Applied Soft Computing Journal 170 (2025) 112719

Contents lists available at ScienceDirect

Applied Soft Computing

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

A semi-supervised soft prototype-based autonomous fuzzy ensemble system for network intrusion detection

Xiaowei Gu[a,*], Gareth Howells[b], Haiyue Yuan[c]

[a] *School of Computer Science and Electronic Engineering, University of Surrey, Guildford GU2 7XH, UK*
[b] *School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, UK*
[c] *School of Computing, University of Kent, Canterbury CT2 7NZ, UK*

HIGHLIGHTS

- A novel transparent soft prototype-based fuzzy ensemble system is proposed.
- The system learns from imbalanced network data streams for intrusion detection.
- The system self-improves from unlabelled data via collaborative pseudo-labelling.
- The system automatically identifies and prunes weaker base models.
- The system facilitates expert involvement by actively learning from challenging data.

ARTICLE INFO

Keywords:
Data stream
Ensemble learning
Intrusion detection
Online semi-supervised learning
Soft prototype

ABSTRACT

The escalating cyber-attacks in recent years have created a significant demand for advanced intrusion detection systems. However, the dynamic characteristics of network data streams, scarcity of labelled data, imbalance of class distribution and concerns about the explainability of predictive models have greatly restricted the applicability of conventional machine learning and cutting-edge deep learning techniques in real-world scenarios for intrusion detection. In this paper, a novel soft prototype-based fuzzy ensemble intrusion detection system is proposed to autonomously exploit semi-supervised learning from network data streams by exploiting the pseudo-labelling method. To reduce the pseudo-labelling errors, although the proposed ensemble system involves all the base models for joint pseudo-labelling, it only utilises pseudo-labelled data with the highest consensus for self-improvement. To foster generalisation, the proposed ensemble system leverages sampling techniques to address the class imbalance within the labelled and pseudo-labelled data, and the qualities of base models are constantly monitored, ensuring that weaker base models are efficiently substituted. Additionally, instead of discarding these challenging-to-classify samples during online semi-supervised learning, the proposed ensemble system summarises them into a smaller number of unlabelled soft prototypes, allowing human experts to contribute to the learning at any point by manually labelling these soft prototypes to further augment the learned knowledge base. Numerical examples on public network intrusion detection datasets demonstrated the superior performance of the proposed ensemble system.

1. Introduction

Over the last couple of years, cyber-attacks have posed significant and persistent challenges to the governments, businesses and individuals due to the increasing reliance on digital technologies. According to the Cyber Security Breaches Survey published by the UK government in April 2024 [1], 70 % of medium businesses, 74 % of large businesses and 66 % of high-income charities have encountered cybersecurity breaches and/or cyber-attacks in the last 12 months. The widespread and persistent threat has resulted in a great demand for defensive techniques to detect and neutralise cyber-attacks. As one of the most effective security solutions against cyber-attacks, intrusion detection systems (IDSs) have been a hot topic in cyber security research [2,3].

IDSs are a key component of cybersecurity infrastructure. They

* Corresponding author.
E-mail addresses: xiaowei.gu@surrey.ac.uk (X. Gu), gareth.howells@essex.ac.uk (G. Howells), h.yuan957@googlemail.com (H. Yuan).

https://doi.org/10.1016/j.asoc.2025.112719
Received 20 March 2024; Received in revised form 30 November 2024; Accepted 29 December 2024
Available online 7 January 2025

shield computer networks by monitoring and analysing network traffic to detect malicious activities. Based on the detection approaches used, existing IDSs can be classified into two distinct categories, namely, signature-based and anomaly-based [4,5]. Signature-based IDSs (SIDSs) are based on pattern matching methods. SIDSs monitor network activities and attempt to find patterns that match against the signature database of known attacks. SIDSs typically offer excellent detection accuracy for previously known attacks, but struggle with novel attacks due to the lack of matching signatures [6]. In contrast, anomaly-based IDSs (AIDSs) primarily leverage machine learning techniques to model the behaviour of computer networks and distinguish between normal and malicious traffic patterns [7]. Thanks to the utilisation of advanced machine learning models that can be trained from large amounts of data, AIDSs are capable of identifying cyber-attacks of unseen types without relying on the signature database. The increasing complexity and diversity of cyber-attacks, coupled with the limitations of SIDSs, have made AIDSs an intensively researched direction in the recent decades.

Depending on the learning techniques used, AIDSs can be categorised into three major types, namely, supervised, unsupervised and semi-supervised [2,6]. Supervised AIDSs utilise labelled training data to construct predictive models for classifying benign and malicious behaviours. Contrary to supervised AIDSs, unsupervised AIDSs do not require labelled data for training. They analyse the underlying patterns of unlabelled data and identify outliers by thresholding to detect potential intrusions. Semi-supervised AIDSs strike a balance by constructing predictive models using a small amount of labelled training data combined with a greater amount of unlabelled training data.

Existing supervised AIDSs predominantly rely on conventional supervised learning techniques [6,8–10], such as decision tree, naïve Bayes, evolutionary algorithm, fuzzy system, support vector machine, k-nearest neighbour, etc. Conventional supervised machine learning techniques have achieved many successes in real-world applications on network intrusion detection. However, their effectiveness has been diminishing due to the rising sophistication and complexity of cyber-attacks. To attain greater intrusion detection performance, ensemble learning techniques, e.g., bagging and boosting, have been exploited to build more powerful AIDSs by combining multiple simpler predictive models [5,11–13]. In recent years, there have been a significant increase in the use of deep neural networks (DNNs) for constructing AIDSs [14–17]. DNNs have demonstrated promising results on many challenging problems that conventional machine learning approaches struggle with [18]. The success of DNNs comes from their abilities to utilise huge amounts of training data to learn extremely abstract and nonlinear representations [19]. Currently, DNN-based AIDSs are widely considered as the state-of-the-art (SOTA) in intrusion detection by offering greater accuracy on detecting sophisticated and complex cyber-attacks, outperforming traditional AIDSs [8]. However, DNNs are typical “black box” models with very large numbers of weights that cannot be linked to the physical environment of the given problem. The rationales behind the internal reasoning and decision-making of DNNs cannot be easily determined by humans [20]. The lack of explainability has greatly impaired the trustability of DNNs in high-stake applications, such as intrusion detection. In addition, DNNs are also vulnerable to adversarial attacks [19].

In practice, two significant obstacles faced by supervised AIDSs are the scarcity of labelled training data and the lack of malicious records within the collected data. Collecting large amounts of labelled data for training machine learning models is often impractical because manual labelling can be extremely expensive and time consuming. Moreover, given the nature of network traffic, most activities typically are benign and malicious activities are rare. The lack of training samples of malicious class can introduce significant bias to supervised AIDSs, deteriorating the detection performance. In contrast, unsupervised AIDSs overcome the two obstacles by leveraging unsupervised learning techniques, such as clustering, autoencoder, one-class support vector machine, etc., to discover the inherent patterns within unlabelled data and identify outliers as potential attacks [2,21,22]. However, unsupervised AIDSs rely on the assumption that benign activities significantly outnumber malicious activities. If such an assumption does not hold true, unsupervised AIDSs can produce many false alarms and fail to recognise intrusions effectively. In addition, their performances are also highly sensitive to the chosen thresholds, which typically require prior knowledge of the problem and a high-level of human expertise to set properly.

On the other hand, semi-supervised learning incorporates both labelled and unlabelled training data to build stronger predictive models [23]. By exploiting the rich information from abundant unlabelled data, semi-supervised learning minimises the need for expensive human labour in data labelling and is more applicable to real-life applications compared with supervised learning [4]. Hence, semi-supervised learning techniques have been increasingly explored in the design of AIDSs. A number of semi-supervised AIDSs have been proposed in the literature in recent years and demonstrated promising results, e.g., [24–28]. However, the vast majority of existing semi-supervised AIDSs are limited to offline learning from static data. They lack the capacity to autonomously self-update from streaming data and cannot self-adapt to novel data patterns. Consequently, they are unable to effectively handle concept drifts in data streams within dynamic environments. Indeed, online semi-supervised learning from large-scale data streams, especially, in infinite delay environments [29], is widely recognised as a highly challenging task and has not been sufficiently investigated [30]. Despite strong demand, to date, very few semi-supervised AIDSs have been designed with the capability of online learning from data streams to address the challenges posed by the constantly evolving cyber-attack behaviours [4,31].

In short, the following four key challenges associated with existing AIDSs significantly limit their applicability, especially in large-scale networks that connect a wide range of devices [4,32].

1) **Scarcity of labels:** Network traffic data is generated rapidly over time, making it extremely difficult, if not impossible, to manually label large volumes of records in real time. Hence, only a small portion of network activities are typically labelled by human experts, leaving the majority unlabelled [33].
2) **Concept drifts:** The underlying patterns and distribution of network traffic data streams are constantly changing due to the dynamic nature of the environments [34]. It is also practically impossible to capture all relevant activities in a single attempt in any real-life applications, and unseen activities may emerge at any time.
3) **Class imbalance:** The class distribution of network traffic data is often highly imbalanced as malicious activities are significantly rarer than benign activities [4]. The huge imbalance between malicious and benign traffics can lead to poor generalisation for detection models trained from network traffic data.
4) **Lack of explainability:** Network intrusion detection is a high-stake application. Human experts need to understand the rationale behind the predictions made by detection models to effectively tackle potential threats. A lack of explainability can significantly limit the applicability and acceptance of these models in critical environments.

To address the four key issues, in this paper, a novel semi-supervised soft prototype-based autonomous fuzzy ensemble system (S^{3}PAFES) is proposed for network intrusion detection in infinite delay environments. The proposed S^{3}PAFES employs a group of soft prototype-based autonomous fuzzy inference systems (SPAFISs) as its base models and is able to perform semi-supervised learning from network data streams by exploiting the pseudo labelling method [35].

SPAFIS is a recently introduced zero-order evolving fuzzy system (EFS) with highly transparent structure and human-interpretable internal reasoning for real-time intrusion detection from network traffic data streams [36]. Different from conventional zero-order EFSs that are based on crisp prototypes [37], SPAFIS utilises soft prototypes instead. The concept of soft prototypes is aligned with the concept of soft clustering [38]. In SPAFIS, each soft prototype is a weighted combination of all the data samples of the same class with the respective weights calculated based on mutual distances between data. In comparison with crisp prototypes, soft prototypes have stronger descriptive abilities to better capture the underlying structure and local patterns of data, enabling SPAFIS to attain greater classification performance.

Once primed with a small amount of labelled training data, S^{3}PAFES is capable of autonomously self-learning from unlabelled network data streams in a chunk-wise manner with minimal human expert involvement. Given the crucial impact of pseudo labels on the model's predictive performance, S^{3}PAFES uses all the base models to perform pseudo-labelling jointly during online semi-supervised learning and only selects these unlabelled data samples with predicted class labels exhibiting the highest consensus among the base models for self-improving, thereby greatly enhancing the robustness and reliability of the pseudo-labelling process. To tackle the imbalance in class distribution of network data streams [27], S^{3}PAFES utilises oversampling to enhance the representation of the minority class in the labelled training data during supervised learning, and applies random down sampling to decrease the prevalence of the majority class in the pseudo-labelled training data during semi-supervised learning. Furthermore, S^{3}PAFES constantly monitors individual base learners throughout the learning process. The base learners with poor out-of-sample prediction performance will be pruned promptly and replaced by new models initialised with a small group of historical data to maintain the high-level performance of the overall ensemble system.

Although S^{3}PAFES is designed to operate in a fully autonomous manner, human experts can also be involved optionally during the semi-supervised learning process and use the expert knowledge to help S^{3}PAFES achieve better prediction performance. Existing online semi-supervised learning models that are based pseudo labels typically choose to discard these unlabelled samples with low classification confidence to avoid deteriorating the prediction performances [37,39]. However, thanks to the unique online active learning scheme S^{3}PAFES summarises these challenging unlabelled samples into a much smaller number of unlabelled soft prototypes to capture the underlying patterns, rather than discarding them. These unlabelled soft prototypes contain the highly condensed information of unseen data patterns and can be used for sharpening classification boundaries greatly. At any point of the semi-supervised learning process, human experts have the flexibility to intervene by manually inspecting these unlabelled soft prototypes and assigning class labels, contributing to the continuous self-improvement of S^{3}PAFES.

To summarise, the novel features of the proposed S^{3}PAFES, designed to address the aforementioned four challenges faced by existing AIDSs, include:

1) A two-step collaborative pseudo-labelling scheme for semi-supervised learning from unlabelled data without involvement of human experts (Challenge 1: Scarcity of labels).
2) A flexible system architecture supporting online incremental learning from data streams in a single-pass manner (Challenge 2: Concept drifts).
3) The combined use of oversampling and random down sampling to tackle class imbalance in labelled and pseudo-labelled data (Challenge 3: Class imbalance).
4) A highly transparent prototype-based structure in the form of IF-THEN rules, combined with a human-interpretable decision-making mechanism (Challenge 4: Lack of explainability).

Key contributions of this paper are as follows:

1) A novel transparent soft prototype-based fuzzy ensemble framework with human-interpretable internal reasoning is proposed to autonomously learn from network traffic data streams with minimal human expert involvement.
2) An online semi-supervised learning scheme utilising two-step collaborative pseudo-labelling is designed to select only high-quality pseudo-labelled samples with high consensus among base learners for system self-improvement.
3) A base learner performance monitoring scheme is introduced to automatically identify and replace weaker base learners with poor out-of-sample classification performance and help the ensemble system maintain high-level performance.
4) Different sampling techniques are integrated into the learning process of base learners to correct the class imbalance in labelled and pseudo-labelled data samples, enhancing generalisation on minority class.
5) A challenging sample learner is introduced to the ensemble framework in parallel with base learners to condense hard-to-classify samples into a small set of unlabelled soft prototypes, thereby greatly facilitating the engagement of human experts by minimising manual labelling effort.

The remainder of this paper is organised as follows. Section 2 summarises the details of the SPAFIS as the theoretical background. The proposed S^{3}PAFES are described in Section 3. Numerical examples are presented in Section 4 for performance demonstration. This paper is concluded by Section 5, and directions for future work are also given in this section.

2. Preliminaries - SPAFIS

In this section, technique details of SPAFIS are recalled to make this paper self-contained. Interested readers may refer to [36] for more details.

First of all, it is assumed that $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_K\}$ ($\boldsymbol{x}_i = [x_{i,1}, x_{i,2}, .., x_{i,M}]^T \in \mathbf{X}$) is a particular data stream in the M-dimensional data space $\mathscr{R}^M$ composed of data samples of C different classes, where K is the total number of samples of the data stream; the subscript i denotes the time instance at which $\boldsymbol{x}_i$ is observed and $y_i \in \{1, 2, \ldots, C\}$ denotes the class label of $\boldsymbol{x}_i$. Data samples of the data stream $\mathbf{X}$ continuously arrive in chunks, namely, $\mathbf{X}_n = \{\boldsymbol{x}_{n,1}, \boldsymbol{x}_{n,2}, \ldots, \boldsymbol{x}_{n,K_n}\}$ ($n = 1, 2, 3, \ldots$; K_n is the cardinality of $\mathbf{X}_n$). Based on the corresponding class labels $\mathbf{Y}_n = \{y_{n,1}, y_{n,2}, \ldots, y_{n,K_n}\}$, $\mathbf{X}_n$ can be further divided into C non-overlapping subsets with each subset consisted of data samples of the same class, namely, $\mathbf{X}_n^c = \{\boldsymbol{x}_{n,1}^c, \boldsymbol{x}_{n,2}^c, \ldots, \boldsymbol{x}_{n,K_n^c}^c\}$ ($c = 1, 2, \ldots, C$; K_n^c is the cardinality of $\mathbf{X}_n^c$). Hence, there are $\mathbf{X}_n^i \cap \mathbf{X}_n^j = \varnothing\ \forall i \neq j$, and $\mathbf{X}_n^1 \cup \mathbf{X}_n^2 \cup \ldots \cup \mathbf{X}_n^C = \mathbf{X}_n$. In this study, the city block distance is employed as the default distance measure, namely, $\|\boldsymbol{x}_j - \boldsymbol{x}_i\|_1 = \sum_{m=1}^{M} |x_{j,m} - x_{i,m}|$ same as [36]. The main reason for choosing the city block distance is because of its robustness to dimensionality and computational simplicity. However, one may consider using other commonly used distance metrics, such as Euclidean distance, Mahalanobis distance, etc. It is also worth noting that different data chunks do not necessarily have the same sizes.

SPAFIS is a recently introduced zero-order EFS to learn from data streams on a chunk-by-chunk basis and condense the learned knowledge into a number of soft prototypes, representing the local peaks of multimodal data distribution. The knowledge base of SPAFIS is composed of C soft prototype-based IF-THEN fuzzy rules in the form of

Eq. (1) (one rule per class) [36].

$$R^c: \quad \begin{array}{c}\text{IF}(\boldsymbol{x} \sim \boldsymbol{s}_1^c)\text{OR}(\boldsymbol{x} \sim \boldsymbol{s}_2^c)\text{OR}\ldots\text{OR}(\boldsymbol{x} \sim \boldsymbol{s}_{S^c}^c) \\ \text{THEN}(y = c)\end{array} \tag{1}$$

where the superscript c represents the cth class, $c = 1, 2, \ldots, C$; "$\sim$" denotes similarity; $\boldsymbol{s}_j^c$ is the jth soft prototype of the cth IF-THEN rule, R^c; S^c is the total number of soft prototypes identified from data of the cth class; $\mathbf{S}^c$ is the collection of soft prototypes associated with R^c.

In the next two subsections, SPAFIS's online chunk-wise system identification and decision-making processes are detailed [36]. For clarity, a list of key notations and definitions used in this section is provided in Table 1.

2.1. Online system identification scheme

SPAFIS learns from data streams in a chunk-wise manner. Once the nth data chunk $\mathbf{X}_n$ and the corresponding class labels $\mathbf{Y}_n$ become available, SPAFIS firstly calculates the pairwise distances between any two data samples of $\mathbf{X}_n$ and obtain the following $K_n \times K_n$ dimensional symmetric matrix $\mathbf{d}_n$:

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

where $\left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,i}\right\|_1^2 = \left(\sum_{m=1}^{M}\left|x_{n,j,m} - x_{n,i,m}\right|\right)^2$ represents the element located at the jth row and ith column of the matrix $\mathbf{d}_n$, which is the squared city block distance between data samples $\boldsymbol{x}_{n,j}$ and $\boldsymbol{x}_{n,i}$.

