Engineering Applications of Artificial Intelligence 117 (2023) 105607

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

# Multiclass imbalanced and concept drift network traffic classification framework based on online active learning☆

Weike Liu, Cheng Zhu, Zhaoyun Ding, Hang Zhang, Qingbao Liu *

*Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, China*

ARTICLE INFO

*Keywords:*
Network traffic classification
Multiclass imbalance
Concept drift
Data stream
Online active learning

ABSTRACT

The complex problems of multiclass imbalance, virtual or real concept drift, concept evolution, high-speed traffic streams and limited label cost budgets pose severe challenges in network traffic classification tasks. In this paper, we propose a **m**ulticlass **i**mbalanced and **c**oncept drift network traffic classification **f**ramework based on **o**nline **a**ctive **l**earning (MicFoal), which includes a configurable supervised learner for the initialization of a network traffic classification model, an active learning method with a hybrid label request strategy, a label sliding window group, a sample training weight formula and an adaptive adjustment mechanism for the label cost budget based on a periodic performance evaluation. In addition, a novel uncertain label request strategy based on a variable least confidence threshold vector is designed to address the problems of a variable multiclass imbalance ratio or even the number of classes changing over time. Experiments performed based on eight well-known real-world network traffic datasets demonstrate that MicFoal is more effective and efficient than several state-of-the-art learning algorithms.

## 1. Introduction

Network traffic classification (NTC) plays an important role in network security maintenance and daily management. For example, through network traffic classification, network operators can assign priorities to different traffic applications and use appropriate technologies to set the quality of service (QoS) and security policies according to the specific needs of these applications to further improve the utilization of network resources and achieve high-quality differentiated services. However, with the emergence of new internet applications, the rapid development of network protocols, and the widespread use of encryption and mobile technologies, traditional network traffic classification methods are facing severe challenges (Iliyasu and Deng, 2020). In real-word network traffic classification applications, such as the distribution of the network traffic classes for Entry01 from the Moore dataset[1] in Fig. 1, proportions of network traffic classes are often imbalanced (i.e., the multiclass imbalance problem) (Zhu et al., 2017), and even individual classes will monopolize traffic in stages (such as traffic from No. 1715 to No. 8574 in Fig. 1). Due to the evolution of network technology, different application scenarios, and even deliberate artificial camouflaging, certain old traffic classes may disappear or new traffic classes may emerge (i.e., the concept evolution problem) (Masud et al., 2010). In addition, class concepts may change (i.e., the concept drift problem) (Widmer and Kubat, 1996), such as those seen in most phishing traffic or attack traffic involving network intrusion, and these traffic types constantly change their own network statistical characteristics to avoid the identification and interception of network supervision. Given the above challenges, the development of novel high-performance and low-cost network traffic classification methods remains a research hotspot in the related industries and academia.

In general, the current network traffic classification methods can be divided into three categories: port-based identification methods, deep packet inspection (DPI) methods and machine learning methods based on traffic statistics. In the initial stages of the internet, port-based network traffic classification methods were first proposed. As the Internet Assigned Numbers Authority (IANA) defines a corresponding public port number for each network application, the network traffic application can be directly identified. However, with the development of the internet, many emerging applications no longer have fixed port numbers and may use randomly generated ports. Therefore, port-based classification methods have become unreliable and inaccurate. Another method is based on DPI (De La Torre Parra et al., 2019; Elnawawy et al., 2020; Vu et al., 2016), which finds specific strings and patterns by inspecting the packet payload of network traffic and then classifies network traffic by matching it with a prestored feature library. However, due to privacy policies and legal reasons, DPI cannot be performed

☆ No author associated with this paper has disclosed any potential or pertinent conflicts that may be perceived to have impending conflict with this work.
* Corresponding author.
*E-mail addresses:* liuweike@nudt.edu.cn (W. Liu), zhucheng@nudt.edu.cn (C. Zhu), zyding@nudt.edu.cn (Z. Ding), zhanghang13@nudt.edu.cn (H. Zhang), liuqingbao@nudt.edu.cn (Q. Liu).
[1] https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/nprobe/data/papers/sigmetrics/index.html

https://doi.org/10.1016/j.engappai.2022.105607
Received 29 June 2022; Received in revised form 18 October 2022; Accepted 2 November 2022
Available online 24 November 2022

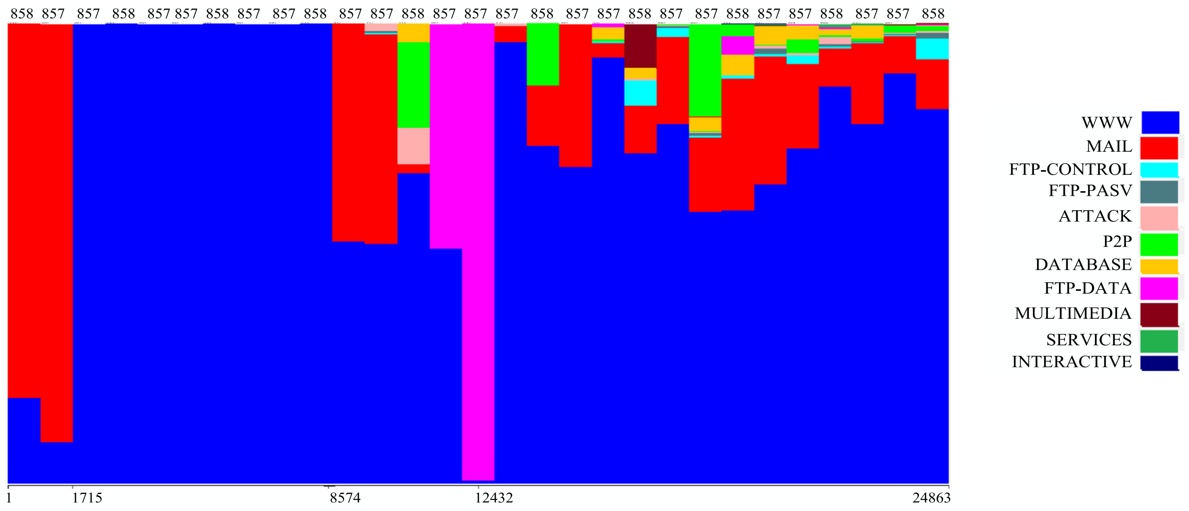

**Fig. 1.** Distribution of network traffic classes for Entry01 from the Moore dataset.

if access to the packet payload is disallowed. To overcome the inherent defects of the two abovementioned classification methods, researchers have increasingly been exploring network traffic classification based on machine learning.

