University of Kent

Kent Academic Repository

Gu, Xiaowei (2023) *A Dual-Model Semi-Supervised Self-Organizing Fuzzy Inference System for Data Stream Classification*. Applied Soft Computing, 136 . ISSN 1568-4946.

Downloaded from
https://kar.kent.ac.uk/99736/ The University of Kent's Academic Repository KAR

The version of record is available from
https://doi.org/10.1016/j.asoc.2023.110053

This document version
Publisher pdf

DOI for this version

Licence for this version
CC BY (Attribution)

Additional information

Versions of research works

Versions of Record
If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts
If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in ***Title of Journal*** , Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries
If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our Take Down policy (available from https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies).

Applied Soft Computing 136 (2023) 110053

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A dual-model semi-supervised self-organizing fuzzy inference system for data stream classification

Xiaowei Gu

School of Computing, University of Kent, Canterbury CT2 7NZ, UK

ARTICLE INFO

Article history:
Received 10 July 2022
Received in revised form 6 January 2023
Accepted 16 January 2023
Available online 31 January 2023

Keywords:
Semi-supervised learning
Fuzzy inference
Data stream
Evolving fuzzy system

ABSTRACT

Semi-supervised learning from data streams is widely considered as a highly challenging task to be further researched. In this paper, a novel dual-model self-organizing fuzzy inference system composed of two recently introduced evolving fuzzy systems (EFSs) is proposed for semi-supervised learning from data streams in infinite delay environments. After being primed with a small amount of labelled data during the warm-up period, the proposed model is able to continuously self-learn and self-expand its knowledge base from unlabelled data on a chunk-by-chunk basis with minimal human expert involvement. Thanks to its dual-model structure, the proposed model combines the merits of the two EFS models such that it can continuously identify new prototypes from new pseudo-labelled data to self-improve its knowledge base whilst keeping the impact of pseudo-labelled errors on its decision-making minimized. Numerical examples based on various benchmark problems demonstrate the efficacy of the proposed method, showing its strong potential in real-world applications by offering higher classification accuracy over the state-of-the-art competitors whilst retaining high computational efficiency.

© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Supervised learning and unsupervised learning are two hotly researched areas in the machine learning domain [1]. Supervised learning methods [2] utilize a set of input data samples with the corresponding output values to construct a predictive (classification or regression) model that can estimate the outputs given new unseen input samples. Unsupervised learning methods [3], on the other hand, attempt to disclose the underlying structure of the input samples and cluster them into different groups according to their statistical characteristics without any output value being provided.

Semi-supervised learning [4,5], as a hybridization of supervised learning and unsupervised learning, is an important branch of machine learning. Unlike supervised learning methods that often suffer from the labelling bottleneck due to the lack of sufficient high-quality labelled training data, semi-supervised learning methods are capable of constructing a stronger predictive model from a greater amount of unlabelled data samples together with a smaller amount of labelled ones. As such, semi-supervised learning methods require much less human labelling efforts than supervised learning methods and can achieve greater prediction performance in the application scenarios, where labelled training samples are scarce and expensive to obtain [6].

Mainstream semi-supervised classification methods can be broadly divided into two categories, namely, inductive methods and transductive methods [5]. Inductive methods generally extend supervised classifiers, e.g., support vector machine (SVM), decision tree (DT), multi-layered perceptron (MLP), k-nearest neighbour (kNN), to incorporate unlabelled data [7–10]. Their primary goal is to construct a classifier from both labelled and unlabelled data that can be used for classifying any new samples in the same data space. Transductive methods are all graph-based, and they do not produce predictive model but directly predict the class labels of unlabelled samples presented during the learning process. This is achieved by constructing a graph structure from data connecting samples of similar characteristics such that label information from labelled data can be gradually propagated through graph edges to all unlabelled samples [11]. Hence, the key difference between inductive methods and transductive methods is that the former optimize the classifiers using labelled and unlabelled data together, whilst the latter optimize the predictions over unlabelled data directly based on the labelled ones [5]. To date, the best examples of inductive methods include semi-supervised SVM (S^3VM) [8], semi-supervised extreme learning machine [12], self-training [13], co-training [9,14], ensemble learning [15], etc., and well-known transductive methods include local and global consistence (LGC) [16], Laplacian SVM (LSVM) [17], greedy gradience max-cut (GGMC) [18], anchor graph regularization (AGR) [19], efficient anchor graph regularization (EAGR) [11], etc.

E-mail address: x.gu@kent.ac.uk.

https://doi.org/10.1016/j.asoc.2023.110053
1568-4946/© 2023 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

Wrapper methods, which include self-training, co-training and ensemble learning, are among the most widely used approaches for semi-supervised learning exploiting the idea of "pseudo label" [4,5]. A typical wrapper method iteratively trains one or multiple classifiers with the enlarged labelled training set augmented by a group of selected unlabelled samples with high-confidence predicted labels. These predicted labels produced by the classifiers are the so-called "pseudo labels". Compared with other semi-supervised learning methods, wrapper methods are, generally, simpler and can be used to extend any given mainstream classifiers in a straightforward manner by allowing unlabelled samples to directly participate the model training process as a part of the training set [20]. However, conventional wrapper methods require both the original labelled samples and all the pseudo-labelled samples to be presented to the classifiers at each learning iteration. This requirement of iterative computation limits the conventional wrapper methods (as well as the vast majority of alternative semi-supervised learning methods) to offline application scenarios, making wrapper methods computationally expensive. Another issue caused by iterative computation is the inevitable error propagation, which may significantly deteriorate the prediction accuracy of wrapper methods. Although there have been a few wrapper methods proposed recently that are designed for data stream classification, e.g., parsimonious network (ParsNet) [21], skip-connected evolving recurrent network [22], self-evolving mutually-operative recurrent network-based model [23], self-training hierarchical prototype-based classifier (STHP) [20], weakly supervised scalable teacher forcing network (WeScatterNet) [24], and semi-supervised self-organizing fuzzy inference system (S^3 OFIS) [25], semi-supervised learning from data streams in infinite delay environments is still widely considered as a highly challenging task, remaining under researched [24,25].

In this paper, a novel duel-model semi-supervised self-organizing fuzzy inference system (DMS^{3}OF) is proposed for tackling data stream classification problems in infinite delay environments. DMS^{3}OF employs the two recently introduced evolving fuzzy systems (EFSs), namely, simplified self-organizing fuzzy inference system (SOFIS+) [26] and self-organizing fuzzy belief inference system (SOFBIS) [27] as its implementation basis. Both EFS models have a transparent prototype-based system structure and perform learning from data streams on a chunk-by-chunk basis through a human interpretable process. By utilizing the pseudo labelling technique, DMS^{3}OF is able to continuously self-expand its knowledge base from unlabelled streaming data in a single pass, non-iterative, computationally efficient manner with minimal human expertise involvement. Very importantly, thanks to the unique semi-supervised learning scheme designed specifically for its dual-model structure, DMS^{3}OF combine the merits of both EFS models such that it is able to produce pseudo labels with high precision whilst effectively limits error-propagation within its knowledge base, thereby constructing a stronger prediction model and achieving greater classification performance. To summarize, key features of the proposed DMS^{3}OF that differentiate itself from mainstream semi-supervised learning methods are as follows.

(1) A novel fuzzy inference system with a dual-model structure to semi-supervised learn from data streams on a chunk-by-chunk basis;
(2) A single-pass semi-supervised learning scheme designed for the dual-model structure that combines the advantages of the two EFS models;
(3) The capability to continuously self-improve the knowledge base by capturing new prototypes from unlabelled streaming samples with the most reliable pseudo labels;
(4) The stronger resistance to error propagation within the knowledge base by confining pseudo-labelling errors to only these prototypes directly affected.

The remainder of this paper is organized as follows. Section 2 summarizes the technical details of SOFIS+ and SOFBIS as the theoretical background of DMS^{3}OF. The proposed DMS^{3}OF is detailed in Section 3. Numerical examples are presented in Section 4 as the proof of concept. This paper is concluded by Section 5.

2. Preliminaries

In this section, technical details of SOFIS+ and SOFBIS are recalled briefly to make this paper self-contained. Both SOFIS+ and SOFBIS are supervised EFSs that learn from labelled training data on a chunk-by-chunk basis. They serve as the implementation basis of the proposed DMS^{3}OF system.

First of all, let $\mathbf{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K, \boldsymbol{x}_{K+1}, \ldots, \boldsymbol{x}_{K+J}\}$ ($\boldsymbol{x}_k = [x_{k,1}, x_{k,2}, \ldots, x_{k,M}]^T \in \mathbf{X}$) be a particular data stream in a M dimensional real space, $\mathfrak{R}^M$, where K is the total number of labelled data samples, J is the total number of unlabelled data samples and there is $K \ll J$. The subscript k denotes the time instance at which the kth data sample, $\boldsymbol{x}_k$ is observed. It is assumed that $\mathbf{X}$ is composed of data samples of C different classes, where y_k is the class label of $\boldsymbol{x}_k$ and there is $y_k \in \{1, 2, \ldots, C\}$ ($k = 1, 2, \ldots, K+J$). As this study considers the infinite delay problems, only the class labels of the first K data samples are assumed to be available for model initialization, namely, $\mathbf{Y} = \{y_1, y_2, \ldots, y_k, \ldots, y_K\}$. The class labels of the remaining J data samples, namely, $\boldsymbol{x}_{K+1}, \boldsymbol{x}_{K+2}, \ldots, \boldsymbol{x}_{K+J}$ are unknown. It is also assumed that the data stream $\mathbf{X}$ continuously arrives in chunks, denoted as $\mathbf{X}_1, \ldots, \mathbf{X}_N, \mathbf{X}_{N+1}, \ldots, \mathbf{X}_{N+U}$. Out of the $N+U$ data chunks, the first N chunks are labelled ones with their corresponding class labels denoted as $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_N$, and the other U chunks are unlabelled ones. The cardinality of $\mathbf{X}_n$ is denotes as L_n, and there are $\sum_{n=1}^{N} L_n = K$, $\sum_{n=N+1}^{U} L_n = J$ and $\sum_{n=1}^{N+U} L_n = K+J$. Note that different data chunks are not necessarily of the same size. The level of granularity for both SOFIS+ and SOFBIS is set as G, which is externally controlled by users.

2.1. SOFIS+

2.1.1. General architecture and decision making policy

SOFIS+ is composed of C prototype-based fuzzy IF-THEN rules in the following form [26]:

$$\mathbf{R}^c: \quad \begin{array}{c} IF\ \left(\boldsymbol{x} \sim \boldsymbol{p}_1^c\right) OR \left(\boldsymbol{x} \sim \boldsymbol{p}_2^c\right) OR \ldots OR \left(\boldsymbol{x} \sim \boldsymbol{p}_{P^c}^c\right) \\ THEN\ (D^c) \end{array} \tag{1}$$

where "$\sim$" denotes the similarity; $\boldsymbol{p}_k^c \in \mathbf{P}^c$ represents the kth prototype of the cth class; $\mathbf{P}^c$ stands for the collection of all prototypes of the cth class; P^c is the cardinality of $\mathbf{P}^c$; D^c is the conclusion "$\boldsymbol{x}$ belongs to class c".

As shown in Eq. (1), the cth fuzzy rule $\mathbf{R}^c$ consists of a number of prototypes identified from data samples of the cth class. Each rule corresponds to a particular class in the data. Prototypes associated with each fuzzy rule are connected by logic "OR" connectives.

During the decision-making process, given an unlabelled sample $\boldsymbol{x}$, the confidence score of the fuzzy rule $\mathbf{R}^c$ will be calculated based on the squared L_2 distance between $\boldsymbol{x}$ and the nearest prototype associated with $\mathbf{R}^c$ [25,26]:

$$\omega^c(\boldsymbol{x}) = \frac{\lambda^c(\boldsymbol{x})}{\sum_{i=1}^{C} \lambda^i(\boldsymbol{x})} \tag{2}$$

where $\lambda^c(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{p}^c_{*1}\|_2^2}{\sigma_G^2}\right)$; σ_G is a data-driven kernel width derived from data at the Gth level of granularity controlled by users; $\|\boldsymbol{x}-\boldsymbol{y}\|_2 = \sqrt{\sum_{i=1}^{M}(x_i-y_i)^2}$; $\boldsymbol{p}^c_{*1} = \underset{\boldsymbol{p}\in\mathbf{P}^c}{\operatorname{argmin}}(\|\boldsymbol{x}-\boldsymbol{p}\|_2)$ $(c = 1, 2, \ldots, C)$.

Then, the class label of $\boldsymbol{x}$ is determined by the fuzzy rule that produces the highest confidence score:

$$\hat{y} = c^*; \quad c^* = \underset{c=1,2,\ldots,C}{\operatorname{argmax}}(\omega^c(\boldsymbol{x})) \tag{3}$$

2.1.2. Learning policy

The learning policy of SOFIS+ is described as follows. As aforementioned, SOFIS+ learns from data on a chunk-by-chunk basis. SOFIS+ begins a new learning cycle when a new data chunk (assuming the nth one, $\mathbf{X}_n$) is available. Note that the processed data chunks will be discarded at the end of the learning cycles such that SOFIS+ can maintain its computation- and memory-efficiency.