The average distance between any two neighbouring data samples within $\mathbf{X}_n$ at the Gth level of granularity, denoted as $\overline{d}_{n,G}$, can be derived iteratively by Eq. (3):

Table 1
Key notations and definitions.

Notation	Definition
$\mathbf{X}$	Data stream
$\boldsymbol{x}_i$	The ith sample of the data stream
y_i	The class label of $\boldsymbol{x}_i$
M	The dimensionality of data samples
$\mathscr{R}^M$	The M-dimenisonal data space
K	Total number of samples of the data stream
C	Number of classes
$\mathbf{X}_n$	The nth data chunk
$\mathbf{Y}_n$	The class labels of $\mathbf{X}_n$
K_n	The cardinality of $\mathbf{X}_n$
$\boldsymbol{x}_{n,i}$	The ith data sample of $\mathbf{X}_n$
$y_{n,i}$	The class label of $\boldsymbol{x}_{n,i}$
$\mathbf{X}_n^c$	The subset of $\mathbf{X}_n$ belonging to the cth class
K_n^c	The cardinality of $\mathbf{X}_n^c$
$\boldsymbol{x}_{n,i}^c$	The ith data sample of $\mathbf{X}_n^c$
$D\left(\boldsymbol{x}_{n,i}^c\right)$	Data density of $\boldsymbol{x}_{n,i}^c$
R^c	The fuzzy rule of the cth class
$\mathbf{S}^c$	The collection of soft prototypes associated with R^c
$\boldsymbol{s}_i^c$	The ith soft prototype of $\mathbf{S}^c$
ν_j^c	The support of $\boldsymbol{s}_i^c$
$\lambda^c(\boldsymbol{x}_i)$	The confidence score of R^c on $\boldsymbol{x}_i$
$\widehat{y}_i$	The predicted label of $\boldsymbol{x}_i$
$\mathbf{d}_n$	Pairwise distance matrix between data samples of $\mathbf{X}_n$
γ_G	Self-adaptive distance threshold at the Gth level of granularity
$\mathbf{d}_n^c$	Pairwise distance matrix between data samples of $\mathbf{X}_n^c$
$\mathbf{A}_n^c$	Symmetric adjacency matrix derived from $\mathbf{d}_n^c$
$\mathbf{P}_n^c$	The collection of soft prototypes extracted from $\mathbf{X}_n^c$
P_n^c	The cardinality of $\mathbf{P}_n^c$
$\boldsymbol{p}_{n,i}^c$	The ith soft prototype of $\mathbf{P}_n^c$
$\rho_{n,i}^c$	The support of $\boldsymbol{p}_{n,i}^c$
$\mathbf{R}^c$	The subset of $\mathbf{S}^c$ to be pruned
R^c	The cardinality of $\mathbf{R}^c$
$\boldsymbol{r}_i^c$	The ith soft prototype of $\mathbf{R}^c$
β_i^c	The support of $\boldsymbol{r}_i^c$

$$\overline{d}_{n,g} = \frac{1}{\sum_{i=1}^{K_n-1}\sum\limits_{j=i+1}^{K_n} w_{g,i,j}}\sum_{i=1}^{K_n-1}\sum_{j=i+1}^{K_n} w_{g,i,j}\left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,i}\right\|_1^2 \tag{3}$$

where $g = 1, 2, \ldots, G$; $w_{g,i,j} = \begin{cases}1, & if \left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,i}\right\|_1^2 \le \overline{d}_{n,g-1} \\ 0, & else\end{cases}$; $\overline{d}_{n,0} = \frac{2}{L_n(L_n-1)}\sum_{i=1}^{L_n-1}\sum_{j=i+1}^{L_n}\left\|\boldsymbol{x}_{n,j} - \boldsymbol{x}_{n,i}\right\|_1^2$ is the average distance between any two data samples of $\mathbf{X}_n$. One can see from Eq. (3) that, essentially, $\overline{d}_{n,g}$ is the average distance between any two data samples whose distance is no greater than $\overline{d}_{n,g-1}$. By iteratively applying Eq. (3), the resulting $\overline{d}_{n,G}$ provides an estimation of the maximum distance between any two neighbouring data samples within $\mathbf{X}_n$ at the Gth level of granularity specified by users.

Next, the self-adaptive distance threshold, γ_G is set as $\overline{d}_{n,G}$ ($\gamma_G \leftarrow \overline{d}_{n,G}$) if $\mathbf{X}_n$ is the very first data chunk, namely, $n = 1$. Otherwise, γ_G is updated with $\overline{d}_{n,G}$ using Eq. (4) to ensure that γ_G provides an accurate estimation of the maximum distance between any two data samples of $\mathbf{X}_n$ and the previous data chunks, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_{n-1}$ that can be seen as neighbours in the data space.

$$\gamma_G \leftarrow \frac{\gamma_G \sum_{j=1}^{n-1} K_j + \overline{d}_{n,G} K_n}{\sum_{j=1}^{n} K_j} \tag{4}$$

Comparing between Eq. (3) and Eq. (4) one can see that γ_G provides an estimation of the maximum distance between neighbouring data samples at the Gth level of granularity globally, whilst $\overline{d}_{n,G}$ provides an estimation of such distance between such data samples of $\mathbf{X}_n$ locally.

Then, the current data chunk $\mathbf{X}_n$ is divided into C nonoverlapping subsets (denoted as $\mathbf{X}_n^1$, $\mathbf{X}_n^2$,..., $\mathbf{X}_n^C$) according to the corresponding class labels, $\mathbf{Y}_n$ with each subset only containing data samples of the same class. For each individual subset (assuming the cth one), the $K_n^c \times K_n^c$ dimensional pairwise distance matrix $\mathbf{d}_n^c$ between any two data samples of the cth class is firstly obtained from $\mathbf{d}_n$ based on the sample indices in $\mathbf{X}_n$:

$$\mathbf{d}_n^c = \left[\left\|\boldsymbol{x}_{n,j}^c - \boldsymbol{x}_{n,i}^c\right\|_1^2\right]_{j=1:K_n}^{i=1:K_n} \tag{5}$$

Based on $\mathbf{d}_n^c$, the symmetric adjacency matrix, $\mathbf{A}_n^c = \left[A_{n,i,j}^c\right]_{j=1:K_n^c}^{i=1:K_n^c}$ is derived using Eq. (6):

$$A_{n,i,j}^c = \begin{cases}1, & if \left\|\boldsymbol{x}_{n,j}^c - \boldsymbol{x}_{n,i}^c\right\|_1^2 \le \gamma_G \\ 0, & else\end{cases} \tag{6}$$

Note that, there is $A_{n,i,j}^c = A_{n,j,i}^c\ \forall j, i$. If the value of $A_{n,i,j}^c$ (and $A_{n,j,i}^c$) is equal to 1, it suggests that $\boldsymbol{x}_{n,j}^c$ and $\boldsymbol{x}_{n,i}^c$ are considered as neighbours at the level of granularity specified by users.

Based on $\mathbf{d}_n^c$ and $\mathbf{A}_n^c$, the data density at each individual sample of $\mathbf{X}_n^c$ can be obtained using Eq. (7):

$$D\left(\boldsymbol{x}_{n,i}^c\right) = \sum_{j=1}^{K_n^c} A_{n,i,j}^c \alpha\left(\boldsymbol{x}_{n,j}^c, \boldsymbol{x}_{n,i}^c\right) \tag{7}$$

where $\alpha\left(\boldsymbol{x}_{n,j}^c, \boldsymbol{x}_{n,i}^c\right) = \exp\left(-\frac{\left\|\boldsymbol{x}_{n,j}^c - \boldsymbol{x}_{n,i}^c\right\|_1^2}{\gamma_G}\right)$. By utilising only the distances between neighbouring samples for data density calculation, Eq. (7) highlights samples located in denser regions, thereby effectively unveiling local data patterns.

The local peaks of the data density are identified from $\mathbf{X}_n^c$ using Condition 1 [36]. Data samples with higher data density values than their neighbours will be selected and added to $\acute{\mathbf{X}}_n^c$

$$Cond.\quad 1: \quad \begin{array}{c} if\left(\max_{j=1,2,\ldots,L_n^c}\left(A_{n,i,j}^c D\left(\boldsymbol{x}_{n,j}^c\right)\right)=D\left(\boldsymbol{x}_{n,i}^c\right)\right) \\ then\left(\acute{\mathbf{X}}_n^c \leftarrow \acute{\mathbf{X}}_n^c \cup\left\{\boldsymbol{x}_{n,i}^c\right\}\right) \end{array} \tag{8}$$

A collection of soft prototypes $\mathbf{P}_n^c$ are extracted from $\mathbf{X}_n^c$ by associating each individual data sample to every local peak with a certain weight calculated based on its distances to the local peaks:

$$\boldsymbol{p}_{n,i}^c=\sum\nolimits_{j=1}^{K_n^c}\alpha_{n,i,j}^*\boldsymbol{x}_{n,j}^c \tag{9}$$

and the support of $\boldsymbol{p}_{n,i}^c$, $\rho_{n,i}^c$ is calculated by Eq. (10):

$$\rho_{n,i}^c=\sum_{j=1}^{K_n^c}\frac{\alpha_{n,i,j}^*}{\sum_{l=1}^{P_n^c}\alpha_{n,l,j}^*} \tag{10}$$

where $\acute{\boldsymbol{x}}_{n,i}^c \in \acute{\mathbf{X}}_n^c$; P_n^c is the cardinality of $\mathbf{P}_n^c$; $\alpha_{n,i,j}^*=\frac{\alpha\left(\acute{\boldsymbol{x}}_{n,i}^c,\boldsymbol{x}_{n,j}^c\right)}{\sum_{l=1}^{K_n^c}\alpha\left(\acute{\boldsymbol{x}}_{n,i}^c,\boldsymbol{x}_{n,l}^c\right)}$; $i=1,2,\ldots,P_n^c$.

One can see from Eq. (9) that every soft prototype, $\boldsymbol{p}_{n,i}^c$ ($i=1,2,\ldots,P_n^c$) is a weighted combination of all data samples of $\mathbf{X}_n^c$, and the weights, $\alpha_{n,i,j}^*$ ($i=1,2,\ldots,P_n^c$; $j=1,2,\ldots,K_n^c$) are calculated based on the distances between data samples and the local peaks identified from $\mathbf{X}_n^c$, namely, $\acute{\mathbf{X}}_n^c$. By allowing all the data samples to be associated with every soft prototype, the identified soft prototypes can more effectively capture the underlying data structure and local patterns.

If $\mathbf{X}_n$ is the first data chunk (namely, $n=1$), the knowledge base in the form of soft prototypes ($\mathbf{S}^1,\mathbf{S}^2,\ldots,\mathbf{S}^C$) will be initialised as follows ($c=1,2,\ldots,C$):

$$\mathbf{S}^c \leftarrow \mathbf{P}_n^c \tag{11}$$

and the corresponding support, ν_i^c of $\boldsymbol{s}_i^c$ is initialised as $\nu_i^c=\rho_{n,i}^c$ ($i=1,2,\ldots,S^c$; $S^c=P_n^c$). The IF-THEN rule base (R^1, R^2,..., R^C) will be initialised with $\mathbf{S}^1,\mathbf{S}^2,\ldots,\mathbf{S}^C$.

Otherwise, namely, $n>1$, $\mathbf{P}_n^c$ will be used for expanding $\mathbf{S}^c$ ($c=1,2,\ldots,C$). To do so, the distances between soft prototypes of $\mathbf{S}^c$ and $\mathbf{P}_n^c$ will be calculated first, and Condition 2 is used for checking whether each newly identified soft prototype $\boldsymbol{p}_{n,i}^c \in \mathbf{P}_n^c$ is sufficiently distinctive from the soft prototypes identified from historical data chunks [36]:

$$Cond.\quad 2: \quad \begin{array}{c} if\left(\min_{\boldsymbol{s}\in\mathbf{S}^c}\left(\left\|\boldsymbol{p}_{n,i}^c-\boldsymbol{s}\right\|_1^2\right)>\gamma_G\right) \\ then\left(\mathbf{S}^c\leftarrow\mathbf{S}^c\cup\left\{\boldsymbol{p}_{n,i}^c\right\};\mathbf{P}_n^c\leftarrow\mathbf{P}_n^c\backslash\left\{\boldsymbol{p}_{n,i}^c\right\}\right) \end{array} \tag{12}$$

If $\boldsymbol{p}_{n,i}^c$ satisfies Condition 2, it represents a local data pattern that has not been observed from historical data and, therefore, $\boldsymbol{p}_{n,i}^c$ will join $\mathbf{S}^c$ from $\mathbf{P}_n^c$ ($S^c\leftarrow S^c+1$; $P_n^c\leftarrow P_n^c-1$) to preserve the new knowledge learned from the current data chunk $\mathbf{X}_n$ in the knowledge base.

After all the soft prototypes satisfying Condition 2 have joined $\mathbf{S}^c$, the remaining soft prototypes within $\mathbf{P}_n^c$ are used for updating $\mathbf{S}^c$ using Eq. (13) ($j=1,2,\ldots,S^c$) [36]:

$$\boldsymbol{s}_j^c\leftarrow\frac{\nu_j^c\boldsymbol{s}_j^c+\sum_{i=1}^{P_n^c}\alpha_{i,j}^*\rho_{n,i}^c\boldsymbol{p}_{n,i}^c}{\nu_j^c+\sum_{i=1}^{P_n^c}\alpha_{i,j}^*\rho_{n,i}^c};\quad \nu_j^c\leftarrow\nu_j^c+\sum_{i=1}^{P_n^c}\alpha_{i,j}^*\rho_{n,i}^c \tag{13}$$

where $\alpha_{i,j}^*=\frac{\alpha\left(\boldsymbol{p}_{n,i}^c,\boldsymbol{s}_j^c\right)}{\sum_{l=1}^{S^c}\alpha\left(\boldsymbol{p}_{n,i}^c,\boldsymbol{s}_l^c\right)}$. Eq. (13) indicates that each soft prototype in $\mathbf{S}^c$ is updated by all remaining soft prototypes in $\mathbf{P}_n^c$ with the corresponding weights calculated based on their mutual distances. The closer $\boldsymbol{s}_j^c$ is to $\boldsymbol{p}_{n,i}^c$, the bigger portion $\boldsymbol{s}_j^c$ receives from $\boldsymbol{p}_{n,i}^c$.

After $\mathbf{P}_n^c$ has been integrated into $\mathbf{S}^c$ ($\forall c$), the IF-THEN rule base (R^1, R^2,..., R^C) will be updated to reflect the latest changes in $\mathbf{S}^1,\mathbf{S}^2,\ldots,\mathbf{S}^C$ and the knowledge base updating process is completed.

However, due to the nonstationary nature of data streams, the areas of influence of some soft prototypes within $\mathbf{S}^1$, $\mathbf{S}^2$,..., $\mathbf{S}^C$ may gradually overlap with others due to the shifts/drifts of underlying patterns of data [40]. To attain a more compact and healthier knowledge base, SPAFIS will periodically examine all the soft prototypes and prune these overlapping soft prototypes to reduce redundant information in the knowledge base. After every θ_o data chunks have been processed ($\theta_o=10$ in this study), Condition 3 will be utilised to examine soft prototypes of $\mathbf{S}^c$ ($c=1,2,\ldots,C$) one by one from the oldest to the youngest and identify these with higher spatial similarity ($j=1,2,\ldots,S^c$) [36]:

$$Cond.\quad 3: \quad \begin{array}{c} if\left(\min_{i=j+1,2,\ldots,S^c}\left(\left\|\boldsymbol{s}_j^c-\boldsymbol{s}_i^c\right\|_1^2\right)<\omega_o\bullet\gamma_G\right) \\ then\left(\mathbf{R}^c\leftarrow\mathbf{R}^c\cup\left\{\boldsymbol{s}_j^c\right\};\mathbf{S}^c\leftarrow\mathbf{S}^c/\left\{\boldsymbol{s}_j^c\right\}\right) \end{array} \tag{14}$$

where ω_o ($\omega_o=0.001$ in this study) is a small non-negative value controlling the tolerance towards spatial similarity; $\mathbf{R}^c=\left\{\boldsymbol{r}_1^c,\boldsymbol{r}_2^c,\ldots,\boldsymbol{r}_{R^c}^c\right\}$ denotes the collection of older soft prototypes that are highly similar to the soft prototypes newly added to $\mathbf{S}^c$; R^c is the cardinality of $\mathbf{R}^c$; β_j^c is the support of $\boldsymbol{r}_j^c$. $\mathbf{R}^c$ will be utilised to update the remaining soft prototypes of $\mathbf{S}^c$. For each remaining soft prototype $\boldsymbol{s}_j^c \in \mathbf{S}^c$, it will be updated by $\mathbf{R}^c$ using Eq. (15), which is similar to Eq. (13).

$$\boldsymbol{s}_j^c\leftarrow\frac{\nu_j^c\boldsymbol{s}_j^c+\sum_{i=1}^{R^c}\alpha_{i,j}^{**}\beta_{n,i}^c\boldsymbol{r}_{n,i}^c}{\nu_j^c+\sum_{i=1}^{R^c}\alpha_{i,j}^{**}\beta_{n,i}^c};\quad \nu_j^c\leftarrow\nu_j^c+\sum_{i=1}^{R^c}\alpha_{i,j}^{**}\beta_{n,i}^c \tag{15}$$

where $j=1,2,\ldots,S^c$; $\alpha_{i,j}^{**}=\frac{\alpha\left(\boldsymbol{r}_i^c,\boldsymbol{s}_j^c\right)}{\sum_{l=1}^{S^c}\alpha\left(\boldsymbol{r}_i^c,\boldsymbol{s}_l^c\right)}$. Once the pruning process is completed, the IF-THEN rule base (R^1, R^2,..., R^C) will be updated accordingly with the updated soft prototypes in the cleaned knowledge base ($\mathbf{S}^1,\mathbf{S}^2,\ldots,\mathbf{S}^C$).