Network traffic classification based on machine learning uses the packet information and statistical characteristics of network traffic (such as the transmission control protocol (TCP) window size, the average number of bytes transmitted, the mean interval between packet arrival times, etc.) to construct the training data of a machine learning model and then classify and identify unknown network traffic. Its advantages are that it does not rely on the analysis of network load content, it can be used for encrypted traffic classification, and it does not involve user privacy issues. Machine learning-based network traffic classification methods can be divided into supervised, unsupervised and semi-supervised learning methods. Supervised learning methods require a large number of network traffic training samples, and each training sample must be labelled with a real network traffic class. However, due to the large scale of network traffic data streams, high transmission speed and the emergence of new network traffic classes, the task of manually labelling each instance in a network traffic data stream becomes very difficult and entails an exorbitant cost. Unsupervised learning methods use the structural knowledge of the instance itself, such as the degree of dispersion, for clustering. However, such methods may also cause a large number of unlabelled instances to be misclassified because the characteristics of network traffic change over time, and such methods may not be able to accurately label the clustering results of new network classes. In the face of the high cost of training samples' labelling of supervised learning methods and the low classification performance of unsupervised learning methods, semi-supervised learning methods use existing labelled and unlabelled instances together to train classification models to obtain better classification performance than that attainable by learning only labelled instances. However, traditional semi-supervised learning methods (Carela-Español et al., 2015; Divakaran et al., 2015; Erman et al., 2007; Fahad et al., 2019) are too dependent on a small number of existing labelled instances and do not consider whether these existing network traffic class labels are representative and whether they change over time and across application scenarios.

As a viable alternative to semi-supervised learning, online active learning is a promising approach to addressing the joint challenges of concept drift, a variable multiclass imbalance ratio and a limited labelling budget. Active learning can improve the performance of a classification model by actively selecting the samples that are difficult to classify or that are representative by asking domain experts to label them (i.e., obtaining the ground truth labels) and then learning these samples with the true labels. With the help of new expert knowledge, in the task of network traffic classification, an active learning method can overcome the problem of overlearning the majority class samples and underlearning the minority class samples. In recent years, several active learning methods have been used for network traffic classification (Deka et al., 2019; Dong, 2021; Liu and Sun, 2014; Shahraki et al., 2021; Trittenbach et al., 2021; Wassermann et al., 2020; Yi-peng, 2013), however, most of these methods focus on concept drift problems (Wassermann et al., 2020) or imbalance problems (Dong, 2021; Torres et al., 2019; Trittenbach et al., 2021) and devote little attention to their combination.

To the best of our knowledge, no online active learning method exists for the above complex problems in network traffic classification tasks. One method with referential value is the comprehensive active learning method for multiclass imbalanced streaming data with concept drift (CALMID) (Liu et al., 2021), proposed recently to overcome the abovementioned challenges in general classification tasks. CALMID mainly consists of an ensemble learning framework, an uncertain active learning labelling strategy based on an asymmetric margin threshold matrix, and a class imbalance evaluation method under an online active learning strategy. The experimental results demonstrate that CALMID is effective and efficient on general classification tasks. However, network traffic classification is more difficult. First, in real-world network traffic, the phenomenon of concept evolution is common; i.e., the number of classes changes dynamically. Second, a network traffic class may persist for a long time; i.e., when only one class exists and there are no other so-called minority classes, the relative majority-vs.-minority relationship among multiple classes cannot be established, and thus the uncertain label request strategy based on an asymmetric threshold matrix in CALMID will be invalid.

Inspired by the simultaneous appearance of these issues and the lack of findings in the literature that remedy them, in this paper, we propose a multistage innovative framework that integrates active learning to blend the knowledge of human supervision with supervised learners that are based on a limited number of initially labelled samples. This framework includes a configurable supervised learner for the initialization of a network traffic classification model, an active learning method with a hybrid label request strategy, a label sliding window group, a sample training weight formula and an adaptive adjustment mechanism for the label cost budget based on a periodic performance evaluations. The main contributions of this research are as follows.

(1) The proposed framework provides flexibility regarding the selection of the underlying supervised learner or label query strategies and is composed of several separate stages that can be easily

Table 1
Frequently used mathematical notations.

| | |
|---|---|
| $d_i = (X_i, y_i)$ | Data instance |
| $X_i$ | Feature vector of data instance $d_i$ |
| $y_i$ | True label belonging to the label set $Y_{[0,t]}$= $\{c_1, c_2, \ldots, c_k\}$ |
| $IR$ | Imbalance ratio of $S_{[0,t]}$ |
| $L_{init}$ | Initialization learner |
| $S_{init}$ | Initial training sample set |
| $M_{ntc}$ | Network traffic classification model |
| $V$ | Variable least confidence threshold vector |
| $d = (X, y_u)$ | Unlabelled data instance |
| $y_u$ | Unknown label of data instance $d$ |
| $\tilde{y}$ | Predicted label of data instance $d$ |
| $y$ | True label of data instance $d$, which is given by export |
| $s_{tr}$ | Training sample, which is a triple $s_{tr} = (X, y, \tilde{y})$ |
| $W_t$ | True-label sliding window |
| $W_p$ | Predicted-label sliding window |
| $W_r$ | Random-label sliding window |
| $\varnothing$ | Empty placeholder |
| $\mu$ | Balance degree of the network traffic class |
| $w_{tr}$ | Training weight of the training sample $s_{tr}$ |
| $z$ | Size of the sliding window, also the length of the evaluation period |
| $b$ | Label cost budget, $0 < b \leq 1$ |
| $\varphi$ | Confidence in prediction result $\tilde{y}$, $0 \leq \varphi \leq 1$ |
| $\sigma$ | Label cost that has been spent, $0 \leq \sigma \leq 1$ |
| $h_{tf}$ | Indicator of whether $d$ is selected by the hybrid strategy |
| $r_{tf}$ | Indicator of whether $d$ is selected by the random label request strategy |
| $n$ | A network traffic stream with "-$n$ data instances |
| $n_{class}$ | Number of classes in the current period, which is equivalent to the number of distinct labels in the true-label sliding window $W_t$ |
| $n_{all}$ | Number of all labels in the random-label sliding window $W_r$ |
| $n_y$ | Number of the label $y$ in the random-label sliding window $W_r$ |
| $f_{dif}$ | Classification difficulty of $s_{tr}$ |
| $t$ | Timestamp |
| $T_r$ | Time required to predict an instance and to determine whether to request a true label based on the hybrid label request strategy |
| $T_a$ | Time required to add a pair of true labels and predicted labels into the sliding window group |
| $T_t$ | Time required for the classification model to learn a training sample online |
| $T_e$ | Computing time required for a periodic performance evaluation |
| $C$ | Large constant: although the number of classes in the entire network traffic stream can be a few, or several, dozen, or even in the hundreds, it is certainly finite; therefore, we might as well set it to a large constant $C$ |

modified or tuned. Thus, this framework is more generic and flexible than a dedicated approach.