Stage 1. Prototype identification. Given a new data chunk $\mathbf{X}_n$ and the corresponding class labels $\mathbf{Y}_n$, SOFIS+ firstly derives the data-driven kernel width, σ_G from $\mathbf{X}_n$ using Eq. (4):

$$\sigma_G = \sqrt{\frac{\sum_{j=1}^{n} L_j \sigma_{j,G}^2}{\sum_{j=1}^{n} L_j}} \tag{4}$$

where there are $(g = 1, 2, \ldots, G)$:

$$\sigma_{n,g} = \sqrt{\frac{\sum_{j=1}^{L_n-1}\sum_{k=j+1}^{L_n} w_{g,j,k}\left\|\boldsymbol{x}_{n,j}-\boldsymbol{x}_{n,k}\right\|_2^2}{\sum_{j=1}^{L_n-1}\sum_{k=j+1}^{L_n} w_{g,j,k}}} \tag{5a}$$

$$w_{g,j,k} = \begin{cases} 1, & \left\|\boldsymbol{x}_{n,j}-\boldsymbol{x}_{n,k}\right\|_2 \le \sigma_{n,g-1} \\ 0, & else \end{cases} \tag{5b}$$

$$\sigma_{n,0} = \sqrt{\frac{2\sum_{j=1}^{L_n-1}\sum_{k=j+1}^{L_n}\left\|\boldsymbol{x}_{n,j}-\boldsymbol{x}_{n,k}\right\|_2^2}{(L_n-1)L_n}} \tag{5c}$$

Next, SOFIS+ divides $\mathbf{X}_n$ into C different nonoverlapping subsets denoted as $\mathbf{X}_n^1, \mathbf{X}_n^2, \ldots, \mathbf{X}_n^C$ according to $\mathbf{Y}_n$ for prototype identification, where $\mathbf{X}_n^c$ is composed of data samples of the cth class only $(c = 1, 2, \ldots, C)$. Given the cth subset $\mathbf{X}_n^c$, the radius of area of influence around each prototype of the cth class at the Gth level of granularity, denoted as γ_G^c is estimated by Eq. (6):

$$\gamma_G^c = \sqrt{\frac{\sum_{j=1}^{n} L_j^c \left(\gamma_{j,G}^c\right)^2}{\sum_{j=1}^{n} L_j^c}} \tag{6}$$

where L_n^c is the cardinality of $\mathbf{X}_n^c$; and there are $(g = 1, 2, \ldots, G)$:

$$\gamma_{n,g}^c = \sqrt{\frac{\sum_{j=1}^{L_n^c-1}\sum_{k=j+1}^{L_n^c} w_{g,j,k}^c\left\|\boldsymbol{x}_{n,j}^c-\boldsymbol{x}_{n,k}^c\right\|_2^2}{\sum_{j=1}^{L_n^c-1}\sum_{k=j+1}^{L_n^c} w_{g,j,k}^c}} \tag{7a}$$

$$w_{g,j,k}^c = \begin{cases} 1, & \left\|\boldsymbol{x}_{n,j}^c-\boldsymbol{x}_{n,k}^c\right\|_2 \le \gamma_{n,g-1}^c \\ 0, & else \end{cases} \tag{7b}$$

$$\gamma_{n,0}^c = \sqrt{\frac{2\sum_{j=1}^{L_n^c-1}\sum_{k=j+1}^{L_n^c}\left\|\boldsymbol{x}_{n,j}^c-\boldsymbol{x}_{n,k}^c\right\|_2^2}{(K_n-1)K_n}} \tag{7c}$$

Note that γ_G^c is obtained in a similar way as σ_G. However, the key difference between γ_G^c and σ_G is that γ_G^c is derived based on the mutual distances between data samples of the same class, whilst σ_G is derived based on the mutual distances between data samples of all classes. Both γ_G^c and σ_G are directly calculated from data and are guaranteed to be meaningful.

Then, the first prototype of the cth class is identified as the first sample of $\mathbf{X}_n^c$, namely, $\boldsymbol{p}^c_{n,P_n^c} \leftarrow \boldsymbol{x}^c_{n,1}$ $(P_n^c \leftarrow 1)$. The support of $\boldsymbol{p}^c_{n,P_n^c}$, namely, the number of data samples associated with it is set as: $S^c_{n,P_n^c} \leftarrow 1$. The remaining prototypes of the cth class are identified one by one from the remaining samples of $\mathbf{X}_n^c$ using Condition 1 $(k = 1, 2, \ldots, L_n^c)$ [25,26]:

$$Cond.\,1: \quad \begin{array}{c} if\left(\min_{\boldsymbol{p}\in\mathbf{P}_n^c}\left(\left\|\boldsymbol{x}_{n,k}^c-\boldsymbol{p}\right\|_2\right) > \gamma_G^c\right) \\ then\left(\mathbf{P}_n^c \leftarrow \mathbf{P}_n^c \cup \left\{\boldsymbol{x}_{n,k}^c\right\}\right) \end{array} \tag{8}$$

where $\mathbf{P}_n^c$ denotes the collection of prototypes identified from $\mathbf{X}_n^c$.

If $\boldsymbol{x}_{n,k}^c$ satisfies Condition 1, it represents a new data pattern that is different from the known ones identified from $\mathbf{X}_n^c$ previously and, thus, is added to $\mathbf{P}_n^c$:

$$P_n^c \leftarrow P_n^c + 1; \quad \boldsymbol{p}^c_{n,P_n^c} \leftarrow \boldsymbol{x}^c_{n,k}; \quad S^c_{n,P_n^c} \leftarrow 1 \tag{9}$$

Otherwise (namely, Condition 1 is unsatisfied), $\boldsymbol{x}_{n,k}^c$ is used for updating the nearest prototype within $\mathbf{P}_n^c$:

$$\boldsymbol{p}^c_{n,*} \leftarrow \boldsymbol{p}^c_{n,*} + \frac{\boldsymbol{x}^c_{n,k}-\boldsymbol{p}^c_{n,*}}{S^c_{n,*}+1}; \quad S^c_{n,*} \leftarrow S^c_{n,*} + 1 \tag{10}$$

where $\boldsymbol{p}^c_{n,*} = \underset{\boldsymbol{p}\in\mathbf{P}_n^c}{\operatorname{argmin}}\left(\left\|\boldsymbol{x}^c_{n,k}-\boldsymbol{p}\right\|_2\right)$.

After prototypes $\mathbf{P}_n^1, \mathbf{P}_n^2, \ldots, \mathbf{P}_n^C$ have been identified from the current data chunk, the current learning cycle enters the next stage.

Stage 2. Rule base initialization/updating. Given $\mathbf{P}_n^1, \mathbf{P}_n^2, \ldots, \mathbf{P}_n^C$ available, SOFIS+ will initialize C fuzzy IF-THEN rules in the form of Eq. (1) if $\mathbf{X}_n$ is the very first data chunk, namely, $n = 1$. Otherwise, SOFIS+ will use $\mathbf{P}_n^1, \mathbf{P}_n^2, \ldots, \mathbf{P}_n^C$ to update the existing rule base. First, SOFIS+ will identify prototypes from $\mathbf{P}_n^c$ that are distinctive from existing ones identified from historical data chunks and add them to $\mathbf{P}^c$ using Condition 2 [25,26]:

$$Cond.\,2: \quad \begin{array}{c} if\left(\min_{\boldsymbol{p}\in\mathbf{P}^c}\left(\left\|\boldsymbol{p}_{n,j}^c-\boldsymbol{p}\right\|_2\right) > \gamma_G^c\right) \\ then\left(\mathbf{P}^c \leftarrow \mathbf{P}^c \cup \left\{\boldsymbol{p}_{n,j}^c\right\}; \mathbf{P}_n^c \leftarrow \mathbf{P}_n^c / \left\{\boldsymbol{p}_{n,j}^c\right\}\right) \end{array} \tag{11}$$

where $c = 1, 2, \ldots, C$; $j = 1, 2, \ldots, P_n^c$.

After that, Condition 3 is used to identify these prototypes that contribute to building more precise classification boundaries from the remaining prototypes of $\mathbf{P}_n^c$ to join $\mathbf{P}^c$ [25,26]:

$$Cond.\,3: \quad \begin{array}{c} if\left(\min_{\boldsymbol{p}\in\mathbf{P}_n^i}\left(\left\|\boldsymbol{p}_{n,j}^c-\boldsymbol{p}\right\|_2\right) < 2\gamma_G^i \forall i \ne c\right) \\ then\left(\mathbf{P}^c \leftarrow \mathbf{P}^c \cup \left\{\boldsymbol{p}_{n,j}^c\right\}; \mathbf{P}_n^c \leftarrow \mathbf{P}_n^c / \left\{\boldsymbol{p}_{n,j}^c\right\}\right) \end{array} \tag{12}$$

The prototypes selected by Condition 3 are these ones that are spatially close to prototypes of other classes. By adding these prototypes into the knowledge base, SOFIS+ can refine its classification boundaries, thereby improving its classification accuracy.

Once $\mathbf{P}_n^1, \mathbf{P}_n^2, \ldots, \mathbf{P}_n^C$ have been merged into $\mathbf{P}^1, \mathbf{P}^2, \ldots, \mathbf{P}^C$ and the fuzzy rules $\mathbf{R}^1, \mathbf{R}^2, \ldots, \mathbf{R}^C$ are updated with the latest prototypes, the current learning cycle is completed and SOFIS+ enters the next learning cycle if a new data chunk $\mathbf{X}_{n+1}$ becomes available.

Algorithmic procedure of the learning process of SOFIS+ is summarized by Algorithm 1 [26].

Algorithm 1. Learning police of SOFIS+

while ($\mathbf{X}_n$ and $\mathbf{Y}_n$ are available) **do**:
  ### Stage 1 ###
  calculate σ_G using (4);
  for $c = 1$ to C **do**:
    calculate γ_G^c using (6);
    add $\boldsymbol{x}_{n,1}^c$ to $\mathbf{P}_n^c$;
    for $k = 2$ to L_n^c **do**:
      if (Condition 1 is satisfied) **do**
        add $\boldsymbol{x}_{n,k}^c$ to $\mathbf{P}_n^c$;
      else
        identify $\boldsymbol{p}_{n,*}^c$ from $\mathbf{P}_n^c$;
        update $\boldsymbol{p}_{n,*}^c$ using (10);
      end if
    end for
  end for
  ### Stage 2 ###
  for $c = 1$ to C **do**:
    if ($n = 1$) do:
      $\mathbf{P}^c \leftarrow \mathbf{P}_n^c$;
      initialize $\mathbf{R}^c$ with $\mathbf{P}^c$;
    else
      expand $\mathbf{P}^c$ with $\mathbf{P}_n^c$ using Conditions 2 and 3;
      update $\mathbf{R}^c$ with $\mathbf{P}^c$;
    end if
  end for
  $n \leftarrow n + 1$;
end while

2.2. SOFBIS

2.2.1. General architecture and decision making policy

Unlike SOFIS+, which constructs C fuzzy IF-THEN rules from data with each rule corresponding to one particular class, SOFBIS is composed of T fuzzy belief IF-THEN rules in the following form ($k = 1, 2, \ldots, T$) [27]:

$$\mathcal{R}_k: \quad \begin{array}{c} IF\ \left(\boldsymbol{x} \sim \boldsymbol{p}_{k,1}\right)\ AND\ \left(\boldsymbol{x} \sim \boldsymbol{p}_{k,2}\right)\ AND \ldots AND\ \left(\boldsymbol{x} \sim \boldsymbol{p}_{k,P_k}\right) \\ THEN\ \left\{\boldsymbol{D}, \boldsymbol{\beta}_k\right\} \end{array} \tag{13}$$

where P_k is the number of prototypes associated with $\mathcal{R}_k$; $\boldsymbol{D} = \left[D^1, D^2, \ldots, D^C\right]^T$; $\boldsymbol{\beta}_k = \left[\beta_k^1, \beta_k^2, \ldots, \beta_k^C\right]^T$ is the vector of belief degrees; β_k^c is the corresponding belief degree of the conclusion D^c.

Similar to fuzzy IF-THEN rules as given in Eq. (1), each fuzzy belief rule is also composed of a number of prototypes, but these prototypes are connected by logic "AND" connectives. This difference is caused by the unique decision-making scheme of SOFBIS that utilizes all the identified prototypes in decision-making. Another key difference between SOFBIS and SOFIS+ is that prototypes of SOFBIS are identified from data samples of all classes together and, due to the use of belief structure, each prototype can belong to multiple classes at the same time. Thanks to the two key improvements, SOFBIS can better handle the potential class overlaps during prototype identification, and achieve greater classification performance.

During decision-making, given a particular data sample $\boldsymbol{x}$, the activation of $\mathcal{R}_k$ is calculated as follows [27]:

$$\varphi_k\left(\boldsymbol{x}\right) = \frac{\vartheta_k\left(\boldsymbol{x}\right)}{\sum_{j=1}^{T} \vartheta_j\left(\boldsymbol{x}\right)} \tag{14}$$

where $\vartheta_k\left(\boldsymbol{x}\right) = \sum_{i=1}^{P_k} \vartheta_{k,i}\left(\boldsymbol{x}\right)$ denotes the activation produced by $\mathcal{R}_k$; $\vartheta_{k,i}\left(\boldsymbol{x}\right)$ is produced by Eq. (15) [27]:

$$\vartheta_{k,i}\left(\boldsymbol{x}\right) = \exp\left(-\frac{\left\|\boldsymbol{x} - \boldsymbol{p}_{k,i}\right\|_1^2}{\delta_G^2}\right) \tag{15}$$

and there is $\|\boldsymbol{x} - \boldsymbol{y}\|_1 = \sum_{i=1}^{M} |x_i - y_i|$; δ_G is a data-driven kernel width derived from data at the Gth level of granularity. Hence, the activation of $\mathcal{R}_k$ is calculated based on the L_1 distance between $\boldsymbol{x}$ and all the prototypes associated with $\mathcal{R}_k$.