The system identification process of SPAFIS is summarised by the following diagram for better illustration. (Fig. 1)

2.2. Decision making scheme

In this study, the decision making scheme used by the original SPAFIS [36] is modified to better handle the class imbalance. Instead of involving all the identified soft prototypes in decision-making, for each unlabelled testing sample, $\boldsymbol{x}_i$, the confidence score of each individual IF-THEN rule R^c ($c=1,2,\ldots,C$) is calculated based on the spatial similarity between $\boldsymbol{x}_i$ and the k_o nearest soft prototypes associated with R^c:

$$\lambda^c(\boldsymbol{x}_i)=\frac{1}{k_o}\sum_{j=1}^{k_o}\nu_{j*}^c\alpha\left(\boldsymbol{x}_i,\boldsymbol{s}_{j*}^c\right) \tag{16}$$

where $\boldsymbol{s}_{j*}^c$ is the jth nearest soft prototype of the cth class; in this study, $k_o=9$ is used by default.

The class label of $\boldsymbol{x}_i$ is determined by the IF-THEN rule giving the highest confidence score by Eq. (17):

$$\widehat{y}_i=\underset{c=1,2,\ldots,C}{\mathrm{argmax}}\left(\lambda^c(\boldsymbol{x}_i)\right) \tag{17}$$

3. The proposed S^{3}PAFES

As aforementioned, S^{3}PAFES is an ensemble framework based on SPAFIS designed to learn from network data streams online in a semi-supervised, single-pass manner for intrusion detection. It overcomes the four key challenges faced by conventional AIDSs with the innovative

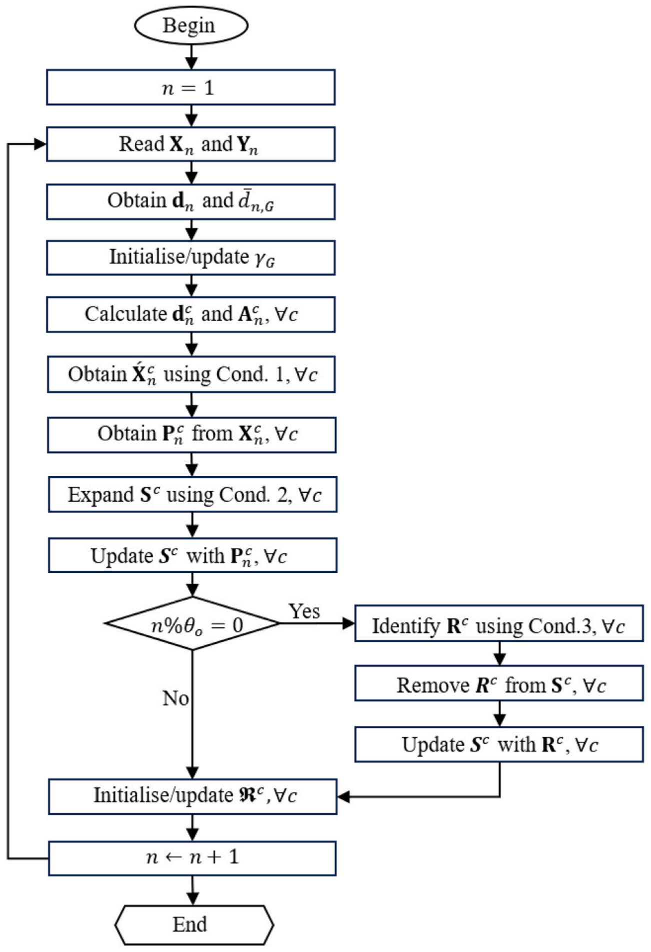

Begin
$n = 1$
Read $\mathbf{X}_n$ and $\mathbf{Y}_n$
Obtain $\mathbf{d}_n$ and $\bar{d}_{n,G}$
Initialise/update γ_G
Calculate $\mathbf{d}_n^c$ and $\mathbf{A}_n^c, \forall c$
Obtain $\acute{\mathbf{X}}_n^c$ using Cond. 1, $\forall c$
Obtain $\mathbf{P}_n^c$ from $\mathbf{X}_n^c, \forall c$
Expand $\mathbf{S}^c$ using Cond. 2, $\forall c$
Update $\boldsymbol{S}^c$ with $\mathbf{P}_n^c, \forall c$
$n\%\theta_o = 0$
Yes
No
Identify $\mathbf{R}^c$ using Cond.3, $\forall c$
Remove $\boldsymbol{R}^c$ from $\mathbf{S}^c, \forall c$
Update $\boldsymbol{S}^c$ with $\mathbf{R}^c, \forall c$
Initialise/update $\mathfrak{R}^c, \forall c$
$n \leftarrow n + 1$
End

Fig. 1. Diagram of system identification process of SPAFIS.

features including, 1) a two-step collaborative pseudo-labelling for semi-supervised learning from unlabelled data; 2) a flexible system architecture for online learning from data streams; 3) the combined use of oversampling and down sampling techniques for restoring class imbalance; and 4) a transparent, prototype-based structure with human-interpretable internal reasoning and decision-making. In addition, S^{3}PAFES also supports online active learning to tackle the unseen data patterns more effectively. By condensing hard-to-classify samples into a small set of unlabelled soft prototypes, human experts can choose to intervene at any point of the semi-supervised learning process by manually labelling only these condensed unlabelled soft prototypes, thereby greatly reducing the efforts of manual labelling. However, it is worth noting that human intervention is entirely optional.

In the rest of this section, an overview is firstly presented to introduce the general architecture and learning process of the proposed S^{3}PAFES. The supervised learning, semi-supervised learning, performance monitoring, active learning and decision-making schemes are detailed in subsections 3.2–3.6, respectively.

3.1. Overview

The general architecture of S^{3}PAFES is depicted in Fig. 2. It can be seen from this figure that the proposed S^{3}PAFES is composed of the following components: 1) one data distributor; 2) F base learners; 3) one challenging sample learner; 4) one joint pseudo-label generator; 5) one performance monitor, and; 6) one joint decision maker. The inner structure of a particular base learner is given by Fig. 3, where one can see that the core of the base learner is a SPAFIS model that learns from the input data streams on a chunk-by-chunk basis. To facilitate semi-supervised learning from unlabelled data samples, each SPAFIS model in the proposed ensemble framework is also equipped with 1) one labelled training data pool; 2) one validation data pool; 3) one pseudo-labelled data pool, and; 4) one nearest neighbour ensemble (NNE) classifier composed of multiple k-nearest neighbour (kNN) models. Note that the arrows in Figs. 2, 3 represent the directions of information flow. A single arrow indicates information passing from one module to another, while a double arrow indicates bidirectional information exchange between the two connected modules.

The online supervised learning, semi-supervised learning and active learning mechanisms of the proposed S^{3}PAFES are detailed in the following subsections. As aforementioned, this study primarily considers the infinite delay problems, where only a small amount of labelled training samples is available initially during the warming up stage for defining the classification problems, i.e., numbers of classes, feature spaces. Therefore, it is also assumed that only the class labels of the first L data chunks $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_L)$ are available, and all the other U data chunks $(\mathbf{X}_{L+1}, \mathbf{X}_{L+2}, \ldots, \mathbf{X}_{L+U})$ are unlabelled. The number of unlabelled training samples is significantly greater than the number of labelled

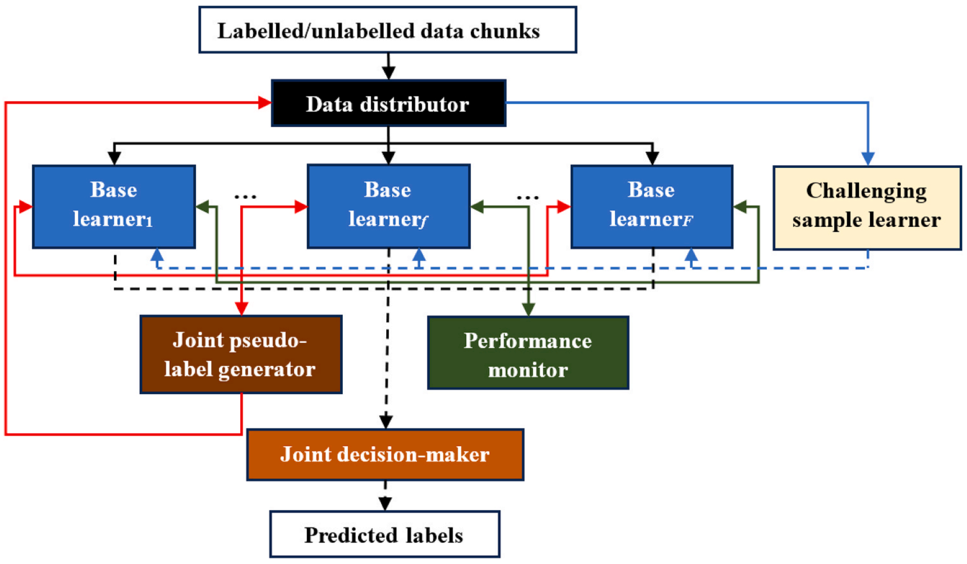


Fig. 2. General architecture of S^{3}PAFES.

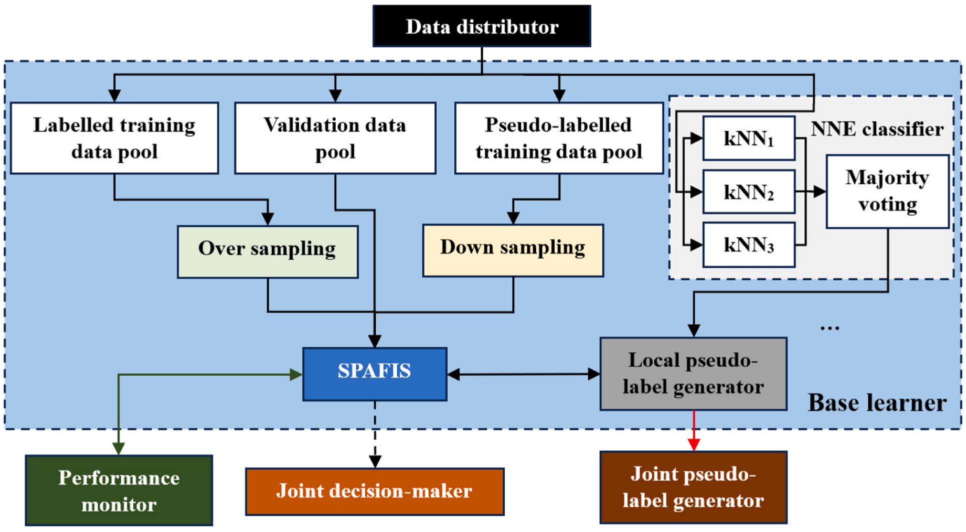


Fig. 3. Inner structure of a base learner.

Table 2
Additional key notations and definitions.

Notation	Definition
L	The number of data chunks with class labels
U	The number of data chunks without class labels
F	The number of base learners
H	The number of base learners each data sample is assigned to
$\mathbf{X}_{n,f}$	The subset of $\mathbf{X}_n$ assigned to the fth base learner
$\mathbf{Y}_{n,f}$	The class labels of $\mathbf{X}_{n,f}$
$K_{n,f}$	The cardinality of $\mathbf{X}_{n,f}$
$\mathbf{T}_f$	Labelled training data pool of the fth base learner
$\mathbf{V}_f$	Validation data pool of the fth base learner
τ_o	Splitting ratio between training and validation samples
$\sigma_f^c(\boldsymbol{x}_{n,i})$	The cth score produced by the fth NNE classifier on $\boldsymbol{x}_{n,i}$
$z_{f,n,i}$	The predicted class label of $\boldsymbol{x}_{n,i}$ by the fth NNE classifier
$\mathbf{X}_n^p$	The subset of $\mathbf{X}_n$ with highly confident predicted labels
$\mathbf{Z}_n$	The collection of pseudo labels of $\mathbf{X}_n^p$
$\delta_{n,i}^c$	The integrated confidence score of the cth class on $\boldsymbol{x}_{n,i}$
$\mathbf{Q}_f$	The pseudo-labelled training data pool of the fth base learner
α_f	The classifier weight of the fth base learner

training samples, namely, $\sum_{n=1}^{L} K_n \ll \sum_{n=L+1}^{L+U} K_n$. For simplification, it is further assumed that all the labelled and unlabelled data chunks are of the same size, namely, $K_n = K_o, \forall n = 1, 2, 3, \ldots, L+U$. For clarity, additional key notations and definitions used in this section are tabulated in Table 2.

S^{3}PAFES is designed to learn from network traffic data streams in infinite delay environments autonomously, where there is an infinite lack of supervision of the incoming data and only a small amount of labelled data exists for priming the system. Its overall learning process can be divided the following three stages, namely, supervised learning, semi-supervised learning and active learning. Note that active learning stage is optional.

3.1.1. Stage 1. Supervised learning

In this stage, S^{3}PAFES learns from labelled training data to prime its knowledge base. For each labelled data chunk, the data distributor divides it into F portions equally and sends each of them to one of the base learners. The F base learners initialise or update their knowledge bases using soft prototypes identified from the respective portions of the labelled data chunk, following the exact same algorithmic procedure described in Section 2.1. The detailed supervised learning process of S^{3}PAFES is described in Section 3.2.

3.1.2. Stage 2. Semi-supervised learning

After being primed with labelled training data in Stage 1, S^{3}PAFES continues to self-learn from unlabelled network data streams in a chunk-wise manner by exploiting pseudo-labelling without any human expert involvement. The processing cycle for each unlabelled data chunk can be further divided into three stages. The detailed semi-supervised learning process is presented in Section 3.3.

Stage 2.1. Two-step pseudo-labelling. In this substage, the data distributor firstly passes the unlabelled data chunk to all the base learners for making predictions. Coordinated by the joint pseudo-label generator, the SPAFIS models and the associated NNE classifiers make predictions on the class labels of the labelled data through a two-step process. The joint pseudo-label generator then collects all the predictions made by the base learners and selects out only these predictions with high confidence as pseudo labels.

Stage 2.2. Base model updating. In this substage, the pseudo labels produced in Stage 2.1 are then passed to the data distributor from the joint pseudo-label generator. For the pseudo-labelled samples, the data distributor divides them into F portions equally and sends them to the base learners. Then, the base learners update their knowledge bases using the revived data portions, similar as Stage 1.

Stage 2.3. Challenging sample learner updating. For these hard-to-classify data samples, namely, samples that are not pseudo-labelled in Stage 2.1, the data distributor assigns them to the challenging sample learner. The challenging sample learner will summarise these samples into a much smaller number of unlabelled soft prototypes. These unlabelled soft prototypes condense the novel information of unseen data patterns to facilitate the involvement of human experts in Stage 3 for active learning.

At the end of each learning cycle, the performance monitor monitors the prediction performances of individual base learners. The base learners with poor out-of-sample prediction performance will be pruned promptly and replaced by new base learners initialised with stored historical data to maintain the high-level performance of S^{3}PAFES. The performance monitoring process is detailed in Section 3.4.

3.1.3. Stage 3. Active learning (optional)

At any point of the semi-supervised learning process, human experts can optionally be involved and provide the expert knowledge to contribute to the continuous self-improvement of S^{3}PAFES by assigning class labels to these unlabelled soft prototypes stored in the knowledge base of the challenging sample learner manually. Then, these manually soft prototypes are fused into the corresponding IF-THEN rules of the F SPAFIS models and removed from the knowledge base of the challenging sample learner. The detailed active learning process is described in Section 3.5.

However, it is worth noting that the involvement of human experts in the learning process of S^{3}PAFES is entirely optional. After being primed with labelled data, S^{3}PAFES is able to perform self-learning and self-improving from unlabelled data streams in a fully autonomous manner. Upon request from users, S^{3}PAFES can be used to predict the class labels of unlabelled data following the decision-making process detailed in Section 3.6.

The flowchart of the learning process of S^{3}PAFES is given in Fig. 4 for better illustration.

3.2. Supervised learning scheme

During the supervised learning stage, S^{3}PAFES will learn from labelled data chunks to prime its knowledge base. Once a new data chunk $\mathbf{X}_n$ with the corresponding class labels $\mathbf{Y}_n$ ($n = 1, 2, \ldots, L$) becomes available, the data distributor will assign each individual data sample within $\mathbf{X}_n$ to H ($H = 2$ by default) base learners randomly such that every base learner will receive a different subset of $\mathbf{X}_n$ with some samples shared with others. All the base learners then use the receive data to initialise/update their system structure and meta-parameters following the same process given below.

After the fth base learner has received its own portion of the current labelled training data chunk, denoted as $\mathbf{X}_{n,f}$ and $\mathbf{Y}_{n,f}$, it will randomly divide the received data and labels into two parts, namely, $\mathbf{T}_{n,f} = \left\{\mathbf{X}^t_{n,f}, \mathbf{Y}^t_{n,f}\right\}$ and $\mathbf{V}_{n,f} = \left\{\mathbf{X}^v_{n,f}, \mathbf{Y}^v_{n,f}\right\}$ with the sizes of $\tau_o K_{n,f}$ and $(1 - \tau_o)\ K_{n,f}$, respectively, where τ_o denotes the splitting ratio ($\tau_o = 0.9$ by default) and; $K_{n,f}$ is the cardinality of $\mathbf{X}_{n,f}$ and there is $K_{n,f} = \frac{H}{F}K_o$.