(2) A novel uncertain label request strategy based on a variable least confidence threshold vector is designed. On the one hand, each least confidence threshold can be adjusted dynamically and independently according to the current characteristics of the corresponding class (such as the prediction difficulty or the balance degree). On the other hand, when new network traffic classes appear, the variable least confidence threshold vector expands easily.
(3) A novel sample training weight formula is designed that tends to assign larger training weights to indistinguishable and minority class samples.
(4) An adaptive adjustment mechanism of the label cost budget based on periodic performance evaluation is proposed that can reduce the label cost budget as much as possible while ensuring a high classification performance of the classification model.

The remainder of this paper is organized as follows. Section 2 provides an overview of the related studies of classification of multiclass imbalanced network traffic with concept drift. Section 3 describes in detail the online network traffic classification framework based on active learning, and Section 4 presents the experiments and a discussion. Finally, Section 5 concludes the paper.

In the rest of the paper, we follow the mathematical notations specified in Table 1.

## 2. Related studies

### 2.1. Joint challenge of multiclass imbalance and concept drift in network traffic classification

The problem of multiclass imbalance can cause learning bias towards the majority class and result in poor generalizability (Wang et al., 2017). Common methods used to address class imbalance in network traffic include data augmentation (Vu et al., 2017; Wang et al., 2019), sampling (Vu et al., 2016), cost-sensitive learning methods (Liu and Liu, 2012; Peng et al., 2017) and algorithms specifically designed for class imbalance (Guo et al., 2020; Peng et al., 2017; Wei et al., 2014). Wang et al. (2019) proposed a PacketCGAN model based on conditional generative adversarial networks (CGANs) to expand imbalanced encrypted traffic data. Vu et al. (2017) suggested using auxiliary classifier generative adversarial networks (AC-GANs) (Odena et al., 2016) to enhance network traffic data to address class imbalance. Vu et al. (2016) combined data generation and data sampling to address the imbalance in network traffic classification. Liu and Liu (2012) proposed a MetaCost approach that incorporated misclassification costs into the learning algorithm to mitigate multiclass imbalance based on the flow ratio and a resampling model that included undersampling and oversampling to make the multiclass training data more balanced. Peng et al. (2017) proposed not only a weighted cost matrix (WCM) that considered both the classification difficulty and class imbalance to represent the cost of different classes but also an imbalanced data gravitation-based classification (IDGC) method to address the class imbalance problem in network traffic classification. The IDGC method used an amplified gravitational coefficient (AGC) based on the degree of class imbalance to control the model's preference for different classes. Wei et al. (2014) proposed a comprehensive processing method for imbalanced network traffic by combining a sampling method, boosting in the ensemble learning method and a feature selection algorithm based on symmetric uncertainty to eliminate the impact of dynamic changes in traffic and to filter redundant features. Guo et al. (2020) proposed a focal loss-based adaptive gradient boosting ensemble learning method (FLAGB) to address the problem of network traffic imbalance. By integrating an adaptive tuning function, the FLAGB method could adaptively process network traffic data with any degree of class imbalance without a prior knowledge of class distribution. In addition, Gómez et al. (2019) comprehensively studied imbalance processing methods for network traffic classification, including a total of 28 methods that consisted of data-level techniques, ensemble learning algorithms and cost-sensitive algorithms. All of the above methods focused on imbalance processing, and the models were trained offline before deployment. However, in real-world environments, network traffic always changes rapidly and is characterized by virtual or real concept drift, which reduces the performance of a model trained based on historical knowledge (Wassermann et al., 2020).

As concept drift can significantly affect the stability of network traffic classification (Lu et al., 2020b), when the problems of concept drift and multiclass imbalance occur at the same time, similarly to the variable imbalance ratio (VarImb) case defined by Liu et al. (2021) as concept drift with a VarImb feature, network traffic classification tasks will become extremely challenging. In the VarImb case, class imbalance and concept drift can significantly affect classification performance, and they will tend to affect one another when they occur simultaneously (Wang et al., 2017). For example, traditional drift detection algorithms based on the classification error that are insensitive to class imbalance will become inefficient because they cannot detect concept drift in the minority class. In addition, methods that apply resampling or undersampling to address class imbalance will be affected because the class imbalance status varies in the case of concept drift with VarImb. To date, only a few methods have been proposed to address the problem of concept drift with class imbalance (Lu et al., 2020b; Wang et al., 2017). Most existing methods focus on binary imbalance problems; this is the case for methods proposed by Ditzler and Polikar (2013), Hoens and Chawla (2012), Yang et al. (2017), Lu et al. (2020b), Wang et al. (2013) and Wang et al. (2015), in which only one majority and one minority class exist. However, in multiclass imbalance cases, the relationship among the classes tends to be more complicated because a single class can act as a majority towards some and a minority towards other classes (Koziarski et al., 2020). Multiclass imbalance problems are considerably more challenging than binary imbalance problems because if decomposed into binary subproblems, they will suffer from a loss of information about class relationships (García et al., 2018; Koziarski et al., 2020). Only a few researchers have proposed algorithms targeted at multiclass imbalanced data streams with concept drift. The relevant studies focus either on virtual concept drift with VarImb (Wang et al., 2016) or on concept drift with a fixed imbalance ratio (Mirza et al., 2015). The meta-cognitive online sequential extreme learning machine (MOS-ELM) (Mirza and Lin, 2016) was the first method targeted at managing the problem of mixed concept drift with a variable multiclass imbalance ratio. However, MOS-ELM was a chunk-based approach that could not react to concept drift that occurred inside a single chunk. Adaptive random forests with resampling (ARFre) (Ferreira et al., 2019) were intended to solve the problem of a simultaneous occurrence of concept drift and multiclass imbalance. ARFre was a one-by-one method that inherited the concept drift detection mechanism in adaptive random forests (ARF) presented by Gomes et al. (2017) and resampled the instances based on the current class label distribution. However, if class roles were inverted (i.e., the majority class became a minority class), its estimation of the current multiclass imbalance ratio would be influenced by a cumulative effect. Overall, all of the above solutions are supervised learning methods that assume unrestricted access to class labels during training. However, because of the large scale and high speed of network traffic, network traffic classification faces labelling difficulties. Therefore, several semi-supervised learning methods for network traffic classification tasks have been proposed that can use unlabelled instances and help address the lack of labelled samples (Pacheco et al., 2019). Erman et al. (2007) clustered labelled samples and unlabelled instances together. The unlabelled instances were then labelled according to the clustering results. Divakaran et al. (2015) and Fahad et al. (2019) applied a self-learning method that learned a small part of labelled samples and subsequently summarized the labels of unlabelled instances. Although the above semi-supervised methods can use the unlabelled instances depending on a small number of labelled samples, they do not consider whether the labels are representative or change over time and scenes, which could lead to an overfitting problem due to the excessive dependence on a small number of labelled samples.

