuzzy rules in response to four selected testing images are reported in Supplementary Tables S15 and S16. As shown in the two tables, given the first testing image, $\boldsymbol{R}_5$ exhibits the highest firing strength. Moreover, the 10^{th} element of its output has the largest value, indicating that this image is most likely the digit "9". Similarly, $\boldsymbol{R}_4$ and $\boldsymbol{R}_3$ produce the highest firing strengths in response to the second and third testing images with the corresponding elements in their output vectors being the highest as well, indicting that the two images are most likely the digits "8" and "3". Because the fourth testing image corresponds to the digit '7', which does not match the prototypes of the five rules, all rules produce low firing strengths for this image. This illustrative example showcases the human-interpretable prototypes and the transparent internal reasoning provided by CLEAF.

TABLE VII
PERFORMANCE COMPARISON ON TWO IMAGE CLASSIFICATION PROBLEMS

Algo.	MNIST	FMNIST
CLEAF	0.0065	0.0722
ResNet20*	0.0071	0.0739
MAMH-DFCNN	0.0048	**0.0577**
DIFL-FR	0.0061	0.0823
DCNFIS	**0.0045**	0.0912
DCFS	0.3885	0.4132
HID-TSK-FC	0.3414	0.4803

*Baseline

D. Ablation analysis

An ablation analysis is carried out and detailed in Supplementary Section G to demonstrate the influence of the chunk-wise learning scheme and the leverage of both input and target output variables in fuzzy rule identification on the performance of CLEAF. The ablation analysis results confirm that:

- The proposed chunk-wise prototype identification scheme enables CLEAF to better capture the multi-model distributions of data streams and attain better prediction performance with less fuzzy rules;
- The proposed chunk-wise fuzzily weighted recursive least squares (FWRLS) algorithm significantly improves the overall computational efficiency of CLEAF by limiting the updating of consequent parameters and covariance matrices associated with the fuzzy rules to once per data chunk;
- The utilisation of both input and target output variables in calculating the data density enhances CLEAF's ability to capture the overlaps between different data patterns, resulting in higher prediction accuracy.

E. Discussions

To summarise, the systematic experimental studies presented in Section IV.C have collectively and consistently demonstrated the superior performance of CLEAF over the SOTA comparative approaches on a wide range of benchmark datasets for classification and regression. Compared to alternative first-order EFSs that rely on sample-wise consequent parameter learning, the proposed chunk-wise learning scheme not only significantly improves CLEAF's computational efficiency, but also helps the proposed model to precisely approximate non-stationary streaming problems with a more compact rule base by reducing its sensitivity to temporary fluctuations in data streams. The illustrative examples presented at the end of Section IV.C further demonstrate the system evolving process overtime when learning from consecutive data chunks as well as the interpretability provided by CLEAF.

In addition, the sensitivity analysis presented in Section IV.B highlights the substantial potential for performance improvement through appropriate tuning of CLEAF's externally controlled parameters. Finally, the ablation study in Section IV.D verifies the effectiveness of the proposed chunk-wise learning scheme in improving the performance of CLEAF, in terms of both prediction accuracy and computational efficiency.

V. Conclusion

This paper has presented a novel evolving fuzzy system (EFS), named chunk-wise learning evolving autonomous fuzzy system (CLEAF), for learning from data streams through chunk-wise structure evolution and parameter learning. The key innovations of CLEAF lie in its chunk-wise learning scheme that considers spatial dissimilarities in both input and output spaces for prototype identification, and in the proposed chunk-wise fuzzily weighted recursive least squares (FWRLS) algorithm for efficient consequent parameter updating. These innovations enable CLEAF to achieve higher computational efficiency, improved prediction accuracy, and stronger resilience to temporary fluctuations compared with conventional EFSs. Extensive experimental studies have systematically validated the effectiveness and robustness of the proposed approach.

However, the proposed CLEAF also has two limitations, observed during the experimental investigations. The first is the significantly higher computational complexity when dealing with high-dimensional data streams. The second is the reduced ability to adequately capture the dynamic behaviour of streaming data, particularly when the chunk size is large. These limitations are inherent to recursive least squares (RLS)-based algorithms and chunk-wise learning, respectively. In addition, a comprehensive theoretical analysis the convergence of CLEAF is currently lacking. Consequently, several open issues remain to be further investigated, including:

1) High dimensionality. The chunk-wise learning scheme designed for CLEAF effectively improves its computational efficiency. Nonetheless, despite the modifications introduced to the FWRLS algorithm, updating the covariance matrices associated with the consequent parameters remains an essential step, which can be computationally expensive if the dimensionality of data is high. This is an inherent issue associated with FWRLS. To facilitate learning from high-dimensional data streams, more advanced consequent parameter learning algorithms are needed. Dimensionality compression techniques that are suitable for online nonstationary environments, e.g., very sparse random projection [11] or rough and fuzzy-rough-based feature selection [73], could also help to reduce the computational complexity.