Then, the outputs of the fuzzy belief rules are integrated together by Eq. (16):

$$\boldsymbol{\beta}\left(\boldsymbol{x}\right) = \sum_{k=1}^{T} \boldsymbol{\beta}_k \varphi_k\left(\boldsymbol{x}\right) \tag{16}$$

and the class label of $\boldsymbol{x}$ is determined using Eq. (17):

$$\hat{y} = c^*; \quad c^* = \underset{c=1,2,\ldots,C}{\operatorname{argmax}}\left(\beta^c\left(\boldsymbol{x}\right)\right) \tag{17}$$

2.2.2. Learning policy

The learning policy of SOFBIS is presented in this section. SOFBIS learns from data streams in a chunk-wise manner. A new learning cycle starts given a new data chunk available, and the data chunk is discarded after the learning cycle finishes.

Stage 1. Prototype identification. Given a new data chunk $\mathbf{X}_n$ and the corresponding class labels $\mathbf{Y}_n$, SOFBIS firstly calculates the L_1 distances between any two data samples of $\mathbf{X}_n$ and obtains a $L_n \times L_n$ dimensional distance matrix in the form of Eq. (18) [27]:

$$\mathbf{d}_n = \left[\left\|\boldsymbol{x}_{n,k} - \boldsymbol{x}_{n,j}\right\|_1^2\right]_{j=1:L_n}^{k=1:L_n} \tag{18}$$

The local kernel width $\delta_{n,G}$ is derived from data in a similar way as $\sigma_{n,G}$ (Eq. (5a)):

$$\delta_{n,G} = \sqrt{\frac{\sum_{j=1}^{L_n-1}\sum_{k=j+1}^{L_n} v_{g,j,k}\left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,k}\right\|_1^2}{\sum_{j=1}^{L_n-1}\sum_{k=j+1}^{L_n} v_{g,j,k}}} \tag{19}$$

where $v_{g,j,k} = \begin{cases} 1, & \left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,k}\right\|_1 \le \delta_{n,g-1} \\ 0, & else \end{cases}$; $\delta_{n,0} = \sqrt{\frac{2\sum_{j=1}^{L_n-1}\sum_{k=j+1}^{L_n}\left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,k}\right\|_1^2}{(L_n-1)L_n}}$. The data-driven kernel width δ_G is updated as: $\delta_G = \sqrt{\frac{\sum_{j=1}^{n} L_j \delta_{j,G}^2}{\sum_{j=1}^{n} L_j}}$. One can see from Eqs. (19) and (5a) that the key difference between $\delta_{n,G}$ and $\delta_{n,0}$ exists in the types of distance employed to measure the spatial dissimilarity between data samples.

Then, a sparse adjacency matrix is derived from $\mathbf{d}_n$ via Eq. (20) [27].

$$\mathbf{A}_n = \left[A_{n,k,j}\right]_{j=1:L_n}^{k=1:L_n} \tag{20}$$

where $A_{n,k,j} = \begin{cases} 1, & \left\|\boldsymbol{x}_{n,k} - \boldsymbol{x}_{n,j}\right\|_1 \le \delta_{n,G} \\ 0, & else \end{cases}$. Next, every data sample of the current chunk, $\boldsymbol{x}_{n,k}$ is treated as a micro-cluster and assigns membership degrees to neighbouring data samples including $\boldsymbol{x}_{n,k}$ itself using Eq. (21) ($k, j = 1, 2, \ldots, L_n$):

$$\mu_{n,k,j} = \begin{cases} \exp\left(-\frac{\left\|\boldsymbol{x}_{n,k} - \boldsymbol{x}_{n,j}\right\|_1^2}{\delta_{G,n}^2}\right), & A_{n,k,j} = 1 \\ 0, & else \end{cases} \tag{21}$$

The membership degrees each data sample receives from its neighbours are then aggregated together by Eq. (22):

$$\hat{\mu}_{n,k} = \sum_{j=1}^{L_n} \mu_{n,k,j} \tag{22}$$

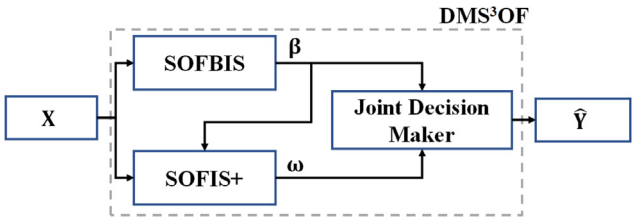


Fig. 1. Dual-model architecture of DMS³OF.

Condition 4 is utilized to identify these data samples that have the higher aggregated membership degrees than their neighbours as prototypes, denoted as $\mathbf{Q}_n$ [27]:

$$Cond.\,4: \quad \begin{array}{c} if\left(\hat{\mu}_{n,k} = \max\limits_{j=1,2,\ldots,L_n}\left(A_{n,k,j}\hat{\mu}_{n,j}\right)\right) \\ then\left(\mathbf{Q}_n \leftarrow \mathbf{Q}_n \cup \left\{\boldsymbol{x}_{n,k}\right\}\right) \end{array} \tag{23}$$

Once $\mathbf{Q}_n$ is identified from $\mathbf{X}_n$, these prototypes are updated as arithmetic means of all the data samples connected to them locally to better represent the local patterns of data ($\forall \boldsymbol{x}_{n,k} \in \mathbf{Q}_n$):

$$\hat{\boldsymbol{x}}_{n,k} = \frac{\sum_{j=1}^{L_n} A_{n,k,j}\boldsymbol{x}_{n,j}}{\sum_{j=1}^{L_n} A_{n,k,j}} \tag{24}$$

The support of $\hat{\boldsymbol{x}}_{n,k}$ is obtained as $S_{n,k} = \sum_{j=1}^{L_n} A_{n,k,j}$ and the belief degrees associated with $\hat{\boldsymbol{x}}_{n,k}$ are derived by Eq. (25) [27]:

$$\boldsymbol{\beta}_{n,k} = \frac{1}{S_{n,k}}\left[\sum_{j=1}^{L_n} A_{n,k,j}\mathbb{I}\left(y_{n,j} = c\right)\right]_{c=1:C}^{T} \tag{25}$$

After the updated prototypes, associated belief degrees and supports, denoted as $\mathbf{P}_n$, $\mathbf{B}_n$ and $\mathbf{S}_n$, have been obtained from $\mathbf{X}_n$ and $\mathbf{Y}_n$, SOFBIS enters the next stage of the learning cycle.

Stage 2. Rule base initialization/updating. Given the learned $\mathbf{P}_n$, $\mathbf{B}_n$ and $\mathbf{S}_n$ from the current data chunk, SOFBIS initializes its knowledge base first if there is $n = 1$:

$$\mathbf{P} \leftarrow \mathbf{P}_n; \quad \mathbf{B} \leftarrow \mathbf{B}_n; \quad \mathbf{S} \leftarrow \mathbf{S}_n \tag{26}$$

Otherwise, SOFBIS will use $\mathbf{P}_n$, $\mathbf{B}_n$ and $\mathbf{S}_n$ to update its knowledge base.

Condition 5 is used to identify these prototypes from $\mathbf{P}_n$ that represent novel data patterns unseen in historical chunks to expand the knowledge base of SOFBIS ($k = 1, 2, \ldots, P_n$; P_n is the cardinality of $\mathbf{P}_n$) [27]:

$$Cond.\,5: \quad \begin{array}{c} if\left(\min\limits_{\boldsymbol{p}\in\mathbf{P}}\left(\left\|\boldsymbol{p}_{n,k} - \boldsymbol{p}\right\|_1\right) > \delta_G\right) \\ then\left(\mathbf{P} \leftarrow \mathbf{P} \cup \left\{\boldsymbol{p}_{n,k}\right\}\right) \end{array} \tag{27}$$

Once $\boldsymbol{p}_{n,k}$ joins $\mathbf{P}$, it is removed from $\mathbf{P}_n$, and its associated belief degrees $\boldsymbol{\beta}_{n,k}$ and support $S_{n,k}$ also join $\mathbf{B}$ and $\mathbf{S}$ after being removed from $\mathbf{B}_n$ and $\mathbf{S}_n$.

After all the prototypes that satisfy Condition 5 have been added to $\mathbf{P}$, the remaining prototypes of $\mathbf{P}_n$ are utilized for updating the knowledge base using Eq. (28):

$$\begin{aligned} &\boldsymbol{p}_* \leftarrow \boldsymbol{p}_* + \frac{S_{n,k}\left(\boldsymbol{p}_{n,k} - \boldsymbol{p}_*\right)}{S_* + S_{n,k}}; \quad \boldsymbol{\beta}_* \leftarrow \boldsymbol{\beta}_* + \frac{S_{n,k}\left(\boldsymbol{\beta}_{n,k} - \boldsymbol{\beta}_*\right)}{S_* + S_{n,k}}; \\ &S_* \leftarrow S_* + S_{n,k} \end{aligned} \tag{28}$$

where $\boldsymbol{p}_* = \underset{\boldsymbol{p}\in\mathbf{P}}{\operatorname{argmin}}\left(\left\|\boldsymbol{p} - \boldsymbol{p}_{n,k}\right\|_1\right)$; $\boldsymbol{\beta}_*$ and S_* are the respective belief degrees and support of $\boldsymbol{p}_*$.

After $\mathbf{P}$, $\mathbf{B}$ and $\mathbf{S}$ have been initialized/updated with $\mathbf{P}_n$, $\mathbf{B}_n$ and $\mathbf{S}_n$, fuzzy belief rules are constructed in the form of Eq. (13), and then SOFBIS continues to process the next data chunk $\mathbf{X}_{n+1}$ by starting a new learning cycle. Algorithmic procedure of the learning process of SOFBIS is summarized by Algorithm 2 [27].

Algorithm 2. Learning police of SOFBIS

while ($\mathbf{X}_n$ and $\mathbf{Y}_n$ are available) **do**:
  ### Stage 1 ###
  calculate $\mathbf{d}_n$ from $\mathbf{X}_n$ using (18);
  calculate $\delta_{n,G}$ using (19);
  derive $\mathbf{A}_n$ from $\mathbf{d}_n$ using (20);
  calculate $\hat{\mu}_{n,k}$ ($k = 1,2\ldots, L_n$) using (22);
  for $k = 1$ to L_n **do**:
    if (Condition 4 is satisfied) **do**:
      obtain $\hat{\boldsymbol{x}}_{n,k}$ using (24);
      add $\hat{\boldsymbol{x}}_{n,k}$ to $\mathbf{P}_n$;
      obtain $\boldsymbol{\beta}_{n,k}$ and $S_{n,k}$;
      add $\boldsymbol{\beta}_{n,k}$ and $S_{n,k}$ to $\mathbf{B}_n$ and $\mathbf{S}_n$;
    end if
  end for
  ### Stage 2 ###
  if ($n = 1$) **do**:
    $\mathbf{P} \leftarrow \mathbf{P}_n$; $\mathbf{B} \leftarrow \mathbf{B}_n$; $\mathbf{S} \leftarrow \mathbf{S}_n$;
  else
    update $\mathbf{P}$, $\mathbf{B}$ and $\mathbf{S}$ with $\mathbf{P}_n$, $\mathbf{B}_n$ and $\mathbf{S}_n$;
  end if
end while

3. Proposed DMS³OF model

General architecture of the proposed DMS³OF model is depicted in Fig. 1. As shown by Fig. 1, DMS³OF is composed of the two EFS models, namely, SOFIS+ and SOFBIS, and a joint decision-maker that determines the class labels of unlabelled data samples based on predictions made by the two models. DMS³OF is designed to perform online semi-supervised learning from data streams on a chunk-by-chunk basis in a single pass manner, and is capable of continuously self-expanding its knowledge base with new prototypes identified from unlabelled data with only minimal involvement of human expert inputs.

As aforementioned, DMS³OF is designed for the highly challenging infinite delay problems [28], where only a small number of labelled samples are available at the beginning of the learning process for the classification model to warm up. Hence, the learning process of DMS³OF is divided into two stages. In the first stage, DMS³OF learns from labelled data chunks to prime its knowledge base in a supervised manner. In the second stage, DMS³OF utilizes unlabelled data chunks to self-expand its knowledge base via exploiting the pseudo labelling technique.

It is worth noting that, instead of updating both EFS models with unlabelled data, DMS³OF utilizes SOFBIS to produce pseudo labels and then augments SOFIS+ with the pseudo-labelled data. There are two important reasons for exploiting such semi-supervised learning scheme. First, thanks to its belief structure and unique decision-making scheme, SOFBIS can better handles class overlaps and can achieve greater prediction accuracy than SOFIS+, even with limited training samples available. Hence, pseudo labels produced by SOFBIS are more reliable and accurate than SOFIS+. Second, SOFIS+ is less sensitive towards pseudo-labelling errors and has stronger resistance to error propagation than SOFBIS. This is because SOFIS+ will not update prototypes learned from previous data chunks and only add prototypes that represent different patterns from existing ones to the

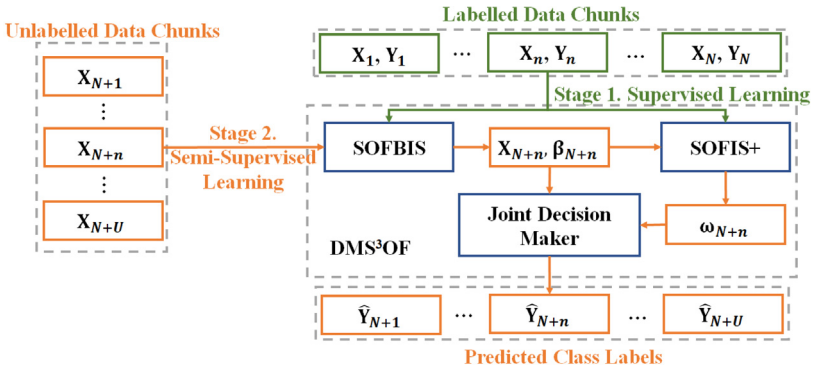


Fig. 2. Flowchart of the two-stage semi-supervised learning process of DMS^{3}OF.

knowledge base. In addition, SOFIS+ only uses the nearest prototype of each class to produce the confidence scores, whilst SOFBIS involves all prototypes to generate the belief degrees. As a result, pseudo-labelling errors propagate in SOFBIS more easily and would inevitably alter the classification boundaries unfavourably. In contrast, pseudo-labelling errors are confined locally by SOFIS+ and there is a much smaller chance that pseudo-labelling errors can influence the overall decision-making of SOFIS+. Therefore, by exploiting the aforementioned semi-supervised learning scheme, DMS^{3}OF can assign pseudo labels to unlabelled samples with greater accuracy whilst largely reduces the impact of pseudo-labelling errors on the overall prediction accuracy, thereby consistently self-improving itself from unlabelled data in a reliable way. Nevertheless, both EFS models will be updated if more labelled samples become available later during the learning process.