### 2.2. *Active learning in network traffic classification method*

In recent years, an increasing number of active learning methods have been applied to classification of network traffic data streams (Deka et al., 2019; Dong, 2021; Liu and Sun, 2014; Shahraki et al., 2021; Torres et al., 2019; Trittenbach et al., 2021; Wassermann et al., 2020; Yi-peng, 2013). Yi-peng (2013) proposed an active learning network traffic classification method based on a support vector machine (SVM). By selecting samples for learning through an active learning method based on uncertainty, Yi-peng (2013) proposed a method that could reduce the number of repeated and meaningless samples in SVM model learning, reduce the annotation time and improve the efficiency of model learning. Liu and Sun (2014) used an active learning strategy to identify peer-to-peer (P2P) network traffic. In each round of the active learning model, only the unlabelled instances located in the vicinity of the current learner's hyperplane were labelled because their uncertainty was higher. Deka et al. (2019) proposed a pool-based active learning algorithm and illustrated its efficiency and high precision in recent distributed denial-of-service (DDoS) scenarios. In this approach, the learning model trained a set of SVM classifiers. From each support vector provided by the SVM classifier, the Euclidean distance was computed for every instance in the batch. Instances with the shortest and longest distances from each support vector were selected to merge with the training samples for the SVM classifier. Wassermann et al. (2020) proposed an adaptive reinforcement learning method for online network monitoring and analysis. It included an adaptive memory (ADAM) strategy-based concept drift detection algorithm and reinforcement active learning (RAL). RAL combined reinforcement learning with stream-based active learning to dynamically select the most representative samples, thereby reducing the number of labelled samples required for the training model. However, ADAM and RAL were designed and evaluated separately. Torres et al. (2019) proposed an active learning approach that used random forests to interactively assist the user in labelling botnet network traffic datasets. Khanchi et al. (2018) described an incremental active learning framework for streaming botnet traffic analysis. The researchers compared the performance of Hoeffding tree, naïve Bayes and genetic programming (GP) machine learning algorithms and contrasted different active learning labelling strategies under a limited label cost budget. Dong (2021) proposed a cost-sensitive SVM (CMSVM) to solve the imbalance problem in network traffic identification. CMSVM was an off-line active learning algorithm that learned the labelled network flow samples, and subsequently classified all unlabelled instances and determined the maximum iteration count. If the requirement of the CMSVM algorithm was satisfied, CMSVM ended. Otherwise, the unlabelled instances close to the cost-sensitive SVM hyperplane were selected by a label query strategy for active learning. Overall, however, most of the above-mentioned methods focused on concept drift problems (Wassermann et al., 2020) or imbalance problems (Dong, 2021; Torres et al., 2019; Trittenbach et al., 2021) and devoted little attention to the case of their coexistence.

If label cost budgets are limited, online active learning classification tasks for network traffic data streams with coexisting multiclass imbalance and concept drift will be extremely challenging. Although the recently proposed CALMID (Liu et al., 2021) is designed to solve these problems, according to the analysis in Section 1, it is difficult for CALMID to address the problems of exclusive traffic and the scenario of an increase or a reduction in the number of classes.

As an improvement of CALMID for NTC application scenarios, MicFoal has following advantages: (1) MicFoal is more generic and flexible than CALMID. The framework of MicFoal is flexible as to the selection of the underlying supervised learner or label query strategies and is composed of several separate stages that can be easily modified or tuned. (2) A novel uncertain label request strategy based on a variable least confidence threshold vector is designed. When new network traffic classes appear, the variable least confidence threshold vector expands easily. (3) An adaptive adjustment mechanism for the label cost budget based on a periodic performance evaluation is proposed that can reduce the label cost budget as much as possible while ensuring a high classification performance of the classification model.

## 3. Method

In this section, we propose a multiclass imbalance and concept drift network traffic classification framework based on online active learning (MicFoal). The content of this section is arranged as follows. Section 3.1 presents a formal description of the problem and the motivation of MicFoal, Section 3.2 outlines the overall framework and processing flow of MicFoal, and Section 3.3 discusses the hybrid label request strategy composed of an uncertain label request strategy based on a variable least confidence threshold vector, a selective label request strategy and a random label request strategy. Section 3.4 introduces a novel sample weight formula, and Section 3.5 describes an adaptive adjustment mechanism for the label cost budget based on a periodic performance evaluation. Finally, the temporal and spatial complexity of MicFoal is analysed in Section 3.6.

### 3.1. Problem description and motivation

This subsection begins with a formal description of complex problems in network traffic classification described in Section 1.

Given a network traffic data stream $S$ and a time period $[0, t]$, a set of samples is denoted by $S_{[0,t]} = \{d_0, d_1, \ldots, d_t\}$, where $d_i = (X_i, y_i)$ is one observation (or an instance), $X_i$ is the feature vector, $y_i$ is the true label belonging to the class label set $Y_{[0,t]} = \{c_1, c_2, \ldots, c_k\}$, $k = |Y_{[0,t]}|$ is the number of classes in $S_{[0,t]}$, and $S_{[0,t]}$ follows a certain distribution $F_{[0,t]}(X, y)$.