If $\mathbf{X}_n$ is the very first data chunk S^{3}PAFES receives, namely, $n = 1$, the labelled training data pool, $\mathbf{T}_f$ and the validation data pool, $\mathbf{V}_f$ of the fth base learner are initialised using Eq. (18).

$$\mathbf{T}_f \leftarrow \mathbf{T}_{n,f}; \quad \mathbf{V}_f \leftarrow \mathbf{V}_{n,f} \tag{18}$$

Otherwise, the labelled training data pool, $\mathbf{T}_f$ and the validation data pool, $\mathbf{V}_f$ will be expanded by $\mathbf{T}_{n,f}$ and $\mathbf{V}_{n,f}$, respectively:

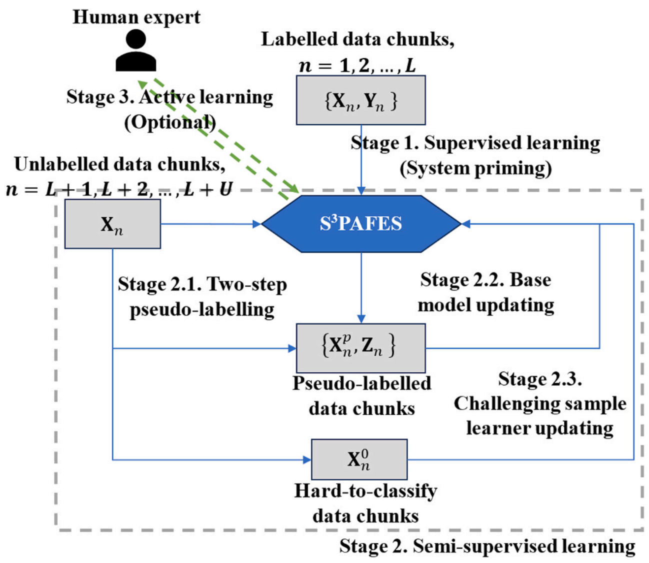
$$\mathbf{T}_f \leftarrow \mathbf{T}_f \cup \mathbf{T}_{n,f}; \quad \mathbf{V}_f \leftarrow \mathbf{V}_f \cup \mathbf{V}_{n,f} \tag{19}$$

If $\mathbf{T}_f$ or $\mathbf{V}_f$ exceeds the maximum size, the data stored in the oversized pool will be randomly sampled to reduce the pool size to the maximum size. In the proposed framework, the maximum sizes of the three data pools, namely, the labelled training data pool, $\mathbf{T}_f$, the validation data pool, $\mathbf{V}_f$ and the pseudo-labelled training data pool, $\mathbf{Q}_f$ are set to be uniformly the same as $L_o = \frac{H}{F}K_o$, which is equal to the amount of data it receives from each labelled data chunk.

Then, $\mathbf{T}_{n,f}$ is augmented using oversampling to balance the class distribution by synthetically creating data samples of minority class. The augmented $\mathbf{T}_{n,f}$, denoted as $\widehat{\mathbf{T}}_{n,f}$ is used for initialising SPAFIS if $n = 1$ or updating SPAFIS if $n > 1$ following the algorithmic procedure described in Section 2.1. In the proposed framework, the synthetic minority oversampling technique (SMOTE) is employed for data augmentation [41]. The main reason for using SMOTE is due to its effectiveness in addressing the challenges posed by class imbalance. However, one may consider alternative oversampling techniques to replace SMOTE, such as k-means SMOTE [42], kNN-based synthetic minority oversampling [43], etc.

In parallel, a new kNN model is trained with $\mathbf{X}_{n,f}$ and $\mathbf{Y}_{n,f}$, and added to the NNE classifier. The value of $k_{n,f}$ is randomly selected from the range of [1,15] for each individual kNN model added to it, thereby further boosting the diversity within the NNE classifier. This also avoids the difficulties in predetermining the value of $k_{n,f}$ for the new kNN model due to the lack of prior knowledge. The NNE classifier will be used for predicting the class labels of unlabelled training samples during the semi-supervised learning stage. The main reason for utilising an ensemble of kNN models for this task is that kNN is a computationally efficiency, reliable and explainable classification model for small datasets, but cannot scale up well for large datasets. Hence, by creating one kNN model per data chunk and building an ensemble, the strengths of kNN are maximised and the limitations are mitigated. However, it is worth noting that the ensemble of kNN models is only involved in the process of pseudo-label generation because they rely on labelled training data for making predictions and are unable to handle the potential shifts and drifts in the unlabelled streaming data.

After the SPAFIS models and the associated NNE classifiers have been initialised/updated with $\mathbf{X}_n$, the current supervised learning cycle is completed and S^{3}PAFES is ready for processing the next data chunk ($n \leftarrow n + 1$). After all the labelled data chunks have been processed, S^{3}PAFES is primed and ready for making predictions on unlabelled data directly following the procedure described in Section 3.6. Alternatively, it can continue to learn from unlabelled data to self-improve its knowledge base for greater prediction performance following the procedure described in Section 3.3 and, optionally, involve human experts during the semi-supervised learning process to provide additional support to further sharpen its knowledge base and classification boundaries.



Human expert
Labelled data chunks, $n = 1, 2, \ldots, L$
$\{\mathbf{X}_n, \mathbf{Y}_n\}$
Stage 3. Active learning (Optional)
Stage 1. Supervised learning (System priming)
Unlabelled data chunks, $n = L+1, L+2, \ldots, L+U$
$\mathbf{X}_n$
S^{3}PAFES
Stage 2.1. Two-step pseudo-labelling
Stage 2.2. Base model updating
$\{\mathbf{X}^p_n, \mathbf{Z}_n\}$
Pseudo-labelled data chunks
Stage 2.3. Challenging sample learner updating
$\mathbf{X}^0_n$
Hard-to-classify data chunks
Stage 2. Semi-supervised learning

Fig. 4. Flowchart of the overall learning process of S^{3}PAFES.

3.3. Semi-supervised learning scheme

3.3.1. Two-step pseudo-labelling

In this stage, given a new unlabelled data chunk, $\mathbf{X}_n$ ($n = L+1, L+2, \ldots, U+L$), S^{3}PAFES will select these data samples with the highest classification confidence from the current data chunk $\mathbf{X}_n$ for expanding its knowledge base through a two-step collaborative pseudo-labelling process.

Firstly, all the F associated NNE classifiers within the ensemble framework will work collaboratively to predict the class labels of the unlabelled data samples. Every NNE classifier will produce C scores (one per class) for each individual data sample of $\mathbf{X}_n$ to determine its class label. The scores are calculated based on the class distributions of the nearest neighbours. The score corresponding to the cth class produced by the fth NNE classifier on a particular data sample, $\boldsymbol{x}_{n,i} \in \mathbf{X}_n$ is calculated by Eq. (20):

$$\sigma^c_f(\boldsymbol{x}_{n,i}) = \frac{1}{L}\sum_{n=1}^{L}\frac{k^c_{n,f}}{k_{n,f}} \tag{20}$$

where $k^c_{n,f}$ is the number of nearest neighbours that belong to the cth class among the $k_{n,f}$ nearest neighbours of $\boldsymbol{x}_{n,i}$ identified by the nth kNN model of the fth NNE classifier. The predicted class label of $\boldsymbol{x}_{n,i}$ is predicted using the "winner takes all" principle:

$$z_{n,f,i} = \underset{c=1,2,\dots,C}{\operatorname{argmax}}\left(\sigma_f^c(\boldsymbol{x}_{n,i})\right) \tag{21}$$

where $z_{f,n,i}$ is the predicted class label of $\boldsymbol{x}_{n,i}$ by the fth NNE classifier.

The joint pseudo-label generator will collect the predicted class labels produced by the F NNE classifiers and determine the pseudo label of $\boldsymbol{x}_{n,i}$ by majority voting. The pseudo label of $\boldsymbol{x}_{n,i}$, denoted as $z_{n,i}$ is corresponding to the class label that the majority of the F NNE classifiers agree on.

Then, Condition 4 is utilised to examine whether all the NNE classifiers agree on the class label of $\boldsymbol{x}_{n,i}$.

$$Cond.\quad 4: \quad \begin{array}{c} if\left(z_{n,j,i} = z_{n,i}, \quad \forall j = 1, 2\dots, F\right) \\ then\left(\mathbf{X}_n^p \leftarrow \mathbf{X}_n^p \cup \{\boldsymbol{x}_{n,i}\}; \mathbf{Z}_n \leftarrow \mathbf{Z}_n \cup \{z_{n,i}\}; \mathbf{X}_n \leftarrow \mathbf{X}_n / \{\boldsymbol{x}_{n,i}\}\right) \end{array} \tag{22}$$

where $\mathbf{X}_n^p$ denotes the subset of $\mathbf{X}_n$ containing data samples with predicted class labels of high confidence; $\mathbf{Z}_n$ denotes the collection of pseudo labels of $\mathbf{X}_n^p$.

If Condition 4 is satisfied, it suggests that all the NNE classifiers agree with each other on the class label of $\boldsymbol{x}_{n,i}$. Hence, S^{3}PAFES can trust the prediction, and $\boldsymbol{x}_{n,i}$ is used for expanding the knowledge base by treating the pseudo label $z_{n,i}$ as the ground truth. Otherwise, namely, Condition 4 is not met, it suggests that there are some contradictions between the predictions made by individual NNE classifiers. The existence of contradictions does not necessarily indicate that the pseudo label $z_{n,i}$ is wrong. The disagreement could be caused by the gradual drifts of underlying data patterns in the data stream since the NNE classifiers only have the knowledge learned from labelled training data presented to them, which could be outdated. Therefore, further examination is needed for these unlabelled data samples that fail to satisfy Condition 4.

After the most confident predictions have been selected by Condition 4 and joined $\mathbf{X}_n^p$, the second stage of pseudo-labelling begins by involving the SPAFIS models in this process to jointly determine the class labels of these samples that do not meet Condition 4. For every unlabelled training sample $\boldsymbol{x}_{n,i}$ remaining in $\mathbf{X}_n$, namely, $\boldsymbol{x}_{n,i} \in \mathbf{X}_n \wedge \boldsymbol{x}_{n,i} \notin \mathbf{X}_n^p$, the F SPAFIS models will jointly determine its class label, $\hat{y}_{i,n}$ following the decision-making process described in Section 3.6. Then, Condition 5 is used to examine whether $\boldsymbol{x}_{n,i}$ can be used for updating the knowledge base of S^{3}PAFES:

$$Cond.\quad 5: \quad \begin{array}{c} if\left(\hat{y}_{i,n} = z_{n,i}\right) and\left(\delta_{n,i}^{\hat{y}_{i,n}} \geq \delta_o\right) \\ then\left(\mathbf{X}_n^p \leftarrow \mathbf{X}_n^p \cup \{\boldsymbol{x}_{n,i}\}; \mathbf{Z}_n \leftarrow \mathbf{Z}_n \cup \{z_{n,i}\}; \mathbf{X}_n \leftarrow \mathbf{X}_n / \{\boldsymbol{x}_{n,i}\}\right) \end{array} \tag{23}$$

where $\delta_{n,i}^{\hat{y}_{i,n}}$ is the integrated confidence score of the $\hat{y}_{i,n}$ class; δ_o is the confidence threshold and $\delta_o = 0.9$ is used in this study. Condition 5 suggests that if the F SPAFIS models also agree with the NNE classifiers on the pseudo label of $\boldsymbol{x}_{n,i}$ with high confidence, the pseudo label $z_{n,i}$ is reliable enough and $\boldsymbol{x}_{n,i}$ should join $\mathbf{X}_n^p$ to be used for updating S^{3}PAFES.

The rationales behind the two-step collaborative pseudo-labelling process using Conditions 4 and 5 are as follows. The unlabelled data samples satisfying Condition 4 represent similar data patterns that have been observed from the labelled training data and, therefore, the predicted class labels can be directly used as the ground truth. If unlabelled data samples meet Condition 5 but not Condition 4, it suggests that concept drifts may have occurred and the data patterns have evolved gradually, but S^{3}PAFES is still confident about these predictions, these data samples can be used for expanding the knowledge base reliably. The two-step collaborative pseudo-labelling process helps S^{3}PAFES to select out these unlabelled data with the most confident pseudo labels for self-expanding the knowledge base. The effectiveness and validity of the two-step pseudo-labelling scheme designed for S^{3}PAFES are justified by the ablation analysis presented in Section 4.5.

Although these unlabelled data samples that fail to satisfy both conditions might be more informative because they either represent data patterns unseen before or are located at the boundaries of two different classes, S^{3}PAFES will not use them for updating the knowledge base due to the high uncertainties. The predicted labels of these difficult-to-classify samples are more likely to be wrong without additional assistance from human experts and involving these data samples directly in system updating may significantly deteriorate the prediction performance of S^{3}PAFES due to the accumulation of pseudo-labelling errors in the knowledge base.

3.3.2. *Base model updating*

After all the data samples that satisfy either Condition 4 or Condition 5 have been selected out and joined $\mathbf{X}_n^p$ together with the corresponding pseudo labels $\mathbf{Z}_n$, the data distributor will randomly distribute every pseudo-labelled data sample to H base learners in the same manner as the labelled data samples, and every base learner will receive a different subset of $\mathbf{X}_n^p$ and $\mathbf{Z}_n$.

After the fth base learner receives the corresponding set of pseudo-labelled data, denoted as $\mathbf{Q}_{n,f} = \left\{\mathbf{X}_{n,f}^p, \mathbf{Z}_{n,f}\right\}$, the pseudo-labelled training data pool, $\mathbf{Q}_f$ will be initialised if it is the very first pseudo-labelled data chunk, namely, $n = L + 1$, or expanded if $n > L + 1$.

$$\begin{cases} \mathbf{Q}_f \leftarrow \mathbf{Q}_{n,f}, & if\ n = L + 1 \\ \mathbf{Q}_f \leftarrow \mathbf{Q}_f \cup \mathbf{Q}_{n,f}, & if\ n > L + 1 \end{cases} \tag{24}$$

If the pseudo-labelled training data pool $\mathbf{Q}_f$ exceeds its maximum capacity, random down sampling is performed to reduce its pool size to L_o. Then, random down sampling is performed on $\mathbf{Q}_{n,f}$ to reduce the number of data samples of the majority class and restore the class balance. The downsized $\mathbf{Q}_{n,f}$, denoted as $\widehat{\mathbf{Q}}_{n,f}$ is then used for updating the fth SPAFIS model in the same way as described in Section 2.1 by treating the pseudo labels as the ground truth. After all the SPAFIS models have been updated, a new semi-supervised learning cycle starts with the next unlabelled data chunk available ($n \leftarrow n + 1$).

It is worth noting that, unlike the labelled training data on which oversampling is applied to maximise the information mined from data, there are two main reasons for applying random down sampling to the pseudo-labelled data. The first is that, in a typical infinite delay problem, unlabelled training samples are abundant, and class imbalance can also be expected in the unlabelled training data, which inevitably causes the pseudo-labelled data to be imbalanced. Using the imbalanced pseudo-labelled data for updating S^{3}PAFES will cause its predictive performance to deteriorate due to model bias. In addition, due to the nonstationary nature of the data streams, pseudo-labelling errors cannot be avoided practically. Applying oversampling techniques to enhance the minority class will introduce more labelling errors to the pseudo-labelled data, and can have a unfavourable impact on the performance of S^{3}PAFES. Therefore, in S^{3}PAFES, two different sampling techniques are utilised to facilitate semi-supervised learning from imbalanced data.

3.3.3. *Challenging sample learner updating*

The remaining data samples within $\mathbf{X}_n$ after the two-step collaborative pseudo-labelling process, denoted as $\mathbf{X}_n^0$ are difficult to classify compared with others because S^{3}PAFES is less confident on their class labels. Instead of discarding them, the data distributor will pass these hard-to-classify samples to the challenging sample learner with the corresponding class labels set as 0, effectively a "reject" class. The challenging sample learner is a single-class SPAFIS model that only learns from these difficult-to-classify samples following the same procedure described in Section 2.1. However, different from other SPAFIS models in the proposed ensemble framework, it has only one single soft prototype-based IF-THEN rule in its knowledge base in the form of Eq. (25):

$$R^0: \begin{array}{c}\text{IF}(\boldsymbol{x} \sim \boldsymbol{s}_1^0)\text{OR}(\boldsymbol{x} \sim \boldsymbol{s}_2^0)\text{OR}\ldots\text{OR}(\boldsymbol{x} \sim \boldsymbol{s}_{S^0}^0)\\ \text{THEN}(y = 0)\end{array} \tag{25}$$

The challenging sample learner will be operating in parallel with the F base learners during the semi-supervised learning process and summarises these challenging unlabelled training samples from the received unlabelled data chunks into a reduced set of highly informative soft prototypes. These unlabelled soft prototypes either represent unseen unique data patterns or locate near the classification boundaries, and can be used for greatly enhancing the knowledge base of S^{3}PAFES with the assistance of human experts.

3.4. Performance monitoring scheme

These SPAFIS models sitting at the heart of base learners will not receive the same labelled and pseudo-labelled data for training. As a result, one can expect that the prediction performances of these SPAFIS models may vary. However, if one (or more) of the F SPAFIS models perform significantly worse than others, the overall prediction performance of S^{3}PAFES would also deteriorate. To maintain the prediction performances of individual base learners and construct a stronger ensemble classifier, the performance monitor will identify these SPAFIS models that perform poorly on classifying the out-of-sample data stored in the associated validation data pools and replace them with new ones [44].

To do so, at the end of each learning cycle (assuming the nth one) after the nth labelled/unlabelled training data chunk has been processed, the performance of fth SPAFIS model is evaluated on the data samples stored in the associated validation pool, $\mathbf{V}_f$ by making predictions on their class labels. The out-of-sample balanced classification accuracy is then calculated, denoted as $a_{n,f}$, and the weight of the fth SPAFIS is initialised as $a_f \leftarrow a_{n,f}$ if this is the first data chunk it has processed. Otherwise, a_f is updated using Eq. (26).

$$a_f \leftarrow \frac{1}{2}\left(a_f + a_{n,f}\right) \tag{26}$$

Based on the classifier weights, a_f ($f = 1, 2, \ldots, F$), Condition 6 is utilised to identify these SPAFIS models with weaker prediction performances.

$$\textit{Cond.}\quad 6: \begin{array}{c}\textit{if}\ (a_f < \overline{a} - \Delta_a)\\ \textit{then}\ (\text{the } f\text{th SPAFIS model is pruned})\end{array} \tag{27}$$

where a_f denotes the classifier weight of the fth SPAFIS model; $\overline{a}$ is mean of the classifier weights of all the SPAFIS models within the ensemble framework $\overline{a} = \frac{1}{F}\sum_{f=1}^{F} a_f$, and Δ_a is the corresponding standard deviation.