3.1. Learning policy

The algorithmic procedure of the semi-supervised learning process of DMS^{3}OF is detailed as follows. For better illustration, the semi-supervised learning process is visualized by the flowchart in Fig. 2. By default, the level of granularity for both EFS models is set the same as G. However, one may consider using different levels of granularity for the two models to achieve greater performance, but the optimal setting would vary from problem to problem depending on the nature of data.

Stage 1. Supervised learning. In this stage, the two sub-models of DMS^{3}OF, namely, SOFIS+ and SOFBIS are trained with the data chunks, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ and their corresponding class labels $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_N$ following the algorithmic procedures described by Algorithms 1 and 2, respectively. Once the knowledge base has been primed with labelled data, DMS^{3}OF proceeds to the next stage and starts to learn from unlabelled data chunks.

Stage 2. Semi-supervised learning via pseudo labelling. In this stage, DMS^{3}OF learns from unlabelled chunks to self-improve its knowledge base. Given a new data chunk available $\mathbf{X}_n (n > N)$, SOFBIS will produce the belief degrees to each sample of $\mathbf{X}_n$, denoted as $\boldsymbol{\beta}_n = \{\boldsymbol{\beta}(\boldsymbol{x}_{n,1}), \boldsymbol{\beta}(\boldsymbol{x}_{n,2}), \ldots, \boldsymbol{\beta}(\boldsymbol{x}_{n,L_n})\}$ using Eqs. (14)–(16). Then, SOFBIS will pass $\mathbf{X}_n$ and $\boldsymbol{\beta}_n$ to SOFIS+ for augmenting the knowledge base.

After SOFIS+ receives $\mathbf{X}_n$ and $\boldsymbol{\beta}_n$, it will first identify the highly confident predictions as pseudo-labelled samples via Condition 6 $(k = 1, 2, \ldots, K_n)$.

$$\textit{Cond.}\,6: \quad \begin{array}{c} \textit{if} \left(\max_{j=1,2,\ldots,C} \left(\boldsymbol{\beta}\left(\boldsymbol{x}_{n,k}\right)\right) \geq \gamma_0 \right) \\ \textit{then} \left(\hat{\mathbf{X}}_n \leftarrow \hat{\mathbf{X}}_n \cup \{\boldsymbol{x}_{n,k}\}; \hat{\mathbf{Y}}_n \leftarrow \hat{\mathbf{Y}}_n \cup \{\hat{y}_{n,k}\} \right) \end{array} \tag{29}$$

where γ_0 is a threshold to identify these samples that SOFBIS has strong belief towards their class labels; $\hat{y}_{n,k}$ is the predicted label of $\boldsymbol{x}_{n,k}$ derived from $\boldsymbol{\beta}(\boldsymbol{x}_{n,k})$ by Eq. (17); $\hat{\mathbf{X}}_n$ is the set of pseudo-labelled data samples derived from $\mathbf{X}_n$ with $\hat{\mathbf{Y}}_n$ being the corresponding pseudo labels. In this study, $\gamma_0 = 0.99$, which enables SOFIS+ to identify the most confident predictions made by SOFBIS. For all the data samples that satisfy Condition 6, they are added to $\hat{\mathbf{X}}_n$ and removed from $\mathbf{X}_n$. The pseudo labels of these samples are added to $\hat{\mathbf{Y}}_n$.

For the remaining samples of $\mathbf{X}_n$, SOFBIS is less confident on its predictions and the predicted labels it produces may be error-prone. Using these predicted labels by SOFBIS as pseudo labels of these samples to update SOFIS+ may introduce more pseudo-labelling errors to the knowledge base. Hence, before utilizing these samples for expanding the knowledge base, SOFIS+ needs to filter out these challenging samples whose class labels are difficult to predict based on the known data patterns learned from historical data. To do so, SOFIS+ will first predict the class labels of the remaining samples of $\mathbf{X}_n$ using the current IF-THEN rule base. However, instead of using only the nearest prototypes to produce the confidence scores, which is sensitive to outliers, SOFIS+ in DMS^{3}OF model uses multiple nearest prototypes of each class together to calculate the confidence score, thereby increasing the robustness of its predictions towards noisy samples. In this study, the nearest and the second nearest prototypes are considered and the confidence scores of SOFIS+ to $\boldsymbol{x}_{n,k}$ are produced by Eq. (30):

$$\omega_2^c\left(\boldsymbol{x}_{n,k}\right) = \frac{\lambda_2^c\left(\boldsymbol{x}_{n,k}\right)}{\sum_{i=1}^{C} \lambda_2^i\left(\boldsymbol{x}_{n,k}\right)} \tag{30}$$

where $\boldsymbol{x}_{n,k} \in \mathbf{X}_n$; $c = 1, 2, \ldots, C$; $\lambda_2^c(\boldsymbol{x}_{n,k}) = \exp(-\frac{\|\boldsymbol{x}_{n,k}-\boldsymbol{p}_{*1}^c\|_2^2 + \|\boldsymbol{x}_{n,k}-\boldsymbol{p}_{*2}^c\|_2^2}{2\sigma_G^2})$; $\boldsymbol{p}_{*1}^c = \underset{\boldsymbol{p}\in\mathbf{P}^c}{\operatorname{argmin}} \left(\|\boldsymbol{x}_{n,k} - \boldsymbol{p}\|_2\right)$ and $\boldsymbol{p}_{*2}^c = \underset{\boldsymbol{p}\in\mathbf{P}^c;\boldsymbol{p}\neq\boldsymbol{p}_{*1}^c}{\operatorname{argmin}} \left(\|\boldsymbol{x}_{n,k} - \boldsymbol{p}\|_2\right)$.

Based on the confidence scores, SOFIS+ utilizes Condition 7 to check whether the labels it produces on these samples are coincident to the pseudo-labels produced by SOFBIS.

$$\textit{Cond.}\,7: \quad \begin{array}{c} \textit{if} \left(\hat{y}_{n,k} = \underset{j=1,2,\ldots,C}{\operatorname{argmax}} \left(\omega_2^j\left(\boldsymbol{x}_{n,k}\right)\right)\right) \\ \textit{then} \left(\hat{\mathbf{X}}_n \leftarrow \hat{\mathbf{X}}_n \cup \{\boldsymbol{x}_{n,k}\}; \hat{\mathbf{Y}}_n \leftarrow \hat{\mathbf{Y}}_n \cup \{\hat{y}_{n,k}\} \right) \end{array} \tag{31}$$

If $\boldsymbol{x}_{n,k}$ satisfies Condition 7, it suggests that SOFIS+ agrees with SOFBIS on the predicted label of $\boldsymbol{x}_{n,k}$. In such case, $\boldsymbol{x}_{n,k}$ can be used for augmenting SOFIS+ with its pseudo label $\hat{y}_{n,k}$. Otherwise, it shows that SOFIS+ and SOFBIS are unable to agree on the class label of $\boldsymbol{x}_{n,k}$. As a result, $\boldsymbol{x}_{n,k}$ cannot be utilized for expanding the knowledge base and has to be discarded.

Although it is possible to utilize data samples that fail to satisfy Condition 6 or 7 for augmenting the model, DMS^{3}OF would avoid using these samples because the predicted pseudo labels are not

Table 1
Key information of benchmark problems for experimental study.

Dataset	#Samples	#Attributes	#Classes	Characteristics
AB	4177	8	3	Stationary
AU	690	14	2	Stationary
EG	10000	13	2	Stationary
GP	9901	17	5	Stationary
IS	2520	19	7	Stationary
LR	20000	16	26	Stationary
MF	2000	649	10	Stationary
OR	5620	64	10	Stationary
PB	5472	10	5	Stationary
PR	10992	16	10	Stationary
PW	11055	30	2	Stationary
SB	6321	9	2	Stationary
SE	2310	19	7	Stationary
SH	1593	256	10	Stationary
TE	5500	40	11	Stationary
WI	4839	5	2	Stationary
SEA	120000	3	2	Nonstationary
HY	25000	4	2	Nonstationary
PMN	70000	784	10	Nonstationary
RMN	62000	784	10	Nonstationary
SU	5000000	18	2	Nonstationary
PH	1025010	10	10	Nonstationary
Caltech101	8677	Roughly	101	Stationary
Caltech256	29780	$300 \times 200 \times 3$	256	Stationary

reliable enough and may introduce extra errors to SOFIS+. Without having true labels, pseudo-labelling errors are more likely to be accumulated in the system and cannot be corrected easily, which will inevitably decrease the overall prediction accuracy of DMS^{3}OF severely. However, as aforementioned, if the true class labels of these challenging samples become available later, these samples can be used for augmenting SOFIS+ and SOFBIS.

After $\hat{\mathbf{X}}_n$ and $\hat{\mathbf{Y}}_n$ have been extracted from $\mathbf{X}_n$ and $\boldsymbol{\beta}_n$, they will be used for improving SOFIS+ following the same algorithmic procedure described in Algorithm 1. Once SOFIS+ has been updated with the pseudo-labelled samples, the current learning cycle is completed and DMS^{3}OF starts to process the next data chunk if it is available (while setting $n \leftarrow n+1$).

The learning policy of DMS^{3}OF is summarized by Algorithm 3.

Algorithm 3. Learning police of DMS^{3}OF

```
### Stage 1 ###
while (X_n and Y_n are available) do:
  if (n = 1) do
    initialize SOFIS+ and SOFBIS with X_n and Y_n;
  else
    update SOFIS+ and SOFBIS with X_n and Y_n;
  end if
  n ← n + 1;
end while
### Stage 2 ###
while (X_n is available) do:
  use SOFBIS to produce β_n;
  extract X̂_n and Ŷ_n from X_n and β_n by Conditions 6 and 7;
  update SOFIS+ with X̂_n and Ŷ_n
  n ← n + 1;
end while
```

3.2. Decision-making policy

In the decision-making stage, for each unlabelled sample $\boldsymbol{x}$, DMS^{3}OF will determine its class label based on the confidence scores produced by SOFIS+ and the belief degrees produced by SOFBIS jointly. This allows DMS^{3}OF to combine the predictions made by both models, thereby incorporating the newly learned knowledge from unlabelled data into decision-making. The class label $\hat{y}$ of $\boldsymbol{x}$ is obtained by Eq. (32).

$$\hat{y} = c^*; \quad c^* = \underset{c=1,2,\ldots,C}{\operatorname{argmax}} \left(\frac{\omega_2^c(\boldsymbol{x}) + \beta^c(\boldsymbol{x})}{2} \right) \tag{32}$$

where $\omega_2^c(\boldsymbol{x})$ and $\beta^c(\boldsymbol{x})$ are obtained by Eqs. (16) and (30), respectively.

4. Experimental investigation

4.1. Experimental configuration

To evaluate the performance of the proposed DMS^{3}OF model, numerical examples based on a wide range of benchmark problems are presented. In this study, 24 benchmark problems (including 16 classical benchmarks, four synthetic large-scale ones, two very large-scale ones and two visual ones) from UCI Machine Learning Repository,[1] Keel Dataset Repository,[2] Scikit-Multiflow[3] and Caltech Vision Lab[4] are employed for experimental study, which include (1) Abalone (AB); (2) Australian (AU) (3) Electrical grid stability (EG); (4) Gesture phase segmentation (GP); (5) Image segmentation (IS); (6) Letter recognition (LR); (7) Multiple features (MF); (8) Optical recognition of handwritten digits (OR); (9) Page-blocks (PB); (10) Pen-based recognition of handwritten digits (PR); (11) Phishing websites (PW); (12) Shill bidding (SB); (13) Segment (SE); (14) Semeion handwritten digit (SH); (15) Texture (TE); (16) Wilt (WI); (17) SEA; (18) Hyperplane (HY) (19) Permuted MNIST (PMN); (20) Rotated MNIST (RMN); (21) SUSY (SU); (22) Poker Hand (PH); (23) Caltech101; and (24) Caltech256. Key details of the 24 datasets are summarized by Table 1.

In this study, for Caltech101 and Caltech256 datasets, pretrained AlexNet [29] and VGG-VD-16 [30] models are employed for feature extraction. Given a particular image $\mathbf{I}$, a 1×4096 dimensional activation vector will be extracted from the first fully connected layer of each model, and the two activation vectors extracted by AlexNet and VGG-VD-16 are then combined to a 8192×1 dimensional discriminative representation using

[1] https://archive.ics.uci.edu/ml/index.php
[2] https://sci2s.ugr.es/keel/datasets.php
[3] https://scikit-multiflow.github.io/
[4] http://www.vision.caltech.edu/datasets/

Table 2
Influence of G on DMS^{3}OF.