(1) Let $c_{max} = \underset{c_j \in Y_{[0,t]}}{\arg\max}\{|\{(X_i, y_i) : (X_i, y_i) \in S_{[0,t]}, y_i = c_j\}|\}$, $c_{min} = \underset{c_j \in Y_{[0,t]}}{\arg\min}\{|\{(X_i, y_i) : (X_i, y_i) \in S_{[0,t]}, y_i = c_j\}|\}$, and $IR = \frac{|\{(X_i,y_i) : (X_i,y_i) \in S_{[0,t]}, y_i = c_{max}\}|}{|\{(X_i,y_i) : (X_i,y_i) \in S_{[0,t]}, y_i = c_{min}\}|}$. If $IR \gg 1$ and $|Y_{[0,t]}| > 2$, then $S_{[0,t]}$ is a multiclass imbalanced set, $c_{max}$ is the label of the class with the maximum proportion in $S_{[0,t]}$, and $c_{min}$ is the label of the class with the minimum proportion in $S_{[0,t]}$. Here, $IR$ is the imbalance ratio of $S_{[0,t]}$, which is an important indicator that describes how skewed the learning problem is. Smaller values of $IR$ correspond to a greater imbalance of $S_{[0,t]}$.

(2) Concept drift occurs at timestamp $t + 1$ if $F_{[0,t]}(X, y) \neq F_{[t+1,\infty]}(X, y)$; this is denoted as $\exists t$: $P_t(X, y) \neq P_{t+1}(X, y)$ (Lu et al., 2020b, 2016). Concept drift can occur simultaneously with the change in the status of class imbalance, which is denoted as $\exists t$: $P_t(y) \neq P_{t+1}(y)$ and $P_t(X, y) \neq P_{t+1}(X, y)$. In this case, the roles of the minority and the majority may switch (Liu et al., 2021).

(3) An individual class monopolizes the entire network traffic for a period $[t, t + \Delta t]$; this is denoted as $\exists t, \Delta t$: $|Y_{[t,t+\Delta t]}| = 1$ and $\Delta t \gg 1$.

(4) Concept evolution occurs at timestamp $t + 1$ if $y_{t+1}$ is the label $c_{new}$ of a new class just emerging at time $t + 1$; this is denoted as $\exists t$: $Y_{[0,t+1]} = Y_{[0,t]} \cup \{c_{new}\}$ (Masud et al., 2010).

(5) As $t \to \infty$, $S_{[0,t]}$ is a continuous network traffic data stream $S$. Therefore, it is impossible to obtain all true class labels for $S_{[0,t]}$.

As described in Section 1, the CALMID can solve the situation when problems (1) and (3)) occur simultaneously, but it cannot solve problems (3) and (4). To solve problems (1) to (4), inspired by the idea of an asymmetric margin threshold matrix proposed in CALMID, we design a novel uncertain label request strategy based on a variable least confidence threshold vector that sets a minimum confidence threshold for each class of network traffic that can be dynamically adjusted according to its characteristics so that the model can still maintain high classification performance after the occurrence of problem (3). Furthermore, with the emergence of new network traffic classes, this vector structure can also be easily extended synchronously.

To address all of problems (1) to (5) above, we propose an innovative multistage solution that integrates active learning to blend the knowledge of human supervision with supervised learners that are based on initially limited labelled data. The framework and process of this innovative solution are described in Section 3.2.

### 3.2. Novel framework and process

In this subsection, a novel network traffic classification framework based on online active learning and the working process are introduced. As shown in Fig. 2, the proposed MicFoal method proceeds as follows.

(1) The initial stage is a supervised learning process that obtains the initial network traffic classification model $M_{ntc}$. A state-of-the-art supervised learning algorithm is configured as the initialization learner $L_{init}$. Let $L_{init}$ learn the initial training sample set $S_{init}$, which is formed by labelling a certain number of instances in the beginning of the network traffic data stream $S$. Then, we will regard the classification model obtained after the initial learning of $L_{init}$ as the network traffic classification model $M_{ntc}$. The initial size of the variable least confidence threshold vector $V$ is the number of classes in the initial training sample set $S_{init}$. Considering the default value of 0.9 of the least confidence threshold in ALUncentry.java in the Massive Online Analysis (MOA) data stream software suite (Bifet, 2010), we also set the initial value of all elements in $V$ to 0.9.

(2) For a newly arrived unlabelled data instance $d = (X, y_u)$ in the network traffic data stream $S$ (where $y_u$ indicates that the label of data instance $d$ is unknown), the network traffic classification model $M_{ntc}$ provides a predicted label $\tilde{y}$ for $d$ and outputs the prediction results online.

(3) The hybrid label request strategy is used to determine whether the current instance $d$ is selected to request the true label $y$.

(4) If data instance $d$ is requested by the hybrid label request strategy, then an expert will provide $d$'s true label $y$ and combine $y$, $\tilde{y}$ and $d$ itself to form a training sample $s_{tr}$, which is a triple $s_{tr} = (X, y, \tilde{y})$.

(5) If data instance $d$ is requested by the hybrid label request strategy, then $y$ and $\tilde{y}$ are added to the true-label sliding window $W_t$ and the predicted-label sliding window $W_p$, respectively. If $y$ is the label of a new traffic class, the variable least confidence threshold vector $V$ will be expanded. Furthermore, if instance $d$ is requested by the random label request strategy in the hybrid label request strategy, then $y$ will also be added to the random-label sliding window $W_r$; otherwise, an empty placeholder $\varnothing$ will be added to the random-label sliding window $W_r$. If this instance $d$ is not requested by the hybrid label request strategy, then the empty placeholder $\varnothing$ will be added to the three label sliding windows $W_t$, $W_p$ and $W_r$.

(6) The balance degree $\mu$ of the network traffic class is dynamically estimated by using the random-label sliding window $W_r$. The training weight $w_{tr}$ of the training sample $s_{tr}$ is calculated.

(7) The network traffic classification model $M_{ntc}$ is updated by the online learning of the weighted training sample $s_{tr}$.