Condition 6 is based on the one sigma rule, and the rationale behind is that if the balanced classification accuracy of the fth SPAFIS model is one standard deviation lower than the average, it can be concluded that this base model is performing worse than the majority, and therefore, this base model needs to be pruned promptly. During the supervised and semi-supervised learning processes, Condition 6 consistently monitors the quality of the learned SPAFIS models to help S^{3}PAFES maintain high-level classification performance. Once a particular SPAFIS model (assuming the fth one) is pruned from S^{3}PAFES, a new SPAFIS model is initialised to fill in this empty space using the labelled data stored in the fth labelled training data pool, $\mathbf{T}_f$ and another randomly selected training data pool $\mathbf{T}_j$ ($\forall j, j \neq f$). If the fth SPAFIS model is pruned during the semi-supervised learning process, the pseudo-labelled data in the fth pseudo-labelled training data pool $\mathbf{Q}_f$ and another randomly selected pseudo-labelled training data pool $\mathbf{Q}_i$ ($\forall i, i \neq f$) will be used to continue updating the newly initialised SPAFIS model after being primed by $\mathbf{T}_f$ and $\mathbf{T}_j$. The utilisation of the stored labelled and pseudo-labelled training data can effectively help the new SPAFIS model to rapidly capture the latest data patterns whilst having a good understanding about the historical patterns [44].

After the new fth SPAFIS model has been primed with historical data, its out-of-sample prediction performance is evaluated on the corresponding validation data pool, $\mathbf{V}_f$ and a_f is initialised as $a_{n,f}$ ($a_f \leftarrow a_{n,f}$). The pruning process at the current supervised/semi-supervised learning cycle is completed after all the SPAFIS models satisfying Condition 6 have been pruned and replaced with new ones, thereby maintaining the same number of base models within the ensemble framework throughout the learning process.

3.5. Active learning scheme

Unlike other SPAFIS models in the proposed ensemble framework, the challenging sample learner will not participate in the decision-making process. However, human experts can get involved at any point of the semi-supervised learning process and assign the class labels to these soft prototypes learned from challenging data samples manually. Comparing with labelling all the challenging data samples, assigning class labels to unlabelled soft prototypes is a much simpler task because the number of soft prototypes is typically far less than the amount of raw data samples. After manual labelling, these soft prototypes with class labels assigned by human experts will be merged into the corresponding IF-THEN rules of the F SPAFIS models and removed from the knowledge base of the challenging sample learner. Note that, to enhance the diversity between the base learners and make the best use of the additional assistance provided by experts, every manually labelled soft prototype will only be assigned to half of the base learners randomly. It is demonstrated by numerical examples presented in Section 4.3 that S^{3}PAFES can be further enhanced by augmenting the knowledge base with the manually labelled soft prototypes from the “reject” class. Note that the involvement of human experts is entirely optional because S^{3}PAFES is designed to learn from both labelled and unlabelled data in a fully autonomous manner.

3.6. Decision making scheme

Given a particular unlabelled data sample, $\boldsymbol{x}_i$, its class label is determined by all the SPAFIS models jointly in the ensemble framework. Each individual SPAFIS (assuming the fth one; $f = 1, 2, \ldots, F$) will produce C confidence scores, denoted as $\lambda_f^c(\boldsymbol{x}_i)$ ($c = 1, 2, \ldots, C$) following the process described in Section 2.2. The global decision-maker will then predict the class label of $\boldsymbol{x}_i$ based on the confidence scores produced by the F SPAFIS models using Eq. (28):

$$\widehat{y}_i = \underset{c=1,2,\ldots,C}{\text{argmax}}\left(\delta_i^c\right) \tag{28}$$

where $\delta_i^c = \sum_{f=1}^{F} a_f \bullet \frac{\lambda_f^c(\boldsymbol{x}_i)}{\sum_{j=1}^{C}\lambda_f^j(\boldsymbol{x}_i)}$ is the integrated confidence score of the cth class; the denominator $\sum_{j=1}^{C}\lambda_f^j(\boldsymbol{x}_{n,i})$ $\forall f$ is utilised for the purpose of normalisation such that the confidence scores of different SPAFISs are on the same scale; a_f is the classifier weight of the fth SPAFIS model, which will be detailed in Section 3.5 later.

4. Experimental investigation

In this section, numerical examples based on a range of popular benchmark datasets are presented to demonstrate the efficacy of the proposed S^{3}PAFES for network intrusion detection from data streams. The source code of S^{3}PAFES is available at: https://github.com/Gu-X/Semi-Supervised-Soft-Prototype-based-Autonomous-Fuzzy-Ensemble-System.

4.1. Configuration

A. Data description. In this study, the following four commonly used benchmark datasets for network intrusion detection are employed for performance demonstration, namely [3]:

1) UNSWNB15 [45]: This dataset is a modern benchmark created by researchers at the Australian Centre for Cyber Security at the University of New South Wales using the IXIA PerfectStorm platform. This dataset combines real regular activities with synthetic attack behaviours, hence, providing a better representation of contemporary traffic patterns. UNSWNB15 has one training set and one testing set.
2) CICIDS2017 [46]: CICIDS2017 is a modern flow-based network intrusion detection dataset published by Canadian Institute for Cyber Security. This dataset is consisted of both benign and malicious traffic, where the benign traffic is generated by a system named B-Profiles and the malicious traffic is generated by executing existing attack tools at specific time windows.
3) CICIDS2018 [46]: CICIDS2018 is a successor of CICIDS2017 published by Canadian Institute for Cyber Security in collaboration with Communication Security Establishment. CICIDS2018 is generated with the same tool used for creating CICIDS2017 but deployed on the Amazon Web Services cloud computing platform.
4) HIKARI2021 [47]: HIKARI2021 is one of the latest benchmark datasets built by researchers at Keio University, Japan. This dataset contains a mix of encrypted synthetic attacks and benign real traffic, reflecting the up-to-date landscape of cyber-attacks.

Key information of the four datasets is summarised in Table 3, where one can see that, although all four datasets are imbalanced, there is a notably higher level of class imbalance in the HIKARI2021 dataset with over 92.7 % of data belonging to the normal class, distinguishing it from the other three datasets.

Following the common practice [14], UNSWNB15 has been pre-processed by converting the categorical attributes to numerical ones via one-hot encoding. For CICIDS2017 and CICIDS2018 datasets, attributes with redundant information have been dropped, and data samples with missing or infinity values have been removed. To prevent overfitting, attributes directly related to data generation setup, such as source IP address, source port, destination IP address and destination port are removed from CICIDS2017, CICIDS2018 and HIKARI2021 datasets a priori, following [2]. The value ranges of all attributes in the four datasets are further standardised to eliminate the influence of the measurement unit on the model training [45]. As the sizes of CICIDS2017 and CICIDS2018 datasets are significantly larger than the UNSWNB15 and HIKARI2021 datasets, to facilitate simulation, 10 % and 2 % of the data samples are randomly selected from the two datasets, respectively, to create the smaller datasets for running numerical experiments.

Table 3
Key information of benchmark datasets for intrusion detection.

Dataset		# Samples			# Attributes
		Total	Normal	Attack	
UNSWNB15	Training Testing	175,341 82,332	56,000 37,000	119,341 45,332	40 numerical ones + 3 categorical ones + 1 class label
CICIDS2017		2830,743	2273,097	557,646	78 numerical ones + 1 class label
CICIDS2018		16,232,944	13,484,708	2748,236	79 numerical ones + 1 class label
HIKARI2021		555,278	517,582	37,696	83 numerical ones + 1 class label

B. Parameter setting for S^3PAFES. The proposed S^3PAFES has three key externally controlled parameters to be specified by users, which include 1) ensemble scale, F; 2) level of granularity, G, and; 3) confidence threshold, δ_o (used in Condition 5). In this study, the recommended setting for the three parameters, F, G and δ_o are as follows: $F = 10$, $G = 9$ and $\delta_o = 0.9$. Unless specifically declared otherwise, experimental results reported in this section are obtained using the recommended setting by default.

An important feature of S^3PAFES is its capability of online semi-supervised learning from network data streams on a chunk-by-chunk basis. Although the sizes of different labelled/unlabelled data chunks can vary significantly in practice and are relevant to the nature of the data, all the data chunks are assumed to be of the same size, namely, $K_n = K_o, \forall n = 1, 2, 3, \ldots, L + U$ in this study for simplification with $K_o = 10000$. If the number of remaining labelled/unlabelled data samples are less than K_o, all the remaining samples will be included in the final labelled/unlabelled data chunk for supervised/semi-supervised learning.

To better understand the impacts of the externally controlled parameters, F, G, δ_o and K_o, a sensitivity analysis is carried out and presented in Section 4.2.

C. Comparative algorithms and parameter settings. Since this study is focused on intrusion detection from network data streams with infinite delayed class labels, only inductive semi-supervised learning algorithms are involved in the numerical examples for performance evaluation. The SOTA inductive algorithms for comparison include:

1) Tri-training classifier (TriT) [25,48];
2) Transductive minimax probability machine (TMPM) [49];
3) Online reliable semi-supervised learning classifier (ORSSL) [30];
4) Online semi-supervised learning vector quantization (OSSLVQ) [50];
5) Semi-supervised ensemble classifier using performance-based selection metric (SSEPBS) [51];
6) Semi-supervised ensemble classifier using confidence-based selection metric (SSECBS) [51];
7) Self-training hierarchical prototype-based classifier (STHP) [39];
8) Semi-supervised self-organising fuzzy inference system (S^3OFIS) [37], and;
9) Dual-model semi-supervised self-organising fuzzy inference system (DMS^3OF) [52].

Note that, among the nine comparative algorithms, TriT and TMPM require training to be performed offline, the other seven comparative algorithms are designed for semi-supervised learning from data streams, same as the proposed ensemble model.

In running the experiments, TriT employs linear SVM as its base classifier. The box constraint of the linear SVM is set as 1 and scale factor is determined automatically using a heuristic procedure. For TMPM, the values of two externally controlled parameters, λ and ρ are selected from the candidate set $[10^{-4}, 10^{-3}, \ldots, 10^4]$ as recommended by [49]. 20 % of the labelled training data are randomly selected and used as the validation data to help TMPM identify the best values of λ and ρ. The recommended parameter setting given by [30] are used for ORSSL, namely, maxMC = 1000, $\Theta = 4$, $m = 100$, $\kappa = 50$ and $\lambda = 2\times 10^{-6}$. For OSSLVQ, the unlabelled training samples are used for updating the model under the Gaussian mixture model assumption, and the number of prototypes per class is set as $K = 9$ [50]. SSEPBS and SSECBS employ incremental SVM with linear kernel as the base classifier with the box constraint and scale factor both set to be 1. The chunk size, D is set as

$D = 20000$. Other externally controlled parameters are set as $K = 10$, $N = 5$, $\alpha_1 = 0.75$, $\alpha_2 = 0.25$, $\epsilon = 0.0001$ and $T = 0.9$, following the recommended setting given by [51]. STHP, S^{3}OFIS and DMS^{3}OF use the recommended parameter settings given by [37,39,52], respectively.

D. Performance measures. For performance evaluation, three standard criteria for classification are employed, namely, 1) accuracy (*acc*); 2) balanced accuracy (*bacc*) [53], and; 3) Matthew's correlation coefficient (*mcc*) [54]. Note that *bacc* and *mcc* are commonly used performance metrics for imbalanced classification problems., For fair comparison, all the reported results in this study are obtained as the average of 10 Monte Carlo experiments by default to allow a certain degree of randomness.

The proposed S^{3}PAFES was implemented using Python 3.9. The performance evaluation was conducted on a laptop with i7–12700H processor, 64 GB RAM and RTX 3050 Ti GPU.

4.2. Sensitivity analysis

In this section, sensitivity analysis is carried out based on the CICIDS2017 dataset to evaluate the influence of the four externally controlled parameters, F, G, δ_o and K_o on S^{3}PAFES's classification performance. In running the experiments, 5 % of the data samples are randomly selected to create the labelled training set, 55 % of the data samples are used as the unlabelled training set, and the remaining 40 % of the data is used as the testing set to evaluate the out-of-sample classification performance of S^{3}PAFES after semi-supervised learning from both labelled and unlabelled training data.

Firstly, the influence of the ensemble scale, F on the prediction performance of S^{3}PAFES is investigated. The ensemble scale F controls the number of base classifiers in the ensemble framework. In this example, the value of F is varied from 3 to 25 using irregular intervals, and the values of the other three parameters are set as $G = 9$, $\delta_o = 0.9$ and $K_o = 10000$. The out-of-sample classification performance of S^{3}PAFES on the testing data with different values of F are reported in Fig. 5 in terms of *acc*, *bacc* and *mcc*. The average number of soft prototypes identified by the F individual SPAFIS models within the ensemble framework, denoted as S^E, the number of soft prototypes identified from hard-to-classify unlabelled training samples by the challenging sample learner, denoted as S^C, and the time consumption of the overall learning process without parallelisation (in seconds), denoted as t_{exe} are also reported in Fig. 5. Note that the soft prototypes learned by the challenging sample learner will not participate in the decision-making until they have been manually examined by human experts and assigned with class labels.

One can see from Fig. 5 that a greater F allows S^{3}PAFES to have more base models that process the data streams and share the computational load, enabling S^{3}PAFES to achieve higher overall computational efficiency. However, if F is too large, individual base models might not be able to receive sufficient training data from the data distributor for system structure learning and parameter optimisation, affecting the prediction performance of S^{3}PAFES. In addition, the base models are also more likely to disagree with each other on the pseudo labels of unlabelled data samples, especially for these samples that locate near the classification boundaries. This results in more unlabelled training samples being categorised as challenging samples, leading to a loss of information during the semi-supervised learning stage. This will also cause the challenging sample learner to identify more unlabelled soft prototypes, significantly increasing the manual labelling cost. On the other hand, if F is too small, the diversity between individual base models may decrease significantly, which can negatively influence the prediction performance of the ensemble model. The computational efficiency of the base models also decreases. Based on Fig. 5, the best value range of F is [8,13] to help S^{3}PAFES achieve good classification performance with relatively high computational efficiency.

Next, the influence of the level of granularity, G on the prediction performance of S^{3}PAFES is analysed. In this example, the value of G is changed from 3 to 12 with the interval of 1, and the values of the other three parameters are set as $F = 10$, $\delta_o = 0.9$ and $K_o = 10000$. The results obtained by S^{3}PAFES with different values of G are reported in Fig. 6 in terms of *acc*, *bacc*, *mcc*, S^E, S^C and t_{exe}.

It can be seen from Fig. 6 that the level of granularity, G determines the degree of fineness of soft prototype identification outcomes. A smaller G allows individual SPAFIS models to focus more on the global patterns of data and identify a smaller amount of more representative soft prototypes, whilst a greater G helps the individual SPAFIS models to focus on the local patterns of data and identify more soft prototypes that

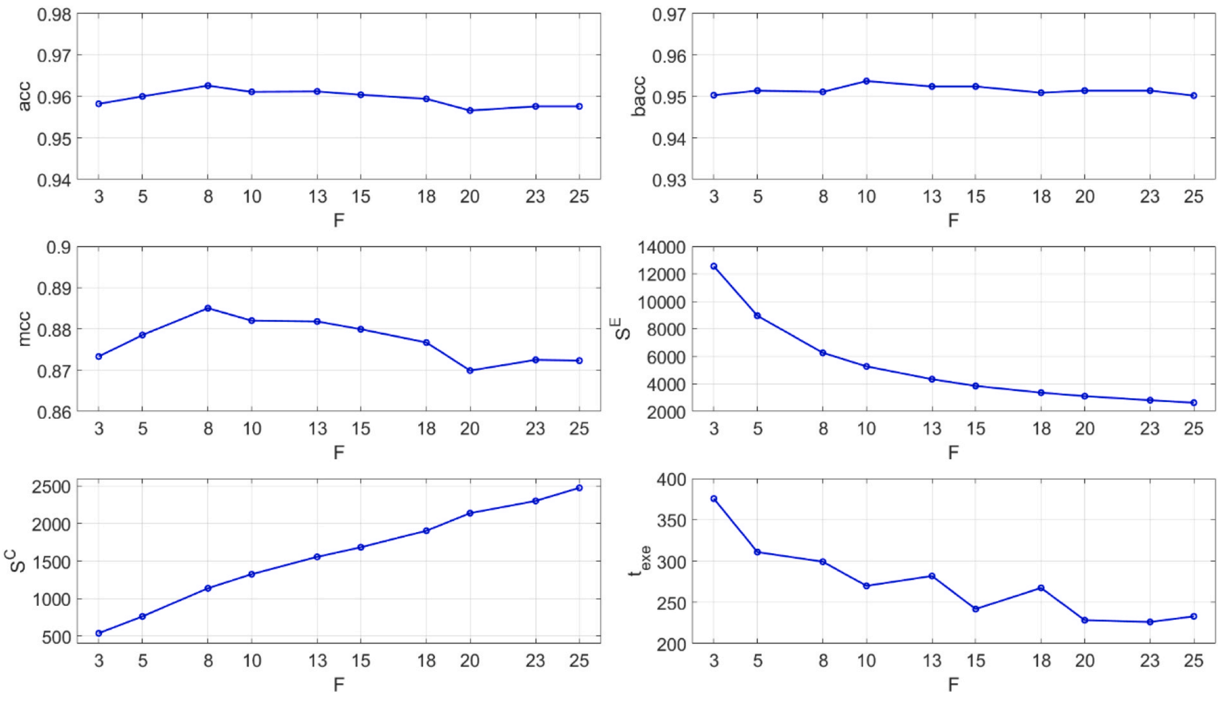


Fig. 5. Influence of ensemble scale F on the prediction performance of S^{3}PAFES.