Dataset	Meas.	4	5	6	7	8	9	10	11	12
EG	*err*	0.358	0.306	0.237	0.222	0.215	**0.214**	**0.214**	0.218	0.223
	t_{exe}	0.605	0.643	0.766	0.956	1.179	1.314	1.368	1.465	1.464
GP	*err*	0.477	0.393	0.356	0.329	0.299	0.282	0.270	0.262	**0.259**
	t_{exe}	0.483	0.626	0.759	0.748	0.906	0.982	1.125	1.226	1.348
LR	*err*	0.705	0.493	0.304	0.175	0.136	**0.128**	0.131	0.136	0.140
	t_{exe}	0.850	1.240	2.055	3.021	3.559	3.761	3.883	3.970	4.165
PR	*err*	0.086	0.053	0.030	0.021	0.017	**0.016**	**0.016**	**0.016**	**0.016**
	t_{exe}	0.610	0.755	0.956	1.129	1.305	1.441	1.561	1.609	1.654
PW	*err*	0.104	0.098	0.092	0.083	**0.079**	0.082	0.090	0.089	0.089
	t_{exe}	0.904	1.012	1.160	1.231	1.408	1.607	1.642	1.634	1.682

Table 3
Influence of L on DMS^{3}OF.

Dataset	Meas.	10	50	100	200	500	1000	1500	2000	3000
EG	*err*	0.231	0.215	0.214	**0.213**	**0.213**	0.214	0.214	**0.213**	**0.213**
	t_{exe}	1.433	1.521	1.158	1.149	1.231	1.381	1.262	1.299	1.402
GP	*err*	0.381	0.283	0.280	**0.279**	0.281	0.282	0.281	0.281	0.280
	t_{exe}	2.115	1.247	1.106	1.023	0.9750	1.036	1.078	1.126	1.205
LR	*err*	0.343	0.128	**0.126**	0.127	0.129	0.128	0.128	0.130	0.131
	t_{exe}	18.418	7.076	4.778	3.517	3.079	3.872	4.025	4.229	4.299
PR	*err*	0.045	0.018	0.018	0.017	0.017	**0.016**	**0.016**	**0.016**	0.017
	t_{exe}	4.535	1.548	1.051	1.083	1.415	1.502	1.582	1.633	1.645
PW	*err*	**0.077**	0.081	0.082	0.081	0.081	0.082	0.083	0.083	0.083
	t_{exe}	2.228	1.404	1.231	1.304	1.509	1.673	1.772	1.827	1.886

Eq. (33). This representation, denoted as $\boldsymbol{x}$ will be used as the feature vector of the image for running numerical experiments.

$$\boldsymbol{x} = \left[\frac{\mathrm{AN}(\mathbf{I})}{\|\mathrm{AN}(\mathbf{I})\|}, \frac{\mathrm{VN}(\mathbf{I})}{\|\mathrm{VN}(\mathbf{I})\|} \right]^T \tag{33}$$

where AN ($\mathbf{I}$) and VN ($\mathbf{I}$) are the activation vectors extracted from AlexNet and VGG-VD-16 models, respectively.

The numerical examples presented in this paper are conducted on a desktop with dual core i7 processor 3.80 GHz $\times$ 2 and 32.0 GB RAM. The algorithms are developed using MATLABR2021b platform. For fair comparison, all the results reported in this section are obtained as the average of 10 Monte Carlo experiments to allow a certain degree of randomness.

Since this paper considers infinite delay problems, in running the experiments, labelled training samples are only presented to DMS^{3}OF for initialization during the warm-up stage. Then, DMS^{3}OF performs semi-supervised learning from unlabelled training samples to self-update its knowledge base and self-evolve its structure. To simulate the online learning environment, only one data chunk is presented to DMS^{3}OF at a time. For simplicity, all the data chunks are assumed to be of the same size, L. However, if the amount of remaining unprocessed labelled/unlabelled samples is less than L, all the remaining samples will be used as the final labelled/unlabelled data chunk. Unless specifically declared otherwise, by default, the level of granularity, G and the chunk size, L for DMS^{3}OF are set as $G = 9$ and $L = 1000$.

It will be demonstrated by the numerical examples presented in Section 4.4 that, with the default parameter setting, DMS^{3}OF outperforms the state-of-the-art methods on a variety of benchmark problems. However, in practice, users may use the default setting as the baseline to adjust the externally controlled parameter, G to achieve the desired results on the given problems with the specific requirements. The amount of data fed to DMS^{3}OF each time, namely, chunk size L can be adjusted as well. Experienced users may further consider setting the parameters of the two EFS models differently and assigning them a different set of weights in the final decision-making to maximize the performance of DMS^{3}OF. In so doing, one can expect a great improvement in the classification performance of DMS^{3}OF, but this would require some prior knowledge of the problem, and the best setting always varies according to the nature of data.

4.2. Sensitivity analysis

In this section, sensitivity analysis is performed to evaluate the influences of the two externally controlled parameters, G and L on the performance of DMS^{3}OF. The following five benchmark datasets, namely, EG, GP, LR, PR and PW are employed for sensitive analysis thanks to their relatively larger sizes, allowing a wider value range of the chunk size that can be chosen for running the experiments. In these experiments, 10% of data samples are randomly selected as labelled training set and the remaining 90% are used as the unlabelled one.

First, the influence of G on the prediction performance of the proposed model. During the experiments, the value of G varies from 4 to 12, and $L = 1000$. The results obtained by DMS^{3}OF at different levels of granularity are reported in Table 2 in terms of classification error (*err*) and training time consumption (t_{exe}, in seconds), where the lowest classification error per dataset is in bold.

Next, the influence of L on the prediction performance of the proposed model. During the experiments, the value of L varies from 10 to 3000, and $G = 9$. The results obtained by DMS^{3}OF with different chunk sizes are reported in Table 3 in terms of *err* and t_{exe} (in seconds), where the lowest classification error per dataset is in bold.

It can be observed from Tables 2 and 3 that both externally controlled parameters, namely, G and L have some mild influence on the prediction performance of DMS^{3}OF. In general, given a greater level of granularity, G, DMS^{3}OF is able to achieve better prediction accuracy with more prototypes being identified from data. However, a larger knowledge base can also reduce the computational efficiency, leading to more resources consumed for training. A greater G also increases the risk of overfitting. Whilst the chunk size, L has less impact on DMS^{3}OF as long as its value is greater enough (i.e., $L \geq 50$).

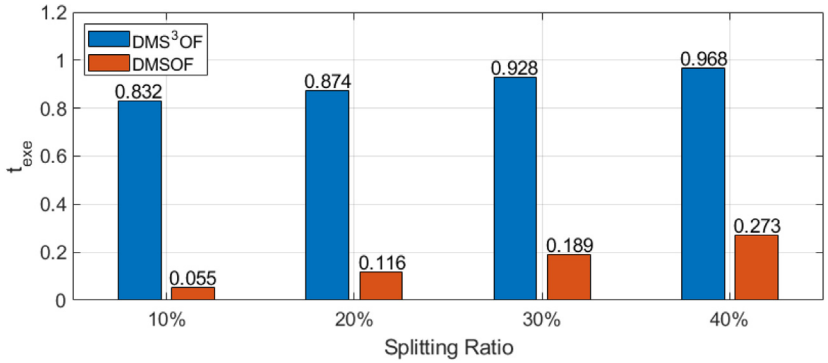


Fig. 3. Ablation analysis results in terms of t_{exe} (in seconds).

Table 4
Ablation analysis results.

Dataset	Ratio	DMS^{3}OF	DMSOF	Δ %	Ratio	DMS^{3}OF	DMSOF	Δ %
AB		0.480	0.483	+0.6%		0.481	0.479	−0.4%
AU		0.414	0.420	+1.5%		0.387	0.391	+1.0%
EG		0.214	0.216	+0.9%		0.201	0.201	+0.0%
GP		0.282	0.289	+2.5%		0.218	0.224	+2.8%
IS		0.097	0.114	+17.5%		0.070	0.076	+8.6%
LR		0.128	0.144	+12.5%		0.080	0.090	+12.5%
MF		0.061	0.079	+29.5%		0.052	0.061	+17.3%
OR		0.026	0.032	+23.1%		0.018	0.021	+16.7%
PB	10%	0.076	0.074	−2.6%	20%	0.066	0.066	+0.0%
PR		0.016	0.020	+25.0%		0.012	0.014	+16.7%
PW		0.082	0.089	+8.5%		0.067	0.070	+4.5%
SB		0.012	0.012	+0.0%		0.006	0.006	+0.0%
SE		0.101	0.121	+19.8%		0.075	0.082	+9.3%
SH		0.179	0.186	+3.9%		0.136	0.139	+2.2%
TE		0.037	0.046	+24.2%		0.026	0.031	+19.3%
WI		0.035	0.036	+2.9%		0.026	0.027	+3.9%
Avg.		**0.140**	**0.148**	**+5.7%**		**0.120**	**0.124**	**+3.3%**
AB		0.478	0.477	−0.2%		0.473	0.474	+0.2%
AU		0.374	0.376	+0.5%		0.370	0.379	+2.4%
EG		0.199	0.196	−1.51%		0.197	0.193	−2.0%
GP		0.187	0.191	+2.1%		0.168	0.172	+2.4%
IS		0.057	0.060	+5.3%		0.049	0.051	+4.1%
LR		0.062	0.068	+9.7%		0.051	0.055	+7.8%
MF		0.046	0.050	+8.7%		0.041	0.044	+7.3%
OR		0.015	0.017	+13.3%		0.014	0.016	+14.3%
PB	30%	0.063	0.065	+3.2%	40%	0.059	0.061	+3.4%
PR		0.009	0.010	+11.1%		0.008	0.008	+0.0%
PW		0.054	0.057	+5.6%		0.048	0.050	+4.2%
SB		0.005	0.005	+0.0%		0.003	0.004	+33.3%
SE		0.062	0.067	+8.1%		0.049	0.052	+6.1%
SH		0.114	0.116	+1.8%		0.096	0.099	+3.1%
TE		0.020	0.024	+20.0%		0.017	0.018	+5.9%
WI		0.023	0.023	+0.0%		0.021	0.022	+4.8%
Avg.		**0.111**	**0.113**	**+1.8%**		**0.104**	**0.106**	**+1.9%**

4.3. Ablation analysis

In this section, ablation analysis is performed to demonstrate the effectiveness of the proposed semi-supervised learning scheme designed for DMS^{3}OF. In this analysis, the performance of DMS^{3}OF is compared against a supervised learning model with the same architecture as DMS^{3}OF but without the semi-supervised learning mechanism. Hence, this new competitive model is named as duel-model self-organizing fuzzy inference system (DMSOF). In this example, the aforementioned 16 classical benchmark datasets are employed for performance evaluation, where the ratio of data being used as labelled training samples varies from 10%, 20%, 30% and 40% (and the remaining samples are used as unlabelled ones). The performances of DMS^{3}OF and DMSOF obtained from the numerical experiments are reported by Table 4 in terms of *err*. The performance improvements in percentage are given by Table 4 following the common practice [9]. The average results are tabulated in Table 4 as well for clarity. The average training time consumptions (in seconds) of DMS^{3}OF and DMSOF over the 16 benchmark datasets under different split ratios are depicted in Fig. 3.

It can be observed from Table 4 that DMS^{3}OF outperforms DMSOF in the vast majority of cases, especially when the ratio of labelled training samples is low (e.g., 10%). In particular, the average classification performance of DMS^{3}OF over the 16 benchmark datasets is improved by 5.7%, 3.3%, 1.8% and 1.9% with the training-testing split ratio set as 1:9 (10%), 2:8 (20%), 3:7 (30%) and 4:6 (40%), respectively. This demonstrates the effectiveness of the proposed DMS^{3}OF model. In addition, one can see from Fig. 3 that, the computational complexity of DMS^{3}OF is also very low. Its computational efficiency is on the same level as its supervised competitor, namely, DMSOF. It takes less than 0.15 ms for DMS^{3}OF to process one data sample whilst DMSOF takes approximately 0.1 ms for each sample.

Table 5
Performance comparison between DMS^{3}OF and 12 state-of-the-art semi-supervised learning methods under the training-testing split ratio of 1:9.

Algorithm	AB	AU	EG	GP	IS	LR	MF	OR
DMS^{3}OF	0.480	0.414	0.214	**0.282**	0.097	**0.128**	0.061	0.026
TMPM	0.469	**0.158**	0.020	0.532	0.166	0.482	0.078	0.090
LGC	**0.463**	0.180	0.361	0.286	0.108	0.142	**0.021**	0.034
GGMC	0.521	0.319	0.122	0.526	0.198	0.423	0.023	0.053
EAGR	0.501	0.429	0.341	0.345	0.168	0.154	0.126	0.027
AGRK	0.498	0.430	0.343	0.372	0.170	0.173	0.143	0.027
AGRL	0.477	0.401	0.269	0.363	0.173	0.144	0.115	**0.020**
LSVM	0.471	0.427	0.285	0.380	0.164	0.170	0.102	0.023
STHP	0.507	0.402	0.278	0.400	0.196	0.139	0.101	0.024
S^3 OFIS	0.499	0.398	0.262	0.285	0.127	0.961	0.108	0.029
TrikNN	0.481	0.404	0.241	0.353	0.178	0.208	0.145	0.035
TriDT	0.501	0.184	**0.001**	0.350	**0.086**	0.302	0.121	0.200
TriMLP	0.477	0.180	0.088	0.506	0.233	0.955	0.072	0.182
Algorithm	PB	PR	PW	SB	SE	SH	TE	WI
DMS^{3}OF	0.076	**0.016**	0.082	**0.012**	**0.101**	0.179	0.037	0.035
TMPM	0.108	0.118	**0.080**	0.027	0.193	0.218	**0.027**	0.071
LGC	0.052	0.022	0.127	0.021	0.132	0.181	0.029	0.054
GGMC	0.656	0.068	0.283	0.024	0.210	0.220	0.034	0.472
EAGR	0.326	0.017	0.118	0.014	0.188	0.214	0.039	0.171
AGRK	0.291	0.019	0.116	0.021	0.196	0.220	0.045	0.190
AGRL	0.371	0.017	0.104	0.068	0.180	**0.161**	0.038	0.282
LSVM	0.094	0.026	0.132	0.007	0.174	0.165	0.117	0.054
STHP	0.091	0.021	0.108	0.016	0.212	0.163	0.035	0.045
S^3 OFIS	0.075	0.019	0.643	0.013	0.137	0.209	0.043	0.035
TrikNN	0.067	0.032	0.103	0.021	0.183	0.207	0.048	0.040
TriDT	**0.049**	0.109	0.082	0.013	0.108	0.433	0.155	**0.033**
TriMLP	0.087	0.398	0.086	0.028	0.244	0.338	0.390	0.056

4.4. Performance demonstration

In this section, the performance of DMS^{3}OF is compared with a number of state-of-the-art approaches on the aforementioned 24 classification problems.