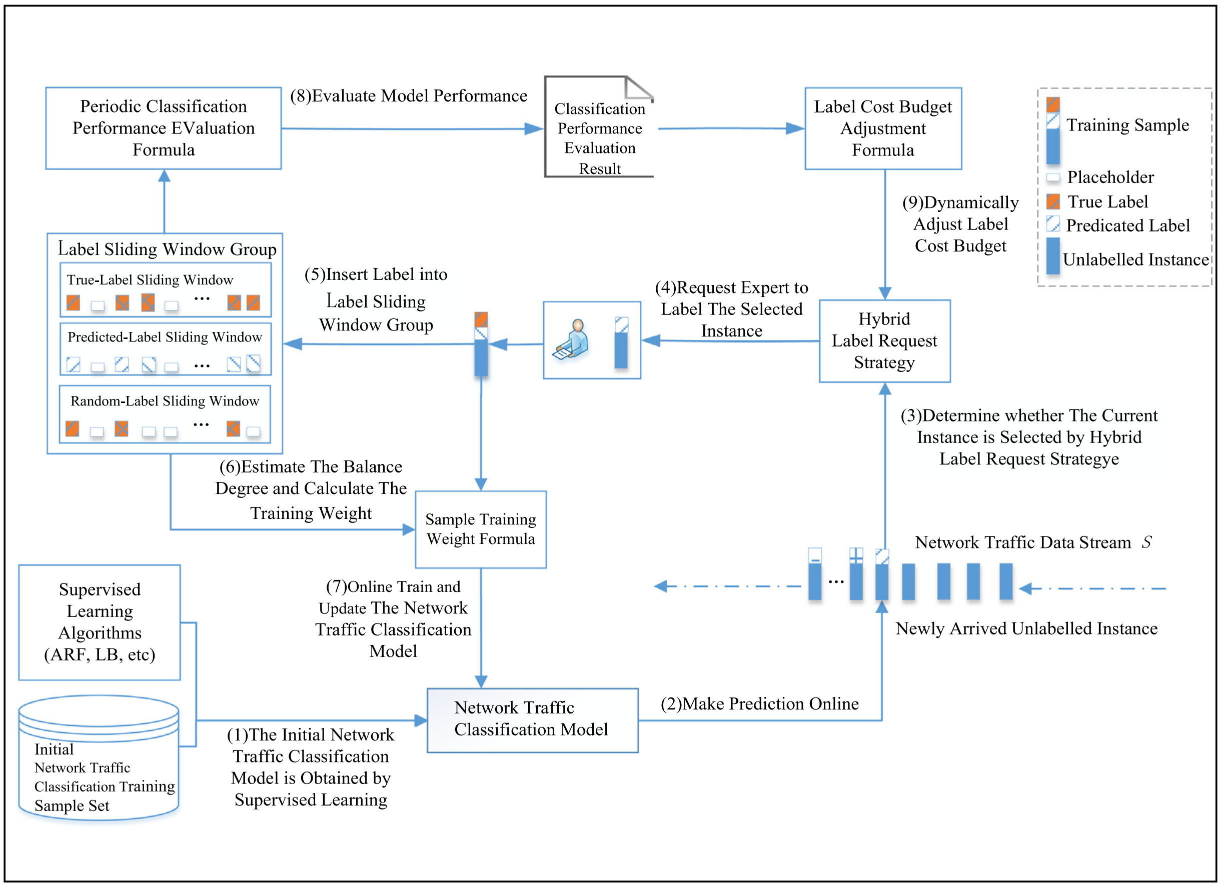
Periodic Classification Performance EValuation Formula
(8)Evaluate Model Performance
Classification Performance Evaluation Result
Label Cost Budget Adjustment Formula
Training Sample
Placeholder
True Label
Predicated Label
Unlabelled Instance
(9)Dynamically Adjust Label Cost Budget
Label Sliding Window Group
True-Label Sliding Window
Predicted-Label Sliding Window
Random-Label Sliding Window
(5)Insert Label into Label Sliding Window Group
(4)Request Expert to Label The Selected Instance
Hybrid Label Request Strategy
(3)Determine whether The Current Instance is Selected by Hybrid Label Request Strategye
(6)Estimate The Balance Degree and Calculate The Training Weight
Sample Training Weight Formula
Network Traffic Data Stream S
Newly Arrived Unlabelled Instance
Supervised Learning Algorithms (ARF, LB, etc)
(7)Online Train and Update The Network Traffic Classification Model
Initial Network Traffic Classification Training Sample Set
(1)The Initial Network Traffic Classification Model is Obtained by Supervised Learning
Network Traffic Classification Model
(2)Make Prediction Online

**Fig. 2.** Novel network traffic classification framework based on online active learning.

(8) According to the evaluation period (the period length is generally set to the size $z$ of the true-label sliding window $W_t$), the periodic performance evaluation is calculated based on the true-label sliding window $W_t$ and predicted-label sliding window $W_p$.
(9) Given the performance evaluation's results, the label cost budget $b$ of active learning is adjusted dynamically according to the label cost budget adjustment formula.

Algorithm 1 shows the pseudocode for the processing procedure of the proposed MicFoal method.

The following is a brief explanation of some of the functions used in Algorithm 1:

- *LabellingIniTrainingSet*$(S, z)$ — The first $z$ instances in data stream $S$ are labelled.
- *IniThresholdVector*$(S_{init})$ — The initial size of $V$ is the number of classes in $S_{init}$, and the value of all elements in vector $V$ is 0.9.
- *IniTrainingModel*$(L_{init}, S_{init})$ — The initial model $M_{ntc}$ is trained by using $L_{init}$ and $S_{init}$.
- *HybridStrategy*$(d, \tilde{y}, \varphi, V, \sigma, b)$ — Determine whether instance $d$ is requested by a hybrid label request strategy. Details are provided in Section 3.3.
- *CalculateTrainingWeight*$(s_{tr}, \varphi, V)$ — Calculate and return the training weight $w_{tr}$ of the training sample $s_{tr}$ and the balance degree $\mu$ of the class corresponding to the true label $y$. The detailed calculation process is shown in Algorithm 3 in Section 3.4.
- *AdjustThresholdVector*$(V, s_{tr}, u_{tf}, \mu)$ — The values of the elements in the least confidence threshold vector $V$ can be dynamically adjusted. If $d$ is predicted correctly (i.e., $y$ equals $\tilde{y}$) but $u_{tf}$ is true (i.e., $d$ is requested by the uncertain label request strategy based on the variable least confidence threshold vector), then the value of the element $V[\tilde{y}]$ will be reduced according to Formula (1).

$$V\left[\tilde{y}\right] = (1-0.1)^q \times V\left[\tilde{y}\right] \quad (1)$$

where

$$q = \begin{cases} 2, & 1 \leq \mu \\ 1, & 0 \leq \mu < 1 \end{cases} \quad (2)$$

Otherwise, if $d$ is predicted incorrectly and is not requested by the uncertain label request strategy (that is, $u_{tf}$ is false), then the value of element $V[\tilde{y}]$ will be increased according to Formula (3).

$$V\left[\tilde{y}\right] = (1+0.1) \times V\left[\tilde{y}\right] \quad (3)$$

The balance degree $\mu$ in Formula (2) can be calculated in lines 10–13 of Algorithm 3 in Section 3.4. If $0 \leq \mu < 1$, the class corresponding to the true label $y$ is a minority class; $\mu = 1$ means that the class corresponding to the true label $y$ is a balanced class; $\mu > 1$ indicates that the class corresponding to the true label $y$ is a majority class.