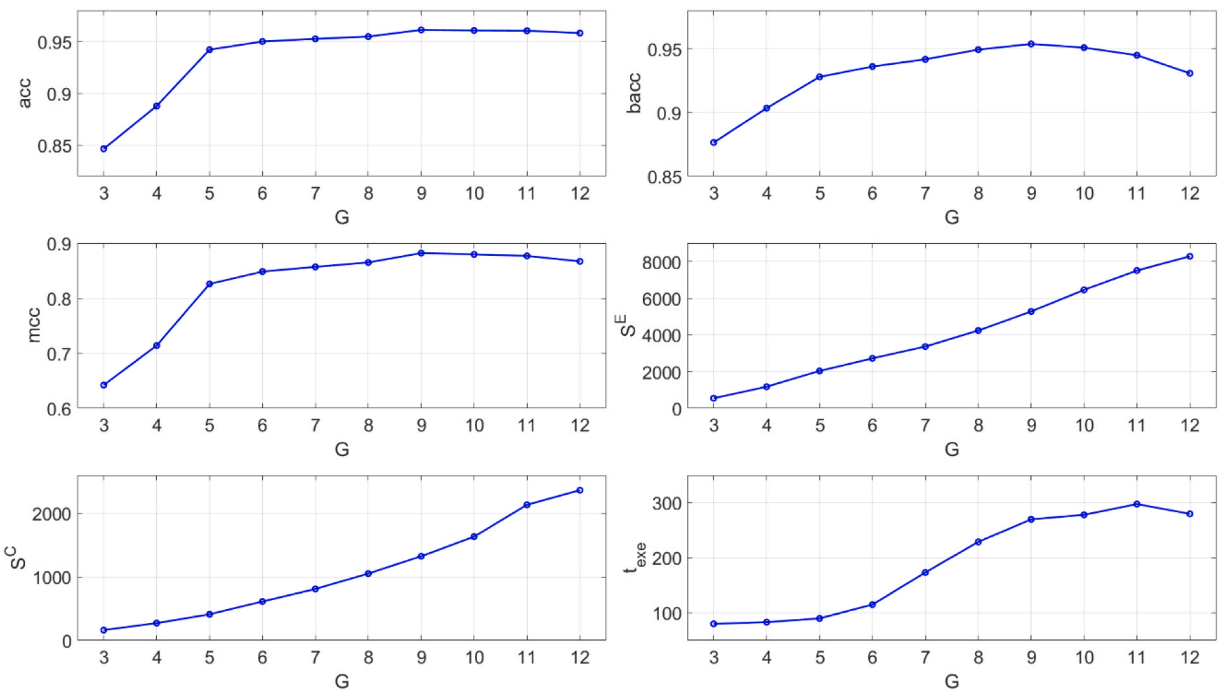


Fig. 6. Influence of level of granularity G on the prediction performance of S^{3}PAFES.

describe local patterns of data. If G is too low, the base classifiers may fail to learn sufficient details from data and cause the prediction performance of S^{3}PAFES to decrease. However, if G is too high, a SPAFIS model may recognise every data sample as a local peak of data density and identify a very large number of soft prototypes. In such cases, individual SPAFIS models may reduce to kNN classifiers with $k = 1$, and S^{3}PAFES is more likely to be overfitting. Hence, the optimal value range for G is [8,10], providing a good balance between model complexity and classification performance.

Then, the influence of the confidence threshold, δ_o on the performance of S^{3}PAFES is studied. The confidence threshold δ_o has a direct impact on the outcome of the two-step collaborative pseudo-labelling process. In this example, the value of δ_o is varied from 0.55 to 0.99 using irregular intervals, and the values of the other three parameters are set as $F = 10$, $G = 9$ and $K_o = 10000$. The classification performance of S^{3}PAFES with different confidence thresholds are given by Fig. 7 in terms of acc, $bacc$, mcc, S^E, S^C and t_{exe}. One can see from this table that if δ_o is set to be a smaller value, more pseudo-labelled data samples with low confidence will be selected by Condition 5 for updating the base models, which can inevitably deteriorate the prediction performance of

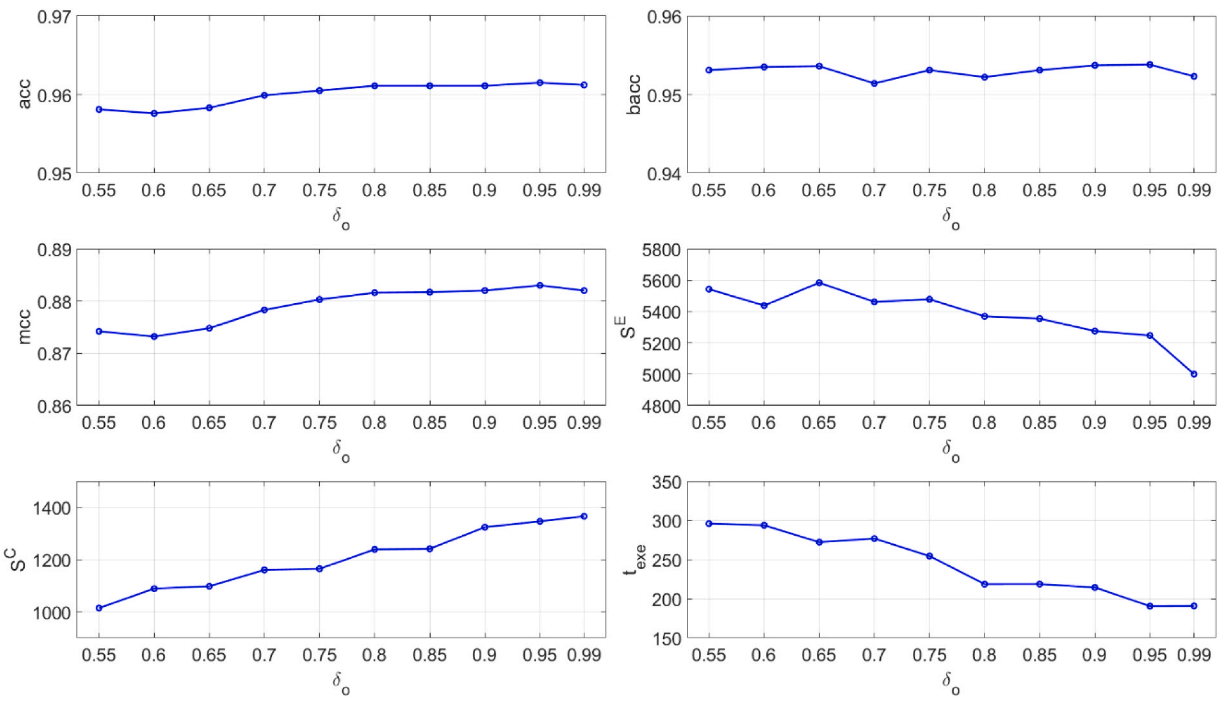


Fig. 7. Influence of confidence threshold δ_o on the prediction performance of S^{3}PAFES.

S^{3}PAFES due to the accumulation of pseudo-labelling errors. On the other hand, if δ_o is set to be very close to 1, i.e., $\delta_o = 0.99$, only very few pseudo-labelled data samples can satisfy Condition 5, which can cause a significant loss of information during the semi-supervised learning stage and increase the manual labelling costs potentially. Based on Fig. 7, the recommended value range of δ_o is $[0.85, 0.95]$.

Finally, the influence of the chunk size, K_o on the performance of S^{3}PAFES is investigated. In this example, the value of K_o is varied from 2500 to 25000 with an interval of 2500, and the values of the other three parameters are set as $F = 10$, $G = 9$ and $\delta_o = 0.9$. The classification performance of S^{3}PAFES with different chunk sizes are given by Fig. 8. It can be observed from Fig. 8 that the chunk size K_o can influence the classification performance of S^{3}PAFES to a certain degree due to its online chunk-wise learning mechanism. If K_o is set to be too small, i.e., $K_o = 2500$, the classification performance of S^{3}PAFES may decrease because individual base models may not be able to receive sufficient amounts of data in each learning cycle to precisely capture the underlying data patterns. The influence of K_o on the performance of S^{3}PAFES becomes less obvious if its value is sufficiently large, i.e., $K_o \geq 10000$.

In the numerical examples presented in the rest of this section, the recommended parameter setting for S^{3}PAFES is used by default, namely, $F = 10$, $G = 9$, $\delta_o = 0.9$ and $K_o = 10000$. However, it is worth noting that the recommended parameter setting only serves as a feasible option for users to consider. In practice, the best parameter setting is always different from problem to problem depending on the nature of data. Human expertise and prior knowledge of the given problem are typically needed when adjusting the externally controlled parameters to maximise the classification performance of S^{3}PAFES.

4.3. Performance demonstration

In this section, numerical experiments based on the aforementioned benchmark problems for network intrusion detection are carried out to illustrate the proposed concepts and general principles.

In this example, the classification performance of S^{3}PAFES in semi-supervised learning configuration is compared with its performance in supervised learning configuration to highlight the effectiveness of the proposed semi-supervised learning scheme. In running the experiments, for UNSWNB15 dataset, the original training set undergoes further subdivision by randomly selecting 1 %, 3 % and 5 % of data samples to build the labelled training set and using the remaining samples to build the unlabelled training set. In the case of CICIDS2017, CICIDS2018 and HIKARI2021 datasets, the labelled training set is constructed by randomly selecting 1 %, 3 % and 5 % of the data. 40 % of data is randomly chosen to form the testing set and the remaining data samples are then employed as unlabelled training set. The out-of-sample classification results obtained by S^{3}PAFES in both supervised learning and semi-supervised learning configurations on the testing sets of the four datasets are presented in Table 4 in terms of *acc*, *bacc* and *mcc*. The average number of soft prototypes identified by the F individual SPAFIS models of S^{3}PAFES, namely, S^E in both configurations are also reported in Table 4. Additionally, the number of soft prototypes identified from hard-to-classify unlabelled training samples by the challenging sample learner, S^C during the semi-supervised learning process is reported in the same table as well.

Moreover, to validate the effectiveness of the active learning scheme of S^{3}PAFES, a manual examination is conducted on the unlabelled soft prototypes learned by the challenging data learner following the semi-supervised learning process. During manual labelling, a naïve approach is used by identifying these soft prototypes that are formed by data samples primarily belonging to a single class (99.9 % in this study) and assigning class labels to them only. These manually labelled soft prototypes are then leveraged to augment S^{3}PAFES's knowledge base. The classification performance of S^{3}PAFES enhanced by manual labelling is also reported in Table 4 for clarity, and the updated values of S^E and S^C are given as well.

One can see from Table 4 that S^{3}PAFES can effectively utilise unlabelled training data to augment the knowledge base by exploiting the pseudo-labelling technique. Human experts can be involved to further enhance the knowledge base by providing additional supervision, helping S^{3}PAFES achieve better performance. In particularly, Table 4 shows that, for CICIDS2017 and CICIDS2018 datasets, the classification performance (in *acc* and *mcc*) of the proposed S^{3}PAFES is improved by continuously learning from unlabelled data chunks after being primed with labelled training data. The performance improvement is more significant when the amount of labelled training data is less, e.g., 1 %. In

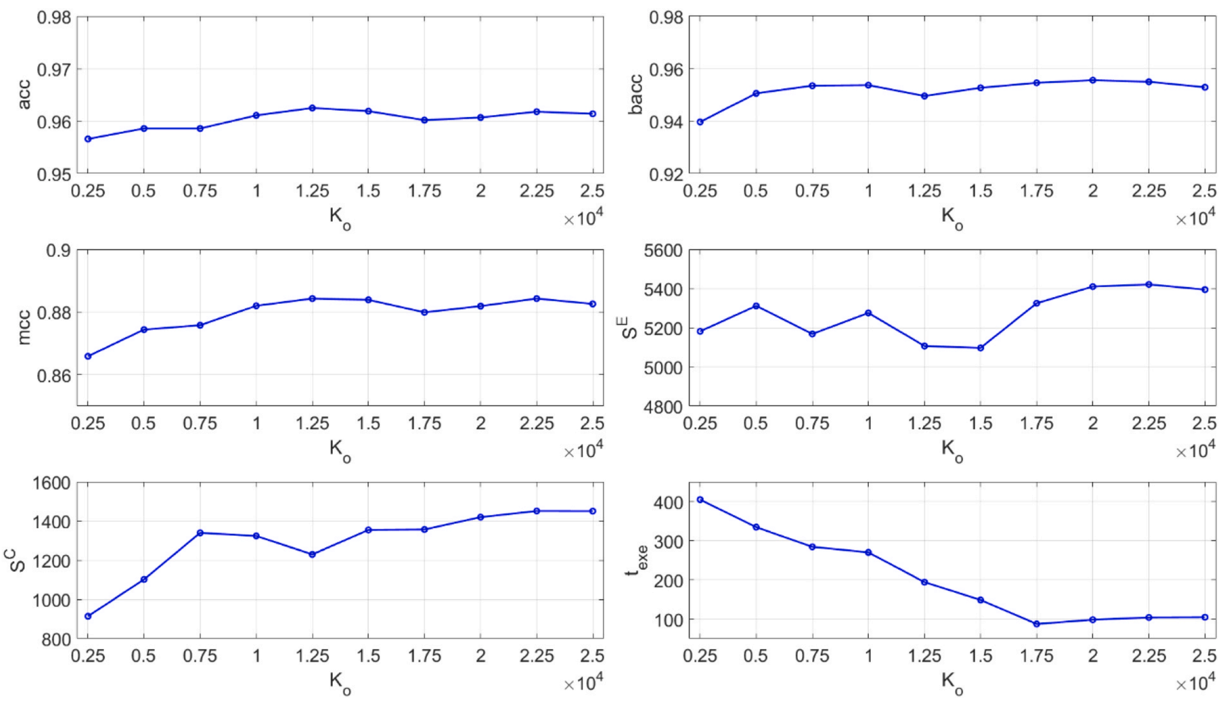


Fig. 8. Influence of chunk size K_o on the prediction performance of S^{3}PAFES.

Table 4
Performance demonstration of S^{3}PAFES on four benchmark problems.

Mode	Meas.	UNSWNB15			CICIDS2017		
		1 %	3 %	5 %	1 %	3 %	5 %
Supervised learning	*acc*	0.8131	0.8401	0.8453	0.9484	0.9571	0.9583
	bacc	0.8016	0.8281	0.8327	0.9229	0.9528	0.9574
	mcc	0.6274	0.6853	0.6977	0.8392	0.8718	0.8766
	S^E	361.21	917.88	1423.94	542.26	1167.41	1778.95
Semi-supervised learning	*acc*	0.8019	0.8239	0.8291	0.9536	0.9611	0.9611
	bacc	0.7832	0.8072	0.8128	0.9268	0.9492	0.9537
	mcc	0.6240	0.6644	0.6750	0.8537	0.8805	0.8820
	S^E	8089.20	8971.13	8914.68	4193.47	4508.66	5275.35
	S^C	12516.30	14223.60	15252.2	2445.40	1646.00	1323.90
Active learning	*acc*	0.8239	0.8355	0.8396	0.9588	0.9644	0.9633
	bacc	0.8065	0.8201	0.8247	0.9328	0.9536	0.9556
	mcc	0.6674	0.6855	0.6924	0.8697	0.8905	0.8881
	S^E	12981.39	13563.07	13264.28	5340.99	5263.05	5706.11
	S^C	2625.70	4902.30	6419.50	111.60	101.30	90.30
Mode	Meas.	CICIDS2018			HIKARI2021		
		1 %	3 %	5 %	1 %	3 %	5 %
Supervised learning	*acc*	0.9801	0.9824	0.9832	0.9121	0.8983	0.8899
	bacc	0.9565	0.9625	0.9649	0.6992	0.7506	0.7838
	mcc	0.9285	0.9371	0.9399	0.3672	0.3987	0.4217
	S^E	575.20	1078.51	1630.84	902.07	2312.22	3293.44
Semi-supervised learning	*acc*	0.9812	0.9833	0.9840	0.8921	0.8553	0.8889
	bacc	0.9541	0.9607	0.9627	0.7763	0.8230	0.7907
	mcc	0.9325	0.9399	0.9425	0.4203	0.4282	0.4279
	S^E	3370.25	3567.04	3942.60	1535.32	2602.16	3562.83
	S^C	2692.90	1247.20	840.90	9907.90	6633.80	7945.20
Active learning	*acc*	0.9847	0.9849	0.9847	0.8933	0.8549	0.8855
	bacc	0.9616	0.9630	0.9635	0.7647	0.8091	0.7663
	mcc	0.9451	0.9457	0.9452	0.4053	0.4109	0.3952
	S^E	4618.67	4144.11	4333.33	4817.90	4570.33	5944.50
	S^C	139.20	74.10	52.80	3328.50	2675.90	3166.10

the case of HIKARI2021 dataset, semi-supervised learning from unlabelled training data enables S^{3}PAFES to better handle the class imbalance, thereby achieving better *bacc* and *mcc*. On the other hand, for UNSWNB15 dataset, the classification performance of S^{3}PAFES decreases after semi-supervised learning from unlabelled training data. This is due to the significantly higher level of class overlaps exhibited by this dataset compared with the other three datasets, namely, data samples belonging to different classes share more common characteristics. The high-level class overlap poses a significant challenge for semi-supervised learning models that utilise pseudo labels. Furthermore, Table 4 also shows that by enhancing the ensemble model with manually labelled soft prototypes learned from hard-to-classify unlabelled data during the semi-supervised learning process, the performance of S^{3}PAFES is further improved in all three metrics on UNSWNB15, CICIDS2017 and CICIDS2018 datasets. Due to the class imbalance, the naïve approach to manual labelling fails to help S^{3}PAFES achieve better performance on HIKARI2021 data because most of the manually labelled soft prototypes belong to the majority class. However, this can be solved by using a more advanced manual labelling strategy.

From Table 4 it can be concluded that the proposed semi-supervised learning scheme can effectively help S^{3}PAFES to improve its classification performance by learning from unlabelled training data. The active learning scheme further enhance performance of S^{3}PAFES by incorporating the manually labelled soft prototypes to augment the knowledge base of the ensemble model.

Next, the classification performance of S^{3}PAFES in semi-supervised learning configuration with optimised externally controlled parameters is evaluated on the four benchmark problems. In this example, the values of the four parameters, namely, F, G, δ_o and K_o are selected from the respective candidate sets [8,10,13], [8–10], $[0.85, 0.9, 0.95]$ and $[10000, 12500, 15000]$, resulting a total of 81 different combinations. 20 % of the labelled training data are randomly selected out to build the

Table 5
Performance demonstration of S^{3}PAFES on four benchmark problems with optimised parameters.