2) Externally controlled parameters. It can be seen from the sensitivity analysis that there is a significant potential for performance improvement by adjusting the externally controlled parameters of CLEAF, based on the characteristics of the domain data. Hence, it would be beneficial if CLEAF would be able to self-adjust these parameters after the initial values have been provided, particularly when applied to less studied problem domains. For example, CLEAF could reduce its chunk size to enhance adaptability when an increased number of novel data patterns is detected within a fixed time window.

3) Stability analysis. Finally, numerical examples have demonstrated the effectiveness and validity of CLEAF on a variety of popular classification and regression benchmark problems, and it has been theoretically proven that its consequent parameter matrices and outputs remain bounded for any bounded input data chunks. However, it remains unclear whether the modifications made to the FWRLS algorithm to enable chunk-wise updating of the consequent parameters have any significant impact on the convergence of CLEAF. Further investigation in this direction would be highly valuable.

References

[1] J. Lu, G. Ma, and G. Zhang, "Fuzzy machine learning: a comprehensive framework and systematic review," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 7, pp. 3861–3878, 2024,

[2] X. Gu et al., "Autonomous learning for fuzzy systems: a review," *Artificial Intelligence Review*, pp. 1–47, 2022.

[3] A. Laghari et al., "Deep residual-dense network based on bidirectional recurrent neural network for atrial fibrillation detection," *Scientific Reports*, vol. 13, no. 1, pp. 15109, 2023.

[4] S. Yin et al., "Brain CT image classification based on Mask R-CNN and attention mechanism," *Scientific Reports*, vol. 14, no. 1, pp. 29300, 2024.

[5] M. Munir et al., "Enhancing gene mutation prediction with sparse regularized autoencoders in lung cancer radiomics analysis," *IEEE Access*, vol. 13, pp. 7407–7425, 2025.

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

[7] L. Yan et al.,"OSSEFS: an online semi-supervised ensemble fuzzy system for data streams learning with missing values," *Expert Systems with Applications*, vol. 255, p. 124695, 2024.

[8] G. Van de Ven, T. Tuytelaars, and A. Tolias, "Three types of incremental learning," *Nature Machine Intelligence*, vol. 4, no.12, pp. 1185-1197, 2022.

[9] N. 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.

[10] P. Angelov and D. 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.

[11] H. Huang et al.,"Multilayer stacked evolving fuzzy system combined with compressed representation learning," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 4, pp. 2223–2234, 2024.

[12] I. Skrjanc et al.,"Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey," *Information Sciences*, vol. 490, pp. 344–368, 2019

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

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

[15] H. Rong, et al., "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.

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

[17] K. Subramanian and S. Suresh, "A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system," *Applied Soft Computing*, vol. 12, no. 11, pp. 3603–3614, 2012.

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

[19] M. Pratama et al., "PANFIS: a novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, 2014.

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

[21] D. Ge and X. 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, 2018.

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

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

[24] I. Skrjanc et al., "Inner matrix norms in evolving Cauchy possibilistic clustering for classification and regression from data streams," *Information Sciences*, vol. 478, pp. 540–563, 2019.

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

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

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

[28] Z. Mei, T. Zhao, and X. Gu,"A dynamic evolving fuzzy system for streaming data prediction," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 8, pp. 4324 - 4337, 2024.

[29] M. Ferdaus et al., "X-Fuzz: an evolving and interpretable neurofuzzy learner for data streams," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 8, pp. 4001 - 4012, 2024.

[30] P. Angelov, X. Gu, and J. Principe, "Autonomous learning multimodel systems from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 4, 2018.

[31] S. Samanta, M. Pratama, and S. Sundaram, "A novel spatio-temporal fuzzy inference system (SPATFIS) and its stability analysis," *Information Sciences*, vol. 505, pp. 84–99, 2019.

[32] A. Laghari, V. Estrela, and S. Yin, "How to collect and interpret medical pictures captured in highly challenging environments that range from nanoscale to hyperspectral imaging," *Current Medical Imaging*, vol. 20, no. 1, pp. e281222212228, 2024.

[33] X. Gu, "Self-adaptive fuzzy learning ensemble systems with dimensionality compression from data streams," *Information Sciences*, vol. 634, pp. 382–399, 2023.