After the adjustment of Formulas (1) and (3) to the least confidence threshold vector $V$, different network traffic classes can have different least confidence thresholds. In general, the least confidence threshold of the majority class is low and that of the minority class is high.

- *IsNewClass(y)* — Determine whether $y$ is the true label of a new class. If yes, *IsNewClass*$(y)$ returns true; otherwise, *IsNewClass*$(y)$ returns false.
- *AdjustBudgetbyPeriodEval*$(b)$ — Adjust the budget $b$ based on periodic classification performance evaluations. Details are provided in Section 3.5.

**Algorithm 1.** *Predicting then Active Learning*

```
Inputs:
$S$ - a network traffic data stream
$L_{init}$ - the state-of-the-art supervised learning algorithm
$z$ - the size of the label sliding window, and the length of the evaluation period
Local variables:
$d$ - an unlabelled data instance in $S$, denoted as $d = (X, y_u)$
$X$ - the feature vector of data instance $d$; $y_u$ - the unknown label of data instance $d$
$S_{init}$ - the initial training sample set; $M_{ntc}$ - the network traffic classification model
$b$ - the label cost budget; $\sigma$- the label cost that has been spent
$\tilde{y}$ - the predicted label of data instance $d$; $y$ - the true label of data instance $d$; $\emptyset$- an empty placeholder
$\varphi$ - the confidence in prediction result $\tilde{y}$; $V$ - the least confidence threshold vector
$s_{tr}$ - the training sample $(X, y, \tilde{y})$; $w_{tr}$ - the training weight of $s_{tr}$; $\mu$ - the balance degree of the network traffic class
$h_{tf}$ - whether $d$ is selected by the hybrid strategy; $r_{tf}$ - whether $d$ is selected by the random label request strategy
$u_{tf}$ - whether $d$ is selected by the uncertain label request strategy based on the variable least confidence threshold vector
$W_t$ - the true-label sliding window; $W_p$ - the predicted-label sliding window; $W_r$ - the random-label sliding window
Initialization:
$S_{init} \leftarrow LabellingIniTrainingSet(S, z)$
$V \leftarrow IniThresholdVector(S_{init})$
$M_{ntc} \leftarrow IniTrainingModel(L_{init}, S_{init})$
$b \leftarrow 0.2$
$\sigma \leftarrow 0$
Process:
1. for each $d$ in $S$ do
2.   $\tilde{y}, \varphi \leftarrow Predict(d, M_{ntc})$   //use $M_{ntc}$ to predict $d$, and obtain the $\tilde{y}$ and $\varphi$
3.   $h_{tf}, u_{tf}, r_{tf} \leftarrow HybridStrategy(d, \tilde{y}, \varphi, V, \sigma, b)$
4.   if $h_{tf}$ then   // if $d$ is selected by the hybrid strategy
5.     ask the expert for the true label $y$ of $d$
6.     update the label cost $\sigma$ that has been spent
7.     $s_{tr} \leftarrow (X, y, \tilde{y})$
8.     add $y$ to the true-label sliding window $W_t$
9.     add $\tilde{y}$ into the predicted-label sliding window $W_p$
10.    if $r_{tf}$ then   // if $d$ is selected by the random label request strategy
11.      add $y$ to the random-label sliding window $W_r$
12.    else
13.      add $\emptyset$ to the random-label sliding window $W_r$
14.    end if
15.    $w_{tr}, \mu \leftarrow CalculateTrainingWeight(s_{tr}, \varphi, V)$
16.    use $s_{tr}$ with $tw$ to update $M_{ntc}$
17.    $V \leftarrow AdjustThresholdVector(V, s_{tr}, u_{tf}, \mu)$
18.    if $IsNewClass(y)$ then
19.      increase the size of $V$
20.    end if
21.  else
22.      add $\emptyset$ to the three label sliding windows $W_t$, $W_p$ and $W_r$
23.  end if
24.  if $ReachEvaluationPeriod(\,)$ then   //check whether an evaluation period has been reached,
25.    $b \leftarrow AdjustBudgetbyPeriodEval(b)$
26.  end if
27. end for
```

### 3.3. Hybrid label request strategy

Compared with general multiclass imbalanced data streams, a network traffic data stream has its own unique features. For example, an individual traffic class may monopolize the entire network traffic for a period of time, as shown in Fig. 1. In this case, the multiclass imbalance problem of network traffic may degenerate into a single-class problem; i.e., only a single class may exist, and there may be no other so-called minority class. Therefore, the relative majority-vs.-minority relationship among multiple classes cannot be established in this case, and thus the uncertain label request strategy based on an asymmetric threshold matrix in CALMID will fail.

To address the above drawbacks of CALMID in classification of network traffic data streams, we propose a novel uncertain label request strategy based on a variable least confidence threshold vector. Algorithm 2 shows the pseudocode for the hybrid label request strategy, which is composed of the uncertain label request strategy based on the variable least confidence threshold vector, the selective label request strategy and the random label request strategy.

Lines 3–5 in Algorithm 2 show the processing procedure of the uncertain label request strategy based on the variable least confidence threshold vector $V$. If $\varphi \le V[\tilde{y}]$, i.e., if the confidence $\varphi$ of the prediction result $\tilde{y}$on data instance $d$ is less than the least confidence threshold that corresponds to $\tilde{y}$, which indicates that the prediction result $\tilde{y}$on data instance $d$ is uncertain, $d$ must be requested by the uncertain label request strategy and its true label $y$ must be provided by experts.

Lines 6–12 in Algorithm 2 show the selective label request strategy. If $\varphi > V[\tilde{y}]$ (i.e., the confidence $\varphi$ of the prediction result $\tilde{y}$ on data instance $d$ is higher than the least confidence threshold that corresponds to $\tilde{y}$) $p_{select}$ is calculated by using the formula shown in line 7 in Algorithm 2, and then a random value $\zeta_{select}$ is generated by comparing $p_{select}$ and $\zeta_{select}$ to determine whether $d$ will be selected by the selective label request strategy. The selective label request strategy can select some instances that are not selected by the uncertain label request strategy and provide them to experts for manual labelling if there is still a remaining label budget. Generally, if the remaining label budget ($b$ - $\sigma$) is large and the difference between the confidence $\varphi$ of prediction result $\tilde{y}$ on data instance $d$ and the least confidence threshold that corresponds to $\tilde{y}$ is small, data instance $d$ is more likely to be selected by the selective label request strategy. As the value of $(\varphi - V[\tilde{y}])$ for minor class instances is often smaller than that for major class instances, the value of $p_{select}$ calculated by using the formula shown in line 7 in Algorithm 2 is often larger for minor class instances than for major class instances. Therefore, under the same remaining label budget $(b-\sigma)$, the selective label request strategy tends to select the minor class instances.