Algorithm	Meas.	UNSWNB15			CICIDS2017		
		1 %	3 %	5 %	1 %	3 %	5 %
Optimised parameter setting	*acc*	**0.8072**	0.8201	0.8235	0.9522	**0.9631**	**0.9657**
	bacc	**0.7876**	0.8025	0.8060	**0.9300**	0.9470	0.9501
	mcc	**0.6399**	0.6595	0.6670	0.8513	**0.8850**	**0.8928**
Recommended parameter setting	*acc*	0.8019	**0.8239**	**0.8291**	**0.9536**	0.9611	0.9611
	bacc	0.7832	**0.8072**	**0.8128**	0.9268	**0.9492**	**0.9537**
	mcc	0.6240	**0.6644**	**0.6750**	**0.8537**	0.8805	0.8820
Algorithm	Meas.	CICIDS2018			HIKARI2021		
		1 %	3 %	5 %	1 %	3 %	5 %
Optimised parameter setting	*acc*	0.9805	**0.9836**	**0.9846**	0.8832	**0.8629**	0.8640
	bacc	0.9540	**0.9613**	0.9622	**0.8043**	**0.8982**	**0.8980**
	mcc	0.9297	**0.9410**	**0.9446**	**0.4380**	**0.5034**	**0.5051**
Recommended parameter setting	*acc*	**0.9812**	0.9833	0.9840	**0.8921**	0.8553	**0.8889**
	bacc	**0.9541**	0.9607	**0.9627**	0.7763	0.8230	0.7907
	mcc	**0.9325**	0.9399	0.9425	0.4203	0.4282	0.4279

validation set to help S^{3}PAFES identify the best parameter setting in an exhaustive manner. The out-of-sample classification results (in *acc*, *bacc* and *mcc*) obtained by S^{3}PAFES on the testing sets of the four datasets with the optimal parameter setting that yields the highest *bacc* and *mcc* on the validation data after semi-supervised learning from both labelled and unlabelled training data are reported in Table 5. For comparison, results obtained using the recommended parameter settings are also included. The best result per measure per dataset is in bold for visual clarity.

Table 5 shows that optimising externally controlled parameters significantly improves performance on the HIKARI2021 dataset. This improvement is largely due to the high class imbalance of HIKARI2021, making S^{3}PAFES more sensitive to parameters like the level of granularity (*G*). However, for UNSWNB15, CICIDS2017, and CICIDS2018 datasets, performance gains from parameter optimisation are minimal, despite the significantly increased computational complexity due to the exhaustive search. The performance comparison given by Table 5 suggests that the recommended parameter setting serves as a strong starting point, enabling S^{3}PAFES to achieve good performance, especially in the absence of prior knowledge.

4.4. Performance comparison

Next, the semi-supervised learning performance of the proposed S^{3}PAFES is compared with the aforementioned nine SOTA semi-supervised learning algorithms on the four benchmark problems under the same experimental protocol used in the previous example. The settings of externally controlled parameters of the comparative algorithms for numerical experiments have been given in Section 4.1. The out-of-sample classification performances of S^{3}PAFES and the nine comparative algorithms on the testing sets of UNSWNB15, CICIDS2017, CICIDS2018 and HIKARI2021 datasets are evaluated based on the three aforementioned metrics and reported in Tables 6, 7, 8, 9, respectively. The best results are in bold for visual clarity.

For better comparison, the classification performances of the 10 semi-supervised learning algorithms on the four benchmark problems with different ratios of labelled training data are ranked from the best (1st) to the lowest (10th), and the average ranks per performance metric of the 10 algorithms across the experiments are reported in Table 10, along with the respective average performances.

It can be clearly seen from Table 10 that S^{3}PAFES outperforms the nine comparative semi-supervised learning algorithms on the four imbalanced benchmark datasets for network intrusion detection with the greatest overall *bacc* and *mcc* values, and it also achieves the second highest *acc* value (DMS^{3}OF achieves the best classification results in *acc*). This comparison shows the superiority of S^{3}PAFES in semi-supervised learning from network data streams for intrusion detection in infinite delay environments. It is also crucial to emphasize that the performance of S^{3}PAFES can be further improved through collaboration with human experts.

In addition, the average time consumption (in seconds) of the 10 semi-supervised learning algorithms during the experiments on the four benchmark datasets are presented in Table 11, offering a comparison of their computational efficiency. The comparison presented in Table 11 highlights the high-level computational efficiency of S^{3}PAFES, with the average runtime of 258 seconds across the four datasets, ranked the fourth place among the 10 algorithms. It is also worth noting that S^{3}PAFES is an ensemble framework comprising 10 base learners and one challenging sample learner. The numerical experiments were carried out on a single machine without parallelisation, suggesting that the reported runtime does not reflect its true computational efficiency. In practice, the computational efficiency of S^{3}PAFES can be further improved through the use of distributed computation.

4.5. Ablation analysis

Finally, ablation analysis is conducted to justify the effectiveness of the proposed base model pruning scheme, two-step collaborative pseudo-labelling scheme and the utilisation of sampling techniques in the proposed S^{3}PAFES. In this example, six alternative versions of S^{3}PAFES, denoted as S^{3}PAFES$_1$, S^{3}PAFES$_2$, S^{3}PAFES$_3$, S^{3}PAFES$_4$, S^{3}PAFES$_5$ and S^{3}PAFES$_6$ are implemented:

1) S^{3}PAFES$_1$: S^{3}PAFES without the base model pruning scheme;
2) S^{3}PAFES$_2$: S^{3}PAFES but only uses Condition 4 for collaborative pseudo labelling;
3) S^{3}PAFES$_3$: S^{3}PAFES but only uses Condition 5 for collaborative pseudo labelling;
4) S^{3}PAFES$_4$: S^{3}PAFES with relaxed Condition 4 based on majority voting instead of requiring full consensus;
5) S^{3}PAFES$_5$: S^{3}PAFES with relaxed Condition 5 by removing the second criterion;
6) S^{3}PAFES$_6$: S^{3}PAFES without using oversampling and down sampling.

The same four datasets are used for comparing the classification performance between S^{3}PAFES and the four alternatives. The same experimental protocol used in Section 4.3 is employed, namely, 1 %, 3 %, 5 % of the data samples are randomly selected as the labelled training data to prime the models, 40 % of the data is used as the testing set to evaluate the out-of-sample classification performances of the learned models, and the remaining data samples are used as the unlabelled training set for the models to continue self-updating in a semi-supervised manner. The classification performances of the seven S^{3}PAFESs on the testing data of the four datasets are reported in Table 12 in terms of *acc*, *bacc* and *mcc*. The best results are in bold. For visual clarity, the average performances of S^{3}PAFES and its variants per criterion are depicted in Fig. 9.

One can see from Fig. 9 that S^{3}PAFES outperforms its variants across the four imbalanced intrusion detection benchmark datasets under different experimental settings by offering the greatest classification results based on *bacc* and *mcc*. In particular, it can be observed from Table 12 that S^{3}PAFES achieves the best performance on CICIDS2017

Table 6
Classification performance comparison between S^{3}PAFES and nine SOTA semi-supervised learning algorithms on UNSWNB15 dataset.

Algorithm	1 %			3 %			5 %		
	acc	*bacc*	*mcc*	*acc*	*bacc*	*mcc*	*acc*	*bacc*	*mcc*
S^{3}PAFES	0.8019	0.7832	0.6240	0.8239	0.8072	0.6644	0.8291	0.8128	0.6750
SSEPBS	0.8008	0.7790	0.6356	0.7859	0.7624	0.6103	0.8025	0.7810	0.6381
SSECBS	0.8081	0.7870	0.6491	0.8080	0.7869	0.6484	0.8076	0.7864	0.6484
OSSLVQ	0.5636	0.5752	0.1967	0.5626	0.5697	0.1737	0.5778	0.5870	0.2135
ORSSL	0.6402	0.6690	0.4019	0.6429	0.6707	0.4060	0.6655	0.6925	0.4460
TMPM	0.4494	0.5000	0.0000	0.4494	0.5000	0.0000	0.4899	0.5000	0.0000
TriT	0.8086	0.7875	0.6500	0.8092	0.7884	0.6502	0.8102	0.7894	0.6524
STHP	**0.8249**	**0.8119**	**0.6553**	0.8351	0.8219	0.6775	0.8392	0.8267	0.6845
S^{3}OFIS	0.8235	0.8095	0.6550	0.8314	0.8182	0.6695	0.8342	0.8217	0.6741
DMS^{3}OF	0.8211	0.8100	0.6431	**0.8403**	**0.8276**	**0.6873**	**0.8425**	**0.8292**	**0.6936**

Table 7
Classification performance comparison between S^3PAFES and nine SOTA semi-supervised learning algorithms on CICIDS2017 dataset.

Algorithm	1 %			3 %			5 %		
	acc	*bacc*	*mcc*	*acc*	*bacc*	*mcc*	*acc*	*bacc*	*mcc*
S^3PAFES	0.9536	0.9268	0.8537	0.9611	0.9492	**0.8805**	0.9611	0.9537	0.8820
SSEPBS	0.8863	0.7154	0.6071	0.8926	0.7410	0.6327	0.8898	0.7284	0.6212
SSECBS	0.9163	0.8469	0.7261	0.9162	0.8475	0.7258	0.9168	0.8478	0.7277
OSSLVQ	0.7619	0.6629	0.3558	0.8092	0.6809	0.3898	0.8083	0.6697	0.3732
ORSSL	0.6258	0.6985	0.3243	0.7033	0.7036	0.3545	0.6350	0.7011	0.3314
TMPM	0.7223	0.8205	0.5111	0.7879	0.8568	0.5804	0.8249	0.8760	0.6247
TriT	0.9165	0.8470	0.7266	0.9205	0.8598	0.7423	0.9233	0.8655	0.7518
STHP	0.9405	0.9289	0.8225	0.9358	0.9306	0.8138	0.9263	0.9254	0.7918
S^3OFIS	0.9315	**0.9389**	0.8112	0.9360	**0.9494**	0.8260	0.9437	**0.9556**	0.8445
DMS^3OF	**0.9563**	0.9190	**0.8593**	**0.9624**	0.9282	0.8792	**0.9634**	0.9281	**0.8822**

Table 8
Classification performance comparison between S^3PAFES and nine SOTA semi-supervised learning algorithms on CICIDS2018 dataset.

Algorithm	1 %			3 %			5 %		
	acc	*bacc*	*mcc*	*acc*	*bacc*	*mcc*	*acc*	*bacc*	*mcc*
S^3PAFES	**0.9812**	**0.9541**	**0.9325**	**0.9833**	**0.9607**	**0.9399**	**0.9840**	**0.9627**	**0.9425**
SSEPBS	0.9182	0.8106	0.6900	0.9195	0.8112	0.6944	0.9194	0.8073	0.6921
SSECBS	0.9307	0.8479	0.7427	0.9162	0.8475	0.7258	0.9325	0.8503	0.7491
OSSLVQ	0.5327	0.6528	0.2450	0.6268	0.6875	0.2907	0.6848	0.7071	0.3212
ORSSL	0.4788	0.6250	0.2002	0.5462	0.6169	0.1812	0.4707	0.5866	0.1342
TMPM	0.6921	0.8002	0.4519	0.7751	0.8489	0.5400	0.8342	0.8816	0.6151
TriT	0.9353	0.8472	0.7576	0.9388	0.8555	0.7715	0.9395	0.8560	0.7740
STHP	0.9306	0.9178	0.7782	0.8869	0.8844	0.6745	0.8692	0.8699	0.6361
S^3OFIS	0.9369	0.9386	0.8063	0.9055	0.9226	0.7364	0.8836	0.9102	0.6946
DMS^3OF	0.9692	0.9341	0.8894	0.9691	0.9273	0.8881	0.9700	0.9271	0.8911

Table 9
Classification performance comparison between S^3PAFES and nine SOTA semi-supervised learning algorithms on HIKARI2021 dataset.

Algorithm	1 %			3 %			5 %		
	acc	*bacc*	*mcc*	*acc*	*bacc*	*mcc*	*acc*	*bacc*	*mcc*
S^3PAFES	0.8921	0.7763	**0.4203**	0.8553	**0.8230**	**0.4282**	0.8889	0.7907	**0.4279**
SSEPBS	0.9318	0.5000	0.0000	0.9323	0.5000	0.0000	0.9322	0.5000	0.0000
SSECBS	0.9318	0.5000	−0.0012	0.9322	0.5000	−0.0013	0.9320	0.5046	0.0353
OSSLVQ	0.6983	**0.7975**	0.3137	0.7017	0.7995	0.3148	0.7010	0.7992	0.3145
ORSSL	0.6980	0.7861	0.3031	0.6873	0.7831	0.2967	0.7001	0.7548	0.2709
TMPM	0.6590	0.7812	0.2901	0.6935	0.8064	0.3214	0.6753	**0.8099**	0.3208
TriT	**0.9315**	0.5084	0.0655	**0.9322**	0.5047	0.0349	**0.9321**	0.5031	0.0263
STHP	0.8964	0.7097	0.3480	0.8770	0.6942	0.3004	0.8667	0.6855	0.2782
S^3OFIS	0.9251	0.6078	0.2788	0.9261	0.5891	0.2493	0.9245	0.5819	0.2306
DMS^3OF	0.9265	0.6071	0.2816	0.9295	0.5990	0.2801	0.9306	0.5990	0.2869

Table 10
Average classification performance comparison between S^3PAFES and nine SOTA semi-supervised learning algorithms across the experiments.

Algorithm	*acc*		*bacc*		*mcc*	
	Rank	Value	Rank	Value	Rank	Value
S^3PAFES	3.5833	0.9096	**2.6667**	**0.8750**	**2.0833**	**0.7226**
SSEPBS	5.1250	0.8843	8.0833	0.7030	7.4167	0.4851
SSECBS	4.6667	0.8957	6.9167	0.7461	6.4167	0.5313
OSSLVQ	8.5833	0.6691	7.5000	0.6824	7.5000	0.2919
ORSSL	9.3333	0.6245	7.5833	0.6907	8.2500	0.3042
TMPM	9.0833	0.6711	6.0000	0.7485	7.1667	0.3546
TriT	3.8750	0.8998	6.0000	0.7510	5.3333	0.5503
STHP	4.5833	0.8857	3.5000	0.8339	3.7500	0.6217
S^3OFIS	4.0000	0.9002	3.3333	0.8203	4.3333	0.6230
DMS^3OF	**2.1667**	**0.9234**	3.4167	0.8196	2.7500	0.6802

and CICDS2018 datasets in the majority of cases, and it offers better performance on HIKARI2021 dataset based on *bacc* and *macc* when the ratio of labelling training data is no more than 3 %. However, S^3PAFES_2 outperforms the rest on UNSWNB15 dataset when the ratio of labelled training data is 1 % and S^3PAFES_3 offers the best performance on this dataset under the other two settings. This is because these unlabelled training samples satisfying Condition 5 but not Condition 4 are more likely assigned with wrong pseudo labels by NNE classifiers due to the higher level of class overlaps exhibited by this dataset. One may also notice that S^3PAFES_4 attains the best *bacc* on four of 12 cases, and both S^3PAFES_4 and S^3PAFES_5 outperform S^3PAFES occasionally in terms of *bacc*. This suggests that, depending on the data characteristics, Conditions 4 and 5 may be too conservative in some cases. As a result, only these unlabelled samples highly similar to known patterns are assigned pseudo labels. The more informative data samples that either represent data patterns unseen before or are located near the classification boundaries are more likely to be classified as challenging samples, excluded from base learner updating. This numerical example serves as the proof of efficacy of the proposed base model pruning scheme, two-step collaborative pseudo-labelling scheme and the utilisation of sampling techniques in the proposed S^3PAFES.

Table 11
Average time consumption of the 10 semi-supervised learning algorithms on the four benchmark datasets.

Algorithm	UNSWNB15	CICIDS2017	CICIDS2018	HIKARI2021	Overall
S^3PAFES	497.4411	121.1062	112.2181	301.2787	258.0110
SSEPBS	209.2709	193.1676	167.0254	256.0396	206.3759
SSECBS	16.1762	16.1917	7.5044	143.9301	45.9506
OSSLVQ	72.7772	48.6389	19.5280	87.3233	57.0669
ORSSL	2010.7122	566.8227	758.6677	1397.5495	1183.4380
TMPM	8723.4738	7593.8200	13605.0866	40216.0251	17534.6014
TriT	29.0566	522.0490	135.8928	2885.1229	893.0303
STHP	33150.7556	4294.1123	730.3300	8168.2564	11585.8636
S^3OFIS	569.3767	256.8998	248.7905	1123.6853	549.6881
DMS^{3}OF	455.8415	156.1632	153.9138	773.8513	384.9425

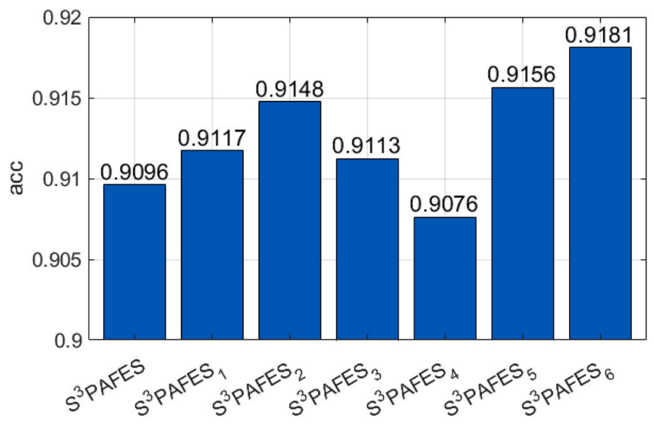
Table 12
Numerical results of ablation analysis.