First, the classification performance (in *err*) of DMS^{3}OF is compared with the following nine popular single-model semi-supervised classification methods on the 16 classical benchmark problems with the split ratio set as 1:9, namely, 10% of data samples are used as labelled training samples and the remaining ones are used as unlabelled ones.

(1) Transductive minimax probability machine (TMPM) [6];
(2) LGC classifier [16];
(3) GGMC classifier [18];
(4) EAGR classifier [11];
(5) AGR classifier with kernel weights (AGRK) [19];
(6) AGR classifier with local anchor embedding weights (AGRL) [19];
(7) LSVM [17];
(8) S^3 OFIS [25], and;
(9) STHP classifier [20].

In running the experiments, the values of two externally controlled parameters, λ and ρ of TMPM are selected from the candidate set $[10^{-4}, 10^{-3}, \ldots, 10^{4}]$ as recommended by [6], and 10% of the labelled training samples are used as the validation set to help the model determine the best parameter setting for each problem; the parameters of LGC and GGMC are set as $\alpha = 0.99$ and $\mu = 0.01$, respectively, as suggested by [16,18]; and both LGC and GGMC use the kNN graph with $k = 5$; EAGR, AGRK and AGRL identify a total of $0.1K$ anchors from data (K is the total number of data samples); the number of the closest anchors s is set as $s = 3$; the iteration number of local anchor embedding is set to be 10 for AGRL [19]; for LSVM, three different parameter settings are considered, namely, (i) $\sigma = 10, \mu_I = 1, \mu_A = 10^{-6}, k = 15$ (recommended by [31]), (ii) $\sigma = 10, \mu_I = 0.5, \mu_A = 10^{-6}, k = 15$ and (iii) $\sigma = 1, \mu_I = 1, \mu_A = 10^{-6}, k = 15$ and the best result (the lowest *err*) on each dataset is reported; STHP uses the following parameter setting: $\gamma_0 = 1.1$ and $H = 6$ [20]; the parameters of S^3 OFIS are set as $G = 10$ [25]; the chunk size for STHP and S^3 OFIS is set as $L = 1000$. It is worth noting that different from the other seven competitive methods, both STHP and S^3 OFIS are designed for semi-supervised learning from data streams on a chunk-by-chunk basis. Hence, STHP and S^3 OFIS follow the same online learning setting as DMS^{3}OF.

In addition to the nine single-model competitors, one of the best known multi-model semi-supervised framework, tri-training [9] is also involved in performance comparison on the 16 benchmark problems. In this example, three different main stream classifiers, namely, kNN, DT and MLP are employed as the base classifiers, resulting in three different tri-training models, denoted by TrikNN, TriDT and TriMLP. In the experiments, k is set to be 5 for kNN; the maximum number of split for DT is set as $K - 1$; MLP has a three-layer structure and there are 128 neurons in its hidden layer.

The performances (in *err*) of DMS^{3}OF and 12 competitors on the unlabelled data samples of each problem are reported by Table 5, where the best results (lowest *err* per dataset) are in bold. The same experiments are repeated under the split ratios of 2:8, 3:7 and 4:6, and the results obtained by DMS^{3}OF and 12 competitors per dataset per split ratio are tabulated in Tables 6–8.

For visual clarity, the average performance ranks of DMS^{3}OF and the 12 competitors over the 16 benchmark problems under the four different training-testing split ratios are visualized in Fig. 4. Note that, the ranking is conducted per dataset per split ratio based on the *err* rates of the methods involved in performance comparison, where the method achieving the lowest *err* is ranked the first place (1st) and the method reporting the highest *err* is ranked the last place (13th). Hence, under each split ratio, DMS^{3}OF and the 12 competitors are ranked 16 times in total, and the average ranks are given by the figure. The average *err* rates of the semi-supervised learning methods involved in this example per split ratio are presented in Fig. 5, and the corresponding average training time costs (in seconds) are also presented in Fig. 6.

It can be seen from Tables 5–8, Figs. 4 and 5 that DMS^{3}OF is able to produce highly accurate classification results on the

Table 6
Performance comparison between DMS^{3}OF and 12 state-of-the-art semi-supervised learning methods under the training-testing split ratio of 2:8.

Algorithm	AB	AU	EG	GP	IS	LR	MF	OR
DMS^{3}OF	0.481	0.387	0.201	0.218	**0.070**	**0.080**	0.052	0.018
TMPM	**0.463**	**0.141**	0.014	0.510	0.148	0.462	0.045	0.072
LGC	**0.463**	0.179	0.362	0.253	0.097	0.119	**0.021**	0.030
GGMC	0.493	0.230	0.114	0.432	0.146	0.315	0.024	0.037
EAGR	0.497	0.387	0.317	0.299	0.147	0.118	0.108	0.022
AGRK	0.495	0.366	0.319	0.336	0.153	0.137	0.130	0.023
AGRL	0.479	0.356	0.260	0.325	0.150	0.107	0.102	**0.017**
LSVM	0.465	0.406	0.226	0.348	0.117	0.110	0.092	0.019
STHP	0.511	0.390	0.273	0.363	0.143	0.091	0.078	0.019
S^3 OFIS	0.503	0.387	0.260	**0.204**	0.085	0.961	0.086	0.020
TrikNN	0.480	0.356	0.233	0.284	0.131	0.129	0.111	0.025
TriDT	0.502	0.182	**0.000**	0.277	**0.070**	0.238	0.086	0.145
TriMLP	0.471	0.159	0.094	0.509	0.235	0.952	0.054	0.171
Algorithm	PB	PR	PW	SB	SE	SH	TE	WI
DMS^{3}OF	0.066	**0.012**	**0.067**	0.006	**0.075**	0.136	0.026	**0.026**
TMPM	0.119	0.111	0.079	0.027	0.139	0.166	**0.020**	0.062
LGC	0.048	0.022	0.109	0.019	0.102	0.161	0.027	0.054
GGMC	0.615	0.063	0.222	0.020	0.152	0.200	0.031	0.464
EAGR	0.317	0.013	0.096	0.010	0.153	0.165	0.034	0.173
AGRK	0.286	0.014	0.096	0.015	0.155	0.171	0.039	0.193
AGRL	0.306	0.014	0.090	0.044	0.158	0.142	0.034	0.251
LSVM	0.089	0.020	0.109	0.004	0.132	0.146	0.088	0.054
STHP	0.095	0.015	0.085	0.011	0.155	**0.132**	0.024	0.045
S^3 OFIS	0.071	0.014	0.818	**0.005**	0.094	0.159	0.030	0.027
TrikNN	0.058	0.020	0.091	0.009	0.133	0.182	0.040	0.033
TriDT	**0.042**	0.078	0.068	0.006	**0.075**	0.370	0.123	**0.026**
TriMLP	0.091	0.417	0.080	0.028	0.229	0.280	0.392	0.057

Table 7
Performance comparison between DMS^{3}OF and 12 state-of-the-art semi-supervised learning methods under the training-testing split ratio of 3:7.

Algorithm	AB	AU	EG	GP	IS	LR	MF	OR
DMS^{3}OF	0.478	0.374	0.199	0.187	**0.057**	**0.062**	0.046	**0.015**
TMPM	**0.457**	**0.143**	0.013	0.531	0.122	0.462	0.042	0.063
LGC	0.463	0.183	0.362	0.244	0.091	0.114	**0.020**	0.029
GGMC	0.491	0.175	0.108	0.378	0.126	0.248	0.022	0.036
EAGR	0.488	0.366	0.296	0.273	0.125	0.098	0.087	0.019
AGRK	0.487	0.363	0.299	0.314	0.132	0.115	0.106	0.020
AGRL	0.472	0.371	0.254	0.302	0.133	0.094	0.087	0.016
LSVM	0.467	0.379	0.211	0.329	0.103	0.090	0.092	0.018
STHP	0.504	0.391	0.266	0.343	0.115	0.074	0.065	0.017
S^3 OFIS	0.504	0.389	0.258	**0.167**	0.062	0.961	0.069	0.019
TrikNN	0.484	0.355	0.228	0.245	0.102	0.097	0.085	0.018
TriDT	0.498	0.178	**0.000**	0.232	0.060	0.215	0.074	0.138
TriMLP	0.473	0.162	0.087	0.499	0.233	0.952	0.049	0.163
Algorithm	PB	PR	PW	SB	SE	SH	TE	WI
DMS^{3}OF	0.063	**0.009**	**0.054**	0.005	**0.062**	**0.114**	0.020	**0.023**
TMPM	0.130	0.107	0.078	0.027	0.125	0.143	**0.015**	0.056
LGC	0.046	0.018	0.099	0.019	0.091	0.165	0.026	0.054
GGMC	0.633	0.030	0.173	0.020	0.133	0.198	0.029	0.443
EAGR	0.325	0.011	0.083	0.008	0.137	0.142	0.028	0.167
AGRK	0.289	0.012	0.085	0.013	0.141	0.148	0.032	0.185
AGRL	0.282	0.011	0.084	0.033	0.145	0.140	0.032	0.226
LSVM	0.086	0.015	0.099	**0.004**	0.118	0.143	0.072	0.054
STHP	0.089	0.011	0.070	0.009	0.119	**0.114**	0.019	0.041
S^3 OFIS	0.063	0.010	0.083	**0.004**	0.070	0.141	0.023	0.024
TrikNN	0.054	0.014	0.081	0.006	0.104	0.150	0.029	0.028
TriDT	**0.039**	0.066	0.062	0.005	**0.062**	0.329	0.104	0.024
TriMLP	0.092	0.400	0.081	0.027	0.226	0.245	0.408	0.055

16 benchmark problems under the four different training-testing split ratios, ranked the top place among the semi-supervised learning methods involved in the experiments. It is interesting to notice from Fig. 4 that the performance of DMS^{3}OF keeps improving with more labelled training data being given at the training phase as its average performance rank increases with a higher ratio of labelled training samples. The average error rates of DMS^{3}OF on are also lower than the 12 state-of-the-art competitors under the split ratios of 2:8, 3:7 and 4:6, and is only 0.002 higher than LGC under the split ratio of 1:9. In addition, one can see from Fig. 5 that the training time consumption of DMS^{3}OF is also on the same level as the most computationally efficient competitors, e.g., EAGR, AGRL, LSVM, S^3 OFIS, TrikNN and TriDT. This example demonstrates the superior performance of the proposed DMS^{3}OF.

Next, the classification performance of DMS^{3}OF is tested on four synthetic large-scale benchmark problems, namely, SEA, HY, PMN and RMN, and compared with the state-of-the-art results reported in the literature. Following the common practice, 1%, 0.8%, 1.4% and 2% of data in the respective four datasets are randomly

Table 8
Performance comparison between DMS^{3}OF and 12 state-of-the-art semi-supervised learning methods under the training-testing split ratio of 4:6.

Algorithm	AB	AU	EG	GP	IS	LR	MF	OR
DMS^{3}OF	0.473	0.370	0.197	0.168	**0.049**	**0.051**	0.041	**0.014**
TMPM	0.463	**0.144**	0.012	0.530	0.119	0.449	0.030	0.058
LGC	**0.462**	0.184	0.362	0.242	0.086	0.107	**0.020**	0.029
GGMC	0.484	0.163	0.105	0.353	0.105	0.194	0.022	0.033
EAGR	0.481	0.363	0.283	0.256	0.118	0.087	0.083	0.017
AGRK	0.483	0.346	0.286	0.298	0.122	0.104	0.098	0.018
AGRL	0.468	0.358	0.252	0.288	0.123	0.086	0.085	0.016
LSVM	0.465	0.362	0.200	0.323	0.099	0.099	0.102	0.020
STHP	0.494	0.397	0.258	0.325	0.095	0.062	0.058	0.015
S^3 OFIS	0.502	0.374	0.256	**0.140**	0.052	0.961	0.063	0.016
TrikNN	0.476	0.356	0.225	0.219	0.083	0.078	0.074	0.017
TriDT	0.499	0.166	**0.000**	0.203	0.050	0.191	0.067	0.126
TriMLP	0.481	0.158	0.087	0.502	0.234	0.957	0.043	0.160
Algorithm	PB	PR	PW	SB	SE	SH	TE	WI
DMS^{3}OF	0.059	**0.008**	**0.048**	**0.003**	**0.049**	**0.096**	0.017	**0.021**
TMPM	0.133	0.107	0.077	0.029	0.119	0.135	**0.011**	0.057
LGC	0.047	0.017	0.099	0.019	0.093	0.158	0.025	0.054
GGMC	0.656	0.016	0.149	0.020	0.119	0.180	0.026	0.442
EAGR	0.314	0.010	0.076	0.008	0.129	0.133	0.022	0.161
AGRK	0.295	0.011	0.078	0.014	0.131	0.139	0.026	0.175
AGRL	0.278	0.010	0.077	0.025	0.138	0.129	0.028	0.210
LSVM	0.084	0.012	0.095	0.004	0.122	0.130	0.059	0.054
STHP	0.096	0.010	0.061	0.008	0.110	0.102	0.017	0.045
S^3 OFIS	0.056	**0.008**	0.073	**0.003**	0.056	0.125	0.018	0.023
TrikNN	0.051	0.011	0.073	0.004	0.092	0.134	0.024	0.025
TriDT	**0.036**	0.057	0.057	0.005	0.053	0.319	0.099	0.023
TriMLP	0.088	0.398	0.081	0.026	0.216	0.223	0.386	0.056

Table 9
Classification performance comparison on four synthetic large-scale problems.