[34] 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.

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

[36] J. Jiang et al., "Dynamic incremental ensemble fuzzy classifier for data streams in green internet of things," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 3, pp. 1316–1329, 2022.

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

[38] E. Lughofer, M. Pratama, and I. Skrjanc, "Online bagging of evolving fuzzy systems," *Information Sciences*, vol. 570, pp. 16–33, 2021.

[39] X. Gu, G. Howells, and H. Yuan, "A soft prototype-based autonomous fuzzy inference system for network intrusion detection," *Information Sciences*, vol. 677, pp. 120964, 2024.

[40] G. Xue, Y. Yang, and J. Wang, "Adaptive Yager T-norm-based Takagi–Sugeno–Kang fuzzy systems," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 12, pp. 9802–9815, 2025.

[41] X. Gu, P. Angelov, and Z. Zhao, "Self-organizing fuzzy inference ensemble system for big streaming data classification," *Knowledge Based Systems*, vol. 218, p. 106870, 2021.

[42] Y. Teng et al., "Self-supervised ensemble for extreme imbalance data streams with concept drift," *Neurocomputing*, vol. 665, p. 132154, 2026.

[43] H. Rong et al., "Stability of evolving fuzzy systems based on data clouds," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2774–2784, 2018.

[44] P. Cunningham and S. Delany, "K-nearest neighbour classifiers-a tutorial," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–25, 2021.

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

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

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

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

[49] M. Ferdaus, et al., "Significance of activation functions in developing an online classifier for semiconductor defect detection," *Knowledge-Based Systems*, vol. 248, p. 108818, 2022.

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

[51] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[52] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*, New York: Springer, 1992, pp. 196–202.

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

[54] D. Ge and X. 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.

[55] P. Wang, et al., "An evolving hierarchical fuzzy system based on variable selection and variable expected outputs," *International Journal of Fuzzy Systems*, pp. 1–17, 2025.

[56] W. Khan, et al. "Stacked encoded cascade error feedback deep extreme learning machine network for manufacturing order completion time," *Journal of Intelligent Manufacturing*, vol. 36, no. 2, pp. 1313-1339, 2025.

[57] A. Rusu et al., "Progressive neural networks," *arXiv Prepr. arXiv1606.04671*, 2016

[58] J. Yoon et al., "Lifelong learning with dynamically expandable networks," *arXiv Prepr. arXiv1708.01547*, 2017.

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

[60] J. Serra et al., "Overcoming catastrophic forgetting with hard attention to the task," in *International Conference on Machine Learning*, 2018, pp. 4548–4557.

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

[62] A. Ashfahani and M. Pratama, "Autonomous deep learning: continual learning approach for dynamic environments," in *SIAM International Conference on Data Mining*, 2019, pp. 666–674.

[63] M. Pratama et al., "Automatic construction of multi-layer perceptron network from streaming examples," in *International Conference on Information and Knowledge Management*, 2019, pp. 1171–1180.

[64] M. Pratama and D. Wang, "Deep stacked stochastic configuration networks for lifelong learning of non-stationary data streams," *Information Sciences*, vol. 495, pp. 150–174, 2019.

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

[66] M. Das et al., 'MUSE-RNN: a multilayer self-evolving recurrent neural network for data stream classification," in *IEEE International Conference on Data Mining*, 2019, pp. 110–119.

[67] L. Heim et al.,"Measuring what really matters: optimizing neural networks for TinyML," *arXiv preprint arXiv:2104.10645*, 2021.

[68] J. Zhao et al.,"A multiple attentions-based multilevel hybrid-guided deep fuzzy convolutional neural network for image recognition," *IEEE Transactions on Fuzzy Systems*, vol. 33, no. 8, pp. 2614-2628, 2025.

[69] X. Ma et al.,"Deep image feature learning with fuzzy rules," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 8, no. 1, pp. 724-737, 2024.

[70] M. Yeganejou et al.,"An end-to-end trainable deep convolutional neuro-fuzzy classifier," in *IEEE International Conference on Fuzzy Systems*, 2022, pp. 1–7.

[71] L. Wang, "Fast training algorithms for deep convolutional fuzzy systems with application to stock index prediction," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1301-1314, 2020.

[72] Y. Zhang, H. Ishibuchi, and S. Wang, "Deep Takagi–Sugeno–Kang fuzzy classifier with shared linguistic fuzzy rules," *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 3, pp. 1535–1549, 2018.

[73] R. Jensen and Q. Shen, "Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 12, pp. 1457-1471, 2004.