Dataset	Algorithm	1 %			3 %			5 %		
		acc	*bacc*	*mcc*	*acc*	*bacc*	*mcc*	*acc*	*bacc*	*mcc*
UNSWNB15	S^3PAFES	0.8019	0.7832	0.6240	0.8239	0.8072	0.6644	0.8291	0.8128	0.6750
	S^3PAFES$_1$	0.8069	0.7882	0.6353	0.8240	0.8073	0.6651	0.8277	0.8112	0.6726
	S^3PAFES$_2$	**0.8090**	**0.7910**	**0.6367**	0.8270	0.8109	0.6693	0.8333	0.8177	0.6812
	S^3PAFES$_3$	0.7980	0.7833	0.6030	**0.8313**	**0.8176**	**0.6710**	**0.8371**	**0.8230**	**0.6852**
	S^3PAFES$_4$	0.7874	0.7702	0.5855	0.8234	0.8078	0.6597	0.8332	0.8178	0.6807
	S^3PAFES$_5$	0.8011	0.7821	0.6231	0.8241	0.8082	0.6628	0.8254	0.8088	0.6680
	S^3PAFES$_6$	0.7941	0.7725	0.6195	0.8124	0.7929	0.6513	0.8168	0.7978	0.6586
CICIDS2017	S^3PAFES	**0.9536**	0.9268	0.8537	**0.9611**	**0.9492**	**0.8805**	**0.9611**	0.9537	**0.8820**
	S^3PAFES$_1$	0.9512	0.9318	0.8495	0.9586	0.9517	0.8752	0.9609	0.9533	0.8814
	S^3PAFES$_2$	0.9532	**0.9322**	**0.8551**	0.9599	0.9474	0.8769	0.9609	0.9527	0.8813
	S^3PAFES$_3$	0.9401	0.9061	0.8114	0.9538	0.9441	0.8603	0.9577	0.9539	0.8735
	S^3PAFES$_4$	0.9407	0.9094	0.8142	0.9556	0.9474	0.8659	0.9583	**0.9544**	0.8751
	S^3PAFES$_5$	0.9506	0.9202	0.8438	0.9567	0.9491	0.8692	0.9576	0.9507	0.8723
	S^3PAFES$_6$	0.9522	0.9159	0.8470	0.9597	0.9289	0.8712	0.9606	0.9310	0.8741
CICIDS2018	S^3PAFES	**0.9812**	**0.9541**	**0.9325**	**0.9833**	0.9607	**0.9399**	**0.9840**	0.9627	**0.9425**
	S^3PAFES$_1$	0.9796	0.9494	0.9266	0.9831	0.9605	0.9393	0.9835	0.9611	0.9408
	S^3PAFES$_2$	0.9793	0.9505	0.9258	**0.9833**	0.9597	**0.9399**	0.9837	0.9610	0.9414
	S^3PAFES$_3$	0.9795	0.9502	0.9262	0.9805	0.9599	0.9304	0.9833	0.9627	0.9401
	S^3PAFES$_4$	0.9804	0.9526	0.9294	0.9828	**0.9617**	0.9372	0.9835	**0.9636**	0.9408
	S^3PAFES$_5$	0.9805	0.9540	0.9299	0.9830	0.9611	0.9393	0.9839	0.9619	0.9421
	S^3PAFES$_6$	0.9717	0.9282	0.8971	0.9771	0.9401	0.9171	0.9771	0.9392	0.9171
HIKARI2021	S^3PAFES	0.8921	**0.7763**	**0.4203**	0.8553	0.8230	0.4282	0.8889	0.7907	0.4279
	S^3PAFES$_1$	0.8835	0.7608	0.3996	0.8925	0.7726	0.4134	0.8893	0.7898	0.4273
	S^3PAFES$_2$	0.9160	0.6857	0.3629	0.8906	0.7660	0.4080	0.8808	**0.8268**	**0.4538**
	S^3PAFES$_3$	0.8971	0.7514	0.3989	0.8896	0.7919	0.4298	0.8870	0.7972	0.4317
	S^3PAFES$_4$	0.8885	0.7554	0.3944	0.8745	**0.8445**	**0.4639**	0.8830	0.8113	0.4410
	S^3PAFES$_5$	0.9181	0.6725	0.3531	0.9074	0.7227	0.3839	0.8993	0.7536	0.4044
	S^3PAFES$_6$	**0.9317**	0.5728	0.2498	**0.9317**	0.6001	0.2949	**0.9321**	0.5976	0.2931

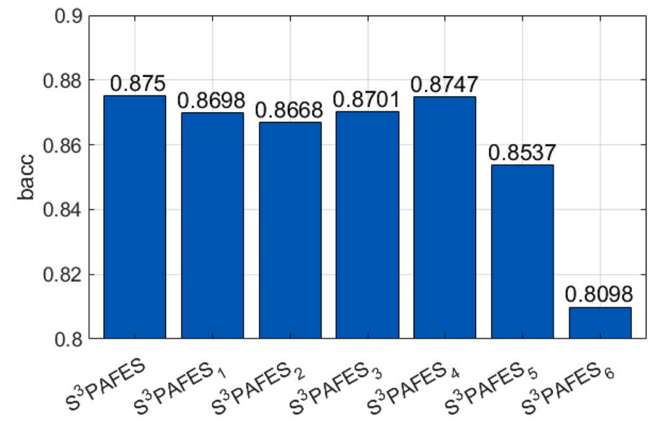
5. Conclusion

In this paper, a novel soft prototype-based autonomous fuzzy ensemble approach, named S^3PAFES is proposed for online learning from network data streams. It is designed for intrusion detection, offering high transparency and minimal reliance on human expert involvement. The proposed S^3PAFES exploits two-step collaborative pseudo-labelling scheme to perform semi-supervised learning from unlabelled data chunks for self-improving its knowledge base. It is capable of effectively handling class imbalance in the data streams and reducing model bias through the combined use of different sampling techniques. To enhance its prediction performance, S^3PAFES constantly monitors the performances of its base models and substitutes the weaker ones promptly. While S^3PAFES is designed for fully autonomous learning from data streams, human experts can still play a valuable role through its unique online active learning scheme. Human experts can contribute to the improvement of the knowledge base by manually labelling a small set of unlabelled soft prototypes extracted from hard-to-classify data samples identified during semi-supervised learning. Numerical examples on four popular network intrusion detection datasets demonstrated the superior performance of the proposed S^3PAFES in infinite delay environments, surpassing a number of SOTA online and offline inductive semi-supervised learning algorithms.

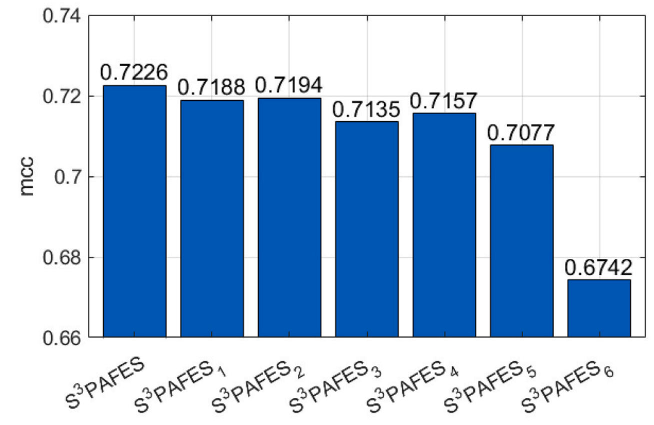
There are several considerations for future work. Firstly, the utilisation of soft prototypes enables S^3PAFES to better summarise the underlying structure and local patterns of data. However, soft prototypes are more vulnerable to pseudo-labelling errors compared to conventional crisp prototypes. This is because every soft prototype is a weighted combination of all the labelled and pseudo-labelled data of the same class, causing pseudo-labelling errors to propagate within the knowledge base more easily. When the data exhibits high-level class overlap (e.g., UNSWNB15), the performance of S^3PAFES may decrease significantly due to the more frequently happening pseudo-labelled errors. Hence, it would be very helpful to design a mechanism to confine error propagation in S^3PAFES. Secondly, the two-step collaborative pseudo-labelling scheme used by S^3PAFES only selects these pseudo-labelled samples with the highest consensus for self-updating its knowledge base. This scheme can significantly reduce the pseudo-labelling errors, facilitating the semi-supervised learning of S^3PAFES from data streams. However, as shown in the ablation analysis, there is a potential drawback that this scheme may be overly conservative as these selected pseudo-labelled samples are more likely to align closely with known patterns, whilst the novel patterns represented by unselected pseudo-labelled samples are likely to be missed out, causing a loss of



(a) Average performance comparison in *acc*



(b) Average performance comparison in *bacc*



(c) Average performance comparison in *mcc*

Fig. 9. Average performances of S^{3}PAFES and its variants per criterion.

valuable information. Therefore, it would be worth further exploring more advanced pseudo-labelling strategies that can achieve a balance between the accuracy of pseudo labels and the diversity of selected pseudo-labelled data. Last, but not the least, the proposed S^{3}PAFES currently has a fixed number of base models in the ensemble framework, and all the levels of granularity of the base models are set to be the same. It would be very useful to design a new scheme to let each individual base model to self-determine the level of granularity based on the statistical characteristics of the input data. In addition, it would also be useful to let S^{3}PAFES self-adjust the number of base models dynamically according to the nature of data to achieve greater intrusion performance with less consumption of computational resources.

CRediT authorship contribution statement

Haiyue Yuan: Writing – review & editing, Writing – original draft, Software, Methodology. **Gareth Howells:** Writing – review & editing, Writing – original draft, Methodology, Funding acquisition, Conceptualization. **Xiaowei Gu:** Writing – review & editing, Writing – original draft, Software, Methodology, Funding acquisition, Conceptualization.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xiaowei Gu, Gareth Howells, Haiyue Yuan report financial support was provided by Defence Science and Technology Laboratory. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Research funded by Frazer-Nash Consultancy Ltd. on behalf of the Defence Science and Technology Laboratory (Dstl) which is an executive agency of the UK Ministry of Defence providing world class expertise and delivering cutting-edge science and technology for the benefit of the nation and allies. The research supports the Autonomous Resilient Cyber Defence (ARCD) project within the Dstl Cyber Defence Enhancement programme.

Data availability

Data will be made available on request.

References

[1] Cyber security breaches survey, *UK Government*, 2024. ⟨https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2024⟩.
[2] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, F. De Turck, Towards model generalization for intrusion detection: unsupervised machine learning techniques, J. Netw. Syst. Manag. 30 (1) (2022) 1–25.
[3] D. Gümüşbaş, T. Yıldırım, A. Genovese, F. Scotti, A comprehensive survey of databases and deep learning methods for cybersecurity and intrusion detection systems, IEEE Syst. J. 15 (2) (2021) 1717–1731.
[4] F. Noorbehbahani, A. Fanian, R. Mousavi, H. Hasannejad, An incremental intrusion detection system using a new semi-supervised stream classification method, Int. J. Commun. Syst. 30 (4) (2017) 1–26.
[5] M.H.L. Louk, B.A. Tama, Dual-IDS: a bagging-based gradient boosting decision tree model for network anomaly intrusion detection system, Expert Syst. Appl. 213 (2023) 119030.
[6] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, Surv. Intrusion Detect. Syst.: Tech., datasets Chall.," *Cybersecur.* 2 (1) (2019) 1–22.
[7] W. Wang, et al., HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection, IEEE Access 6 (2017) 1792–1806.
[8] K. Shaukat, S. Luo, V. Varadharajan, I.A. Hameed, M. Xu, A survey on machine learning techniques for cyber security in the last decade, IEEE Access 8 (2020) 222310–222354.
[9] A.I. Saleh, F.M. Talaat, L.M. Labib, A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers, Artif. Intell. Rev. 51 (3) (2019) 403–443.
[10] N. Naik, R. Diao, Q. Shen, Dynamic fuzzy rule interpolation and its application to intrusion detection, IEEE Trans. Fuzzy Syst. 26 (4) (2018) 1878–1892.
[11] P.A.A. Resende, A.C. Drummond, A survey of random forest based methods for intrusion detection systems, ACM Comput. Surv. 51 (3) (2018) 1–36.
[12] A. Shahraki, M. Abbasi, Ø. Haugen, Boosting algorithms for network intrusion detection: a comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost, Eng. Appl. Artif. Intell. 94 (2020) 103770.
[13] M. Douiba, S. Benkirane, A. Guezzaz, M. Azrour, An improved anomaly detection model for IoT security using decision tree and gradient boosting, J. Supercomput. 79 (3) (2023) 3392–3411.
[14] T. Su, H. Sun, J. Zhu, S. Wang, Y. Li, BAT: deep learning methods on network intrusion detection using NSL-KDD dataset, in: IEEE Access, 8, 2020, pp. 29575–29585.

[15] M. Data, M. Aritsugi, T-DFNN: an incremental learning algorithm for intrusion detection systems, IEEE Access 9 (2021) 154156–154171.

[16] Y. Jia, M. Wang, Y. Wang, Network intrusion detection algorithm based on deep neural network, IET Inf. Secur. 13 (1) (2019) 48–53.

[17] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural betworks, IEEE Access 5 (2017) 21954–21961.

[18] S. Dong, P. Wang, K. Abbas, A survey on deep learning and its applications, Comput. Sci. Rev. 40 (2021) 100379.

[19] K. He, D.D. Kim, M.R. Asghar, Adversarial machine learning for network intrusion detection systems: a comprehensive survey, IEEE Commun. Surv. Tutor. 25 (1) (2023) 538–566.

[20] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nat. Mach. Intell. 1 (5) (2019) 206–215.

[21] P. Casas, J. Mazel, P. Owezarski, Unsupervised network intrusion detection systems: detecting theunknown without knowledge, Comput. Commun. 35 (7) (2012) 772–783.

[22] H. Choi, M. Kim, G. Lee, W. Kim, Unsupervised learning approach for network intrusion detection system using autoencoders, J. Supercomput. 75 (9) (2019) 5597–5621.

[23] J.E. van Engelen, H.H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2) (2020) 373–440.

[24] R.A.R. Ashfaq, X.Z. Wang, J.Z. Huang, H. Abbas, Y.L. He, Fuzziness based semi-supervised learning approach for intrusion detection system, Inf. Sci. (Ny.). 378 (2017) 484–497.

[25] W. Li, W. Meng, M.H. Au, Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in IoT environments, J. Netw. Comput. Appl. 161 (2020) 102631.

[26] S.R. Khonde, V. Ulagamuthalvi, Ensemble-based semi-supervised learning approach for a distributed intrusion detection system, J. Cyber Secur. Technol. 3 (3) (2019) 163–188.

[27] H. Yao, D. Fu, P. Zhang, M. Li, Y. Liu, MSML: a novel multilevel semi-supervised machine learning framework for intrusion detection system, IEEE Internet Things J. 6 (2) (2019) 1949–1959.

[28] S. Cai, D. Han, D. Li, A feedback semi-supervised learning with meta-gradient for intrusion detection, IEEE Syst. J. 17 (1) (2023) 1158–1169.

[29] V.M.A. Souza and 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.

[30] S. Ud Din, J. Shao, J. Kumar, W. Ali, J. Liu, Y. Ye, Online reliable semi-supervised learning on evolving data streams, Inf. Sci. (Ny.). 525 (2020) 153–171.

[31] E.K. Viegas, A.O. Santin, V.V. Cogo, and V. Abreu, “A reliable semi-supervised intrusion detection model: one year of network traffic anomalies,” in *IEEE International Conference on Communications*, 2020, pp. 1–6.

[32] Y. Guo, J. Pu, B. Jiao, Y. Peng, D. Wang, S. Yang, Online semi-supervised active learning ensemble classification for evolving imbalanced data streams, Appl. Soft Comput. 155 (2024) 111452.

[33] M. Abdel-Basset, H. Hawash, R.K. Chakrabortty, M.J. Ryan, Semi-Supervised spatiotemporal deep learning for intrusions detection in IoT networks, IEEE Internet Things J. 8 (15) (2021) 12251–12265.

[34] J. Jiang, F. Liu, W.W.Y. Ng, Q. Tang, W. Wang, Q.V. Pham, Dynamic incremental ensemble fuzzy classifier for data streams in green internet of things, IEEE Trans. Green. Commun. Netw. 6 (3) (2022) 1316–1329.

[35] 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..

[36] X. Gu, G. Howells, H. Yuan, A soft prototype-based autonomous fuzzy inference system for network intrusion detection, Inf. Sci. (Ny.). 677 (2024) 120964.

[37] X. Gu, An explainable semi-supervised self-organizing fuzzy inference system for streaming data classification, Inf. Sci. (Ny.). 583 (2022) 364–385.

[38] J. Xue, F. Nie, R. Wang, X. Li, Iteratively re-weighted algorithm for fuzzy c-means, IEEE Trans. Fuzzy Syst. 30 (10) (2022) 4310–4321.

[39] X. Gu, A self-training hierarchical prototype-based approach for semi-supervised classification, Inf. Sci. (Ny.). 535 (2020) 204–224.

[40] E. Lughofer, P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Appl. Soft Comput. 11 (2) (2011) 2057–2068.

[41] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. (16) (2002) 321–357.

[42] G. Douzas, F. Bacao, F. Last, Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE, Inf. Sci. (Ny.). 465 (2018) 1–20.

[43] T. Zhu, Y. Lin, Y. Liu, Synthetic minority oversampling technique for multiclass imbalance problems, Pattern Recognit. 72 (2017) 327–340.

[44] X. Gu, Self-adaptive fuzzy learning ensemble systems with dimensionality compression from data streams, Inf. Sci. (Ny.). 634 (2023) 382–399.

[45] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, Inf. Secur. J. 25 (1–3) (2016) 18–31.

[46] I. Sharafaldin, A.H. Lashkari, and A.A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in *International Conference on Information Systems Security and Privacy*, 2018, pp. 108–116.

[47] A. Ferriyan, A.H. Thamrin, K. Takeda, J. Murai, Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic, Appl. Sci. 11 (17) (2021) 7868.

[48] Z.H. Zhou, M. Li, Tri-training: exploiting unlabeled data using three classifiers, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1529–1541.

[49] G. Huang, C. Du, The high separation probability assumption for semi-supervised learning, IEEE Trans. Syst. Man, Cybern. Syst. 52 (12) (2022) 7561–7573.

[50] Y.Y. Shen, Y.M. Zhang, X.Y. Zhang, C.L. Liu, Online semi-supervised learning with learning vector quantization, Neurocomputing 399 (2020) 467–478.

[51] S. Khezri, J. Tanha, A. Ahmadi, A. Sharifi, A novel semi-supervised ensemble algorithm using a performance-based selection metric to non-stationary data streams, Neurocomputing 442 (2021) 125–145.

[52] X. Gu, A dual-model semi-supervised self-organizing fuzzy inference system for data stream classification, Appl. Soft Comput. 136 (2023) 110053.

[53] K.H. Brodersen, C.S. Ong, K.E. Stephan, and J.M. Buhmann, “The balanced accuracy and its posterior distribution,” in *International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[54] D. Chicco, G. Jurman, The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation, BMC Genom. 21 (6) (2020) 1–13.