Algorithm	SEA	HY	PMN	RMN
DMS^{3}OF	**0.023**	**0.088**	0.106	**0.096**
SCARGCKNN	0.219	0.218	0.667	0.762
SCARGCSVM	0.174	0.181	0.670	0.791
ParsNet	0.120	0.132	0.537	0.515
DEVDAN	0.215	0.344	0.633	0.710
S^{3}OFIS+	0.040	0.092	**0.104**	0.098

Table 10
Classification performance comparison on two very large-scale problems.

Algorithm	SU	PH
DMS^{3}OF	**0.217**	**0.425**
WeScatterNet	0.243	0.499
ScatterNet	0.246	0.499
SPANFIS	0.243	0.500

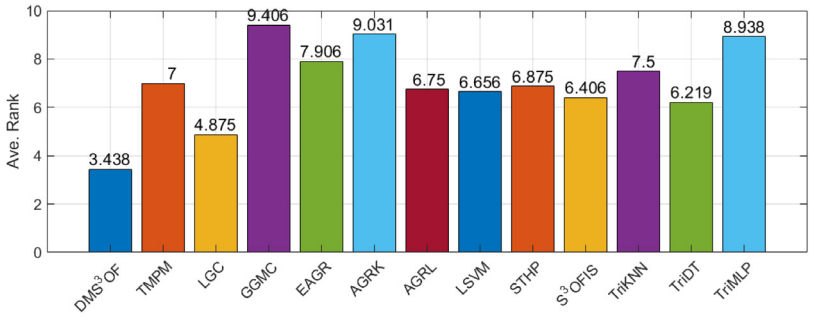
selected to build the labelled training set and the remaining data samples are used as unlabelled training samples [21]. The classification performance (in *err*) of DMS^{3}OF on the four large-scale problems are reported in Table 9. In addition, its performance on the four problems is further compared with the following state-of-the-art approaches under the same experimental protocols: (1) stream classification algorithm guided by clustering with kNN (SCARGCKNN) [32]; (2) stream classification algorithm guided by clustering with SVM (SCARGCSVM) [32]; (3) ParsNet [21]; (4) deep evolving denoising autoencoder (DEVDAN) [33] and (5) S^3 OFIS [25]. The classification results by the five comparative approaches are also tabulated in Table 9, where the best results are in bold. Note that, the results by SCARGCKNN, SCARGCSVM, ParsNet and DEVDAN are obtained directly from [21] and the results by S^3 OFIS is obtained from [25].

Then, the classification performance of DMS^{3}OF is evaluated on two very large-scale datasets (SU and PH). Following the common practice [24], the training-testing split ratio for both datasets is set as 1:4, namely, 25% of the data are randomly selected as labelled training samples and the rest are used as unlabelled ones. The performance of DMS^{3}OF in terms of accumulated one-chunk ahead prediction error is reported in Table 10 and compared with the following three approaches designed for very large-scale classification problems, namely, (1) WeScatterNet [24]; (2) Scalable teacher-forcing network [34] and (3) Scalable parsimonious network based on fuzzy inference system (SPANFIS) [35]. The results by WeScatterNet, ScatterNet and SPANFIS presented in Table 10 are the best results reported [24] obtained with different parameter settings under the same experimental protocol.

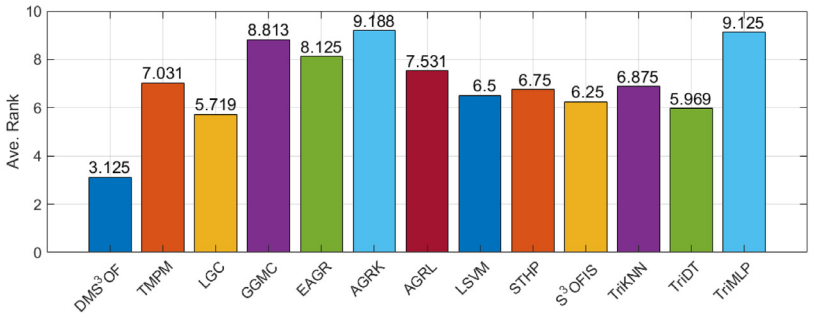
One can see from Tables 9 and 10 that the proposed DMS^{3}OF outperforms the state-of-the-art approaches on five out of the six large-scale classification problems (including two very large ones), demonstrating its strong potential as a semi-supervised learning method for large-scale data stream learning.

Finally, DMS^{3}OF is tested on two well-known image classification problems, namely, Caltech101 and Caltech256. Following the common practice [36], for Caltech101, 15 and 30 images are randomly selected from each class respectively to build the labelled training set and the remaining images are used as the unlabelled training images. For Caltech256, 15, 30,46 and 60 images are randomly selected from each class respectively as the labelled training images, and the rest are used as unlabelled ones. The classification performance (in *err*) of DMS^{3}OF on Caltech101 and Caltech256 are reported in Tables 11 and 12, and the selected state-of-the-art results obtained from the literature are also given for a better evaluation.

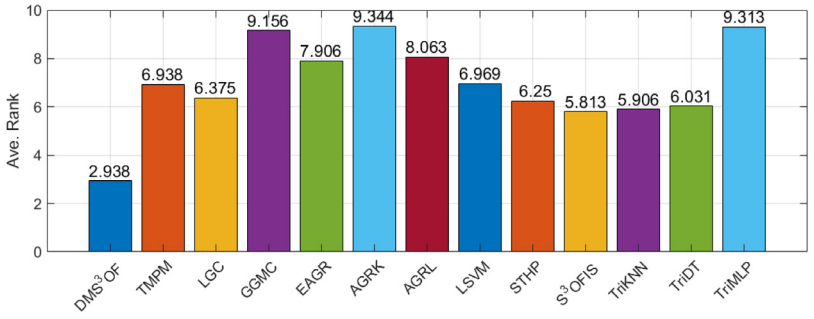
It is shown by Tables 11 and 12 that DMS^{3}OF is able to predict class labels on the unlabelled images with great accuracy on both challenging image classification problems. Despite that both datasets have a large number of classes, which makes pseudo-labelling a particularly difficult task, and only the pretrained AlexNet and VGG-VD-16 models are employed for feature extraction without fine tuning, the classification accuracy of DMS^{3}OF surpasses the most of the state-of-the-art approaches involved



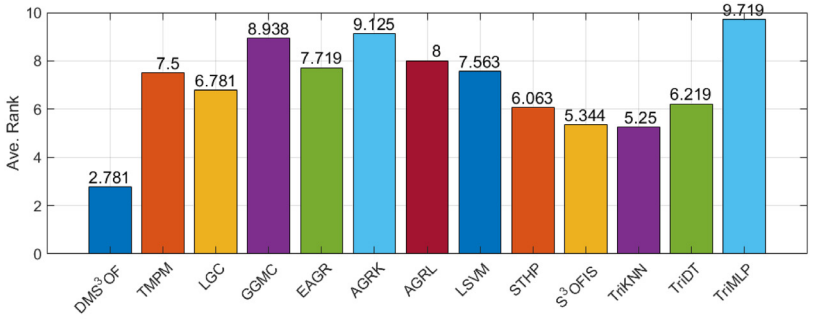
(a) Average performance rank under the split ratio of 1:9



(b) Average performance rank under the split ratio of 2:8



(c) Average performance rank under the split ratio of 3:7

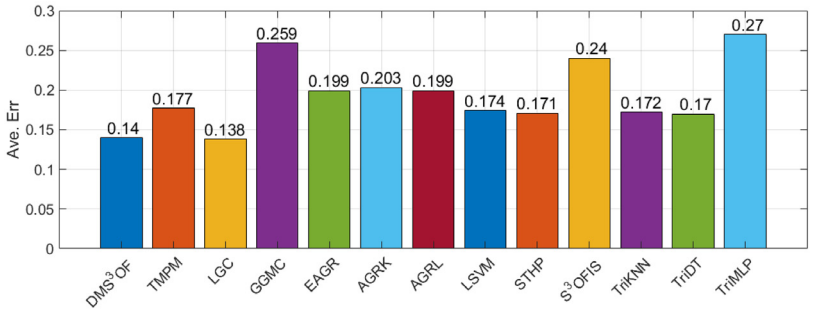


(d) Average performance rank under the split ratio of 4:6

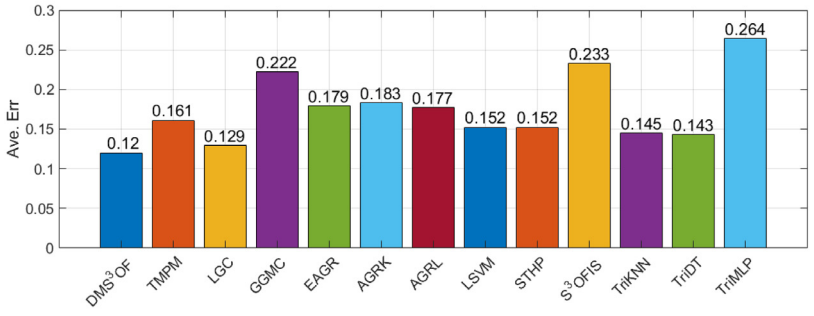
Fig. 4. Average performance rank comparison between DMS^{3}OF and 12 competitors (the lower the better).

in comparison, only being outperformed by SWSS-VGG [43] on Caltech256.

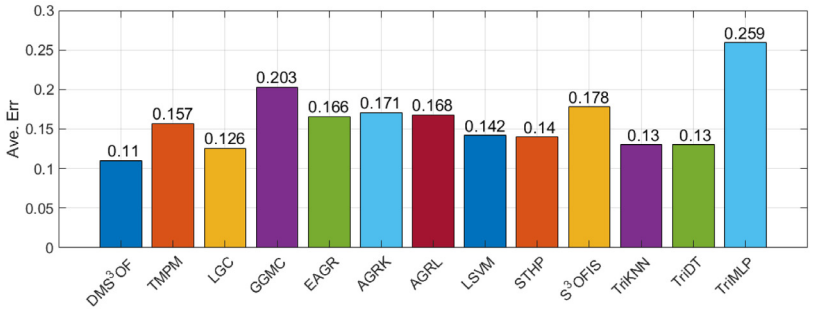
From the above systematic experimental studies (conducted over a variety of benchmark and real-world problems), one can conclude that the proposed DMS^{3}OF model is superior to the state-of-the-art approaches for semi-supervised classification from data streams in infinite delay environments, by offering greater classification accuracy. In addition, it is highly computationally efficient, taking less than 0.15 ms to process one data sample as suggested by Fig. 3. Fig. 6 also shows that the computational efficiency of DMS^{3}OF is at a high level, comparable to the most efficient alternative methods involved in the performance comparison. Very importantly, all the experiments carried out and numerical results reported in this section are obtained using



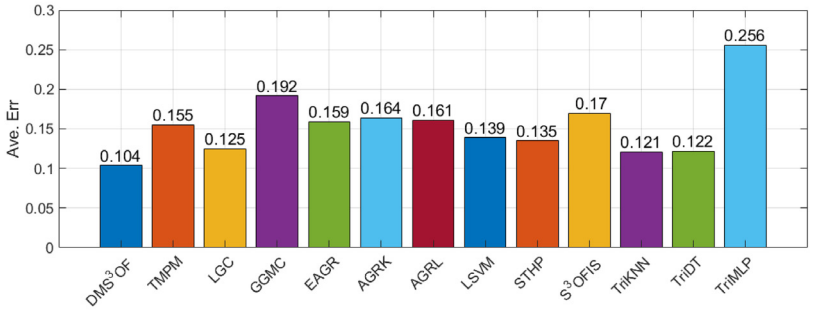
(a) Average classification error under the split ratio of 1:9



(b) Average classification error under the split ratio of 2:8



(c) Average classification error under the split ratio of 3:7



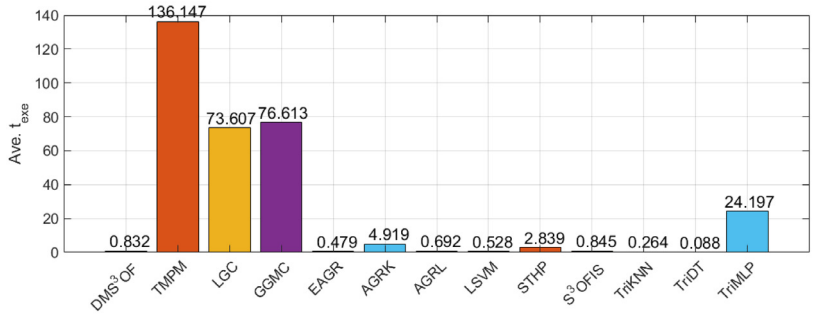
(d) Average classification error under the split ratio of 4:6

Fig. 5. Average classification error comparison between DMS^{3}OF and 12 competitors (the lower the better).

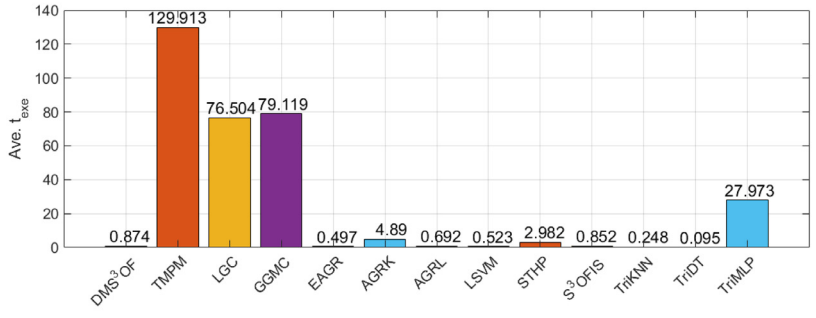
the previously specified parameter setting without any tuning for performance optimization. As aforementioned, one may further improve the classification performance of DMS^{3}OF by adjusting the parameter setting.

5. Conclusion and future works

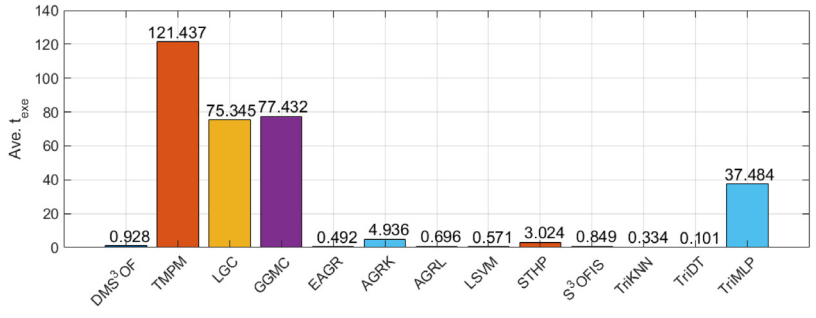
In this paper, a novel semi-supervised learning fuzzy inference system that comprises two cutting-edge zero-order EFS



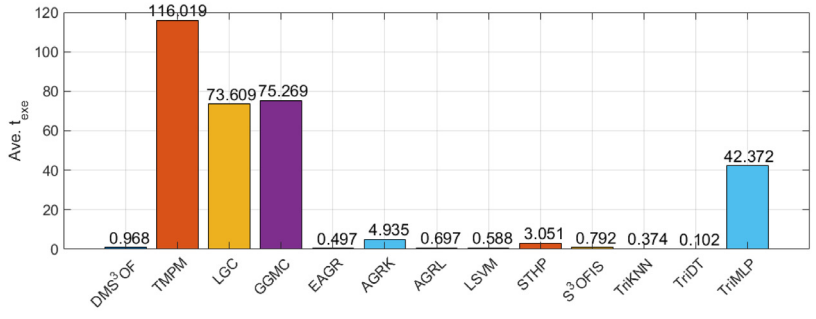
(a) Average training time consumption (in seconds) under the split ratio of 1:9



(b) Average training time consumption (in seconds) under the split ratio of 2:8



(c) Average training time consumption (in seconds) under the split ratio of 3:7



(d) Average training time consumption (in seconds) under the split ratio of 4:6

Fig. 6. Average training time cost comparison between DMS^{3}OF and 12 competitors (the lower the better).

models, namely, SOFIS+ and SOFBIS, called DMS^{3}OF is presented for data stream learning in infinite delay environments. After being primed with labelled data samples, DMS^{3}OF can effectively learn from unlabelled streaming data chunk-by-chunk by exploiting the pseudo-labelling technique for greater classification performance. Thanks to its duel-model structure and the specially designed semi-supervised learning scheme, DMS^{3}OF is able to combine the merits of the two EFS models such that

Table 11
Classification performance comparison on Caltech101.

Algorithm	15	30
DMS^{3}OF	**0.115 ± 0.018**	**0.085 ± 0.018**
DEFEATnet [37]	0.287 ± 0.006	0.224 ± 0.010
ICAC [38]	0.285 ± 0.006	0.234 ± 0.008
CASE-LLC-SVM [39]	0.360 ± 0.004	0.286 ± 0.012
CEC [40]	0.255 ± 0.007	0.211 ± 0.006
CEC-CNN [40]	–	0.139 ± 0.007
YCbCr-SIFT+LSC+ELM [41]	0.274 ± 0.007	0.220 ± 0.005
SPFF [36]	0.325	0.255
DSDPL [42]	0.292	0.232

it continuously self-improves its knowledge base by generating high-quality pseudo-labelled data from unlabelled samples whilst effectively preventing pseudo-labelling errors from affecting decision-making. Numerical examples on a wide range of benchmark problems demonstrate the superior performance of DMS^{3}OF over the state-of-the-art approaches.

There are several considerations for future works. First, to suppress error propagation, the current DMS^{3}OF model only updates SOFIS+ model with pseudo-labelled samples. This is because pseudo-labelling errors can easily propagate within SOFBIS due to its belief structure and unique decision-making mechanism. It would be beneficial to modify SOFBIS such that it can self-update from pseudo-labelled samples to augment its knowledge base whilst being less impacted by pseudo-labelling errors on decision-making. With such modification, DMS^{3}OF can achieve even greater classification accuracy. Second, DMS^{3}OF requires users to specify the level of granularity. Although this is not a user- and problem-specific parameter and can be determined without prior knowledge of the problem, the performance of DMS^{3}OF can be unfavourably affected if its value is not set properly. Therefore, it would be helpful to develop an automated approach to self-determine the most suitable setting based on the statistical characteristics of the data. However, this is a highly challenging task considering the nonstationary nature of data streams.

CRediT authorship contribution statement

Xiaowei Gu: Formal analysis, Methodology, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] C.M. Bishop, Pattern Recognition and Machine Learning, Springer, New York, 2006.
[2] K.R. Müller, S. Mika, G. Rätsch, K. Tsuda, B. Schölkopf, An introduction to kernel-based learning algorithms, IEEE Trans. Neural Netw. 12 (2) (2001) 181–201.
[3] Y. Song, J. Lu, H. Lu, G. Zhang, Fuzzy clustering-based adaptive regression for drifting data streams, IEEE Trans. Fuzzy Syst. 28 (3) (2020) 544–557.
[4] X. Zhu, Semi-supervised learning literature survey, 2008.
[5] J.E. van Engelen, H.H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2) (2020) 373–440.
[6] G. Huang, C. Du, The high separation probability assumption for semi-supervised learning, IEEE Trans. Syst. Man, Cybern. Syst. 52 (12) (2022) 7561–7573.
[7] K.P. Bennett, A. Demiriz, Semi-supervised support vector machines, in: Advances in Neural Information Processing Systems, 1999, pp. 368–374.
[8] O. Chapelle, V. Sindhwani, S. Keerthi, Optimization techniques for semi-supervised support vector machines, J. Mach. Learn. Res. 9 (2008) 203–233.
[9] Z.H. Zhou, M. Li, Tri-training: Exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1529–1541.
[10] D.-H. Lee, Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks, in: Workshop on Challenges in Representation Learning, ICML, 2013, p. 2.
[11] M. Wang, W. Fu, S. Hao, D. Tao, X. Wu, Scalable semi-supervised learning by efficient anchor graph regularization, IEEE Trans. Knowl. Data Eng. 28 (7) (2016) 1864–1877.
[12] G. Huang, S. Song, J. Gupta, C. Wu, Semi-supervised and unsupervised extreme learning machines, IEEE Trans. Cybern. 44 (12) (2014) 2405–2417.
[13] U. Maulik, D. Chakraborty, A self-trained ensemble with semisupervised SVM: An application to pixel classification of remote sensing imagery, Pattern Recognit. 44 (3) (2011) 615–623.
[14] S. Qiao, W. Shen, Z. Zhang, B. Wang, A. Yuille, Deep co-training for semi-supervised image recognition, in: European Conference on Computer Vision, 2018, pp. 135–152.
[15] P.K. Mallapragada, R. Jin, A.K. Jain, Y. Liu, SemiBoost: Boosting for semi-supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 31 (11) (2008) 2000–2014.
[16] D. Zhou, O. Bousquet, T.N. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in: Adv. Neural. Inform. Process Syst, 2004, pp. 321–328.
[17] L. Gómez-Chova, G. Camps-Valls, J. Munoz-Mari, J. Calpe, Semisupervised image classification with Laplacian support vector machines, IEEE Geosci. Remote Sens. Lett. 5 (3) (2008) 336–340.
[18] J. Wang, T. Jebara, S.F. Chang, Semi-supervised learning using greedy max-cut, J. Mach. Learn. Res. 14 (2013) 771–800.
[19] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, in: International Conference on Machine Learning, 2010, pp. 679–689.
[20] X. Gu, A self-training hierarchical prototype-based approach for semi-supervised classification, Inf. Sci. (Ny). 535 (2020) 204–224.
[21] M. Pratama, A. Ashfahani, A. Hady, Weakly supervised deep learning approach in streaming environments, in: IEEE International Conference on Big Data, 2019, pp. 1195–1202.
[22] M. Das, M. Pratama, J. Zhang, Y.S. Ong, A skip-connected evolving recurrent neural network for data stream classification under label latency scenario, in: AAAI Conference on Artificial Intelligence, 2020, pp. 3717–3724.
[23] M. Das, M. Pratama, T. Tjahjowidodo, A self-evolving mutually-operative recurrent network-based model for online tool condition monitoring in delay scenario, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 2775–2783.
[24] M. Pratama, C. Za'in, E. Lughofer, E. Pardede, D. Rahayu, Scalable teacher forcing network for semi-supervised large scale data streams, Inf. Sci. (Ny). 576 (2021) 407–431.
[25] X. Gu, An explainable semi-supervised self-organizing fuzzy inference system for streaming data classification, Inf. Sci. (Ny). 583 (2022) 364–385.

Table 12
Classification performance comparison on Caltech256.

Algorithm	15	30	45	60
DMS^{3}OF	0.355 ± 0.015	0.306 ± 0.006	0.288 ± 0.005	0.278 ± 0.009
CEC [40]	0.608 ± 0.004	0.552 ± 0.003	0.521 ± 0.003	–
CEC-FV [40]	0.595 ± 0.004	0.503 ± 0.004	0.512 ± 0.003	–
SWSS-DeCAF [43]	0.385 ± 0.004	0.323 ± 0.007	0.302 ± 0.005	0.272 ± 0.004
SWSS-VGG [43]	**0.306 ± 0.005**	**0.264 ± 0.005**	**0.252 ± 0.004**	**0.237 ± 0.005**
DGFLP [44]	0.359	0.308	–	–
LR-GCC-FV [45]	0.586 ± 0.004	0.509 ± 0.003	–	–
SPFF [36]	0.667 ± 0.001	0.623 ± 0.003	0.599 ± 0.003	0.567 ± 0.002

[26] X. Gu, P. Angelov, Z. Zhao, Self-organizing fuzzy inference ensemble system for big streaming data classification, Knowledge-Based Syst. 218 (2021) 106870.
[27] X. Gu, P.P. Angelov, Q. Shen, Self-organizing fuzzy belief inference system for classification, IEEE Trans. Fuzzy Syst. 30 (12) (2022) 5473–5483.
[28] V.M.A. Souza, D.F. Silva, Classification of evolving data streams with infinitely delayed labels, in: IEEE International Conference on Machine Learning and Applications, 2015, pp. 214–219.
[29] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[30] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015, pp. 1–14.
[31] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: A geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) (2006) 2399–2434.
[32] V.M.A. Souza, D.F. Silva, J. Gama, G.E.A.P.A. Batista, Data stream classification guided by clustering on nonstationary environments and extreme verification latency, in: SIAM International Conference on Data Mining, 2015, pp. 873–881.
[33] A. Ashfahani, M. Pratama, E. Lughofer, Y.S. Ong, DEVDAN: Deep evolving denoising autoencoder, Neurocomputing 390 (2020) 297–314.
[34] C. Za'in, A. Ashfahani, M. Pratama, E. Lughofer, E. Pardede, Scalable teacher-forcing networks under spark environments for large-scale streaming problems, in: IEEE Conference on Evolving and Adaptive Intelligent Systems, 2020, pp. 1–8.
[35] C. Za'in, M. Pratama, E. Pardede, Evolving large-scale data stream analytics based on scalable PANFIS, Knowledge-Based Syst. 166 (2019) 186–197.
[36] F. Yang, Z. Ma, M. Xie, Image classification with superpixels and feature fusion method, J. Electron. Sci. Technol. 19 (1) (2021) 70–78.
[37] S. Gao, L. Duan, I.W. Tsang, DEFEATnet—A deep conventional image representation for image classification, IEEE Trans. Circuits Syst. Video Technol. 26 (3) (2016) 494–505.
[38] C. Zhang, J. Cheng, Q. Tian, Incremental codebook adaptation for visual representation and categorization, IEEE Trans. Cybern. 48 (7) (2018) 2012–2023.
[39] W. Luo, J. Li, J. Yang, W. Xu, J. Zhang, Convolutional sparse autoencoders for image classification, IEEE Trans. Neural Networks Learn. Syst. 29 (7) (2018) 3289–3294.
[40] C. Zhang, Q. Huang, Q. Tian, Contextual exemplar classifier-based image representation for classification, IEEE Trans. Circuits Syst. Video Technol. 27 (8) (2017) 1691–1699.
[41] Q. Li, Q. Peng, J. Chen, C. Yan, Improving image classification accuracy with ELM and CSIFT, Comput. Sci. Eng. 21 (5) (2019) 26–34.
[42] G. Belous, A. Busch, Y. Gao, Dual subspace discriminative projection learning, Pattern Recognit. 111 (2021) 107581.
[43] C. Zhang, J. Cheng, Q. Tian, Structured weak semantic space construction for visual categorization, IEEE Trans. Neural Networks Learn. Syst. 29 (8) (2018) 3442–3451.
[44] G. Lin, K. Liao, B. Sun, Y. Chen, F. Zhao, Dynamic graph fusion label propagation for semi-supervised multi-modality classification, Pattern Recognit. 68 (2017) 14–23.
[45] C. Zhang, C. Liang, L. Li, J. Liu, Q. Huang, Q. Tian, Fine-grained image classification via low-rank sparse coding with general and class-specific codebooks, IEEE Trans. Neural Networks Learn. Syst. 28 (7) (2017) 1550–1559.