Lines 14–18 in Algorithm 2 represent the random label request strategy. When a new instance $d$ arrives in the data stream, a random number $\zeta_{random}$ is generated and compared with the initially set random selection ratio $\varepsilon$. If $\zeta_{random} < \varepsilon$,data instance $d$ is selected by the random label request strategy.

**Algorithm** 2. *Hybrid Strategy*

```
Inputs:
$d$ - a data instance in network traffic data stream $S$
$\tilde{y}$ - the predicted label of $d$
$\varphi$ - the confidence for the prediction result $\tilde{y}$
$V$ - the least confidence threshold vector
$\sigma$ - the label cost that has been spent
$b$ - the label cost budget
Local variables:
$\varepsilon$ - the random selection ratio, set to $\varepsilon = 0.1$; $\zeta_{select}$ - a random variable, $\zeta_{select} \in [0, 1]$
$\zeta_{random}$ - a random variable, $\zeta_{random} \in [0, 1]$; $p_{select}$ - probability of the selective label request strategy
Output:
$h_{tf}$ - whether $d$ is selected by the hybrid strategy
$u_{tf}$ - whether $d$ is selected by the uncertain label request strategy based on the variable least confidence threshold vector
$r_{tf}$ - whether $d$ is selected by the random label request strategy
Function HybridStrategy($d$, $\tilde{y}$, $\varphi$, $V$, $\sigma$, $b$)
1. $h_{tf}$, $u_{tf}$, $r_{tf} \leftarrow$ false
2. if $\sigma \leq b$ then
3.    if $\varphi \leq V[\tilde{y}]$ then
4.       $u_{tf} \leftarrow$ true   //requested by the uncertain label request strategy based on the variable least confidence threshold vector
5.       $h_{tf} \leftarrow$ true
6.    else
7.       $p_{select} \leftarrow (b - \sigma) / (b - \sigma + \varphi - V[\tilde{y}])$
8.       $\zeta_{select} \leftarrow$ generate a random value
9.       If $\zeta_{select} < p_{select}$ then   //requested by the selective label request strategy
10.         $h_{tf} \leftarrow$ true
11.      end if
12.   end if
13. end if
14. $\zeta_{random} \leftarrow$ generate the next random value
15. if $\zeta_{random} < \varepsilon$ then   // requested by the random label request strategy
16.    $h_{tf} \leftarrow$ true
17.    $r_{tf} \leftarrow$ true
18. end if
19. return $h_{tf}$, $u_{tf}$, $r_{tf}$
```

**Algorithm** 3. *Calculate Training Weight*

```
Inputs:
$s_{tr}$ - a training sample $(X, y, \tilde{y})$
$\varphi$ - the confidence for the prediction result $\tilde{y}$
$V$ - the least confidence threshold vector
Local variables:
$n_{class}$ - the number of classes in the current period, which is the number of distinct labels in the true-label sliding window $W_t$; $n_{all}$ - the number of all labels in the random-label sliding window $W_r$;
$n_y$ - the number of the label $y$ in the random-label sliding window $W_r$;
$f_{dif}$ - the classification difficulty of the training sample $s_{tr}$
Output:
$w_{tr}$ - the training weight of the training sample $s_{tr}$
$\mu$ - the balance degree of the class corresponding to true label $y$;
Function CalculateTrainingWeight($s_{tr}$, $\varphi$, $V$)
1. $n_{class} \leftarrow$ count the number of distinct labels in the true-label sliding window $W_t$
2. $n_{all} \leftarrow$ count the number of all labels in the random-label sliding window $W_r$
3. $n_y \leftarrow$ count the number of the label $y$ in the random-label sliding window $W_r$
4. if $y$ equals $\tilde{y}$ then
5.    $f_{dif} \leftarrow 1 + V[y] - \varphi$
6. else
7.    $f_{dif} \leftarrow 2*(1 + n_{class})$
8. end if
9. if $y$ does not exist in $W_r$ then
10.   $\mu \leftarrow 0$
11.   $w_{tr} \leftarrow 1 + log(1 + f_{dif} + n_{all})$
12. else
13.   $\mu \leftarrow (n_y * n_{class}) / n_{all}$
14.   $w_{tr} \leftarrow 1 + log(1 + f_{dif} + 1/\mu)$
15. end if
16. return $w_{tr}$, $\mu$
```

### 3.4. Sample training weight formula

Due to the characteristics of network traffic transmission protocols or the real-world needs of network applications, there may be dozens of traffic classes in the entire network traffic data stream; however, during a given period time, traffic instances belonging to only a few classes or even a single class may be transmitted over the network. Due to these characteristics of the network traffic, we consider the number of traffic classes in the latest period, the balance degree of traffic classes and the prediction difficulty of the model for the sample as the influencing factors to comprehensively design the sample training weight formula. Algorithm 3 shows the pseudocode for the calculation process of the training sample weight.

Lines 4–5 in Algorithm 3 show that in the case of correct prediction (i.e., $y$ equals $\tilde{y}$), the following holds. If $\varphi > V[y]$, the classification model is certain for the predicted result $\tilde{y}$, and the classification difficulty $f_{dif}$ calculated by line 5 is less than 1, which means that the prediction difficulty of the classification model on data instance $d$ is lower. In contrast, if $\varphi \leq V[y]$, then the classification model is uncertain as to the prediction result $\tilde{y}$, which causes the data instance $d$ to be requested by the uncertain label request strategy based on the variable least confidence threshold vector; thus, the prediction difficulty $f_{dif}$ calculated by line 5 is greater than 1 and less than 2.

In the case of incorrect prediction (i.e., $y \neq \tilde{y}$), line 7 shows that the value of $f_{dif}$ for the prediction is greater than 2, and for a greater number of classes in the current period, the prediction difficulty $f_{dif}$ will be greater as well.

Next, the balance degree $\mu$ of the class corresponding to label $y$ is calculated in lines 10 and 13. The value of $\mu$ has the following meanings.

- If $\mu = 0$, the class corresponding to label $y$ is a minimal minority class or a new class
- If $0 < \mu < 1$, the class corresponding to label $y$ is a minority class
- If $\mu = 1$, the class corresponding to label $y$ is a balanced class
- If $\mu > 1$, the class corresponding to label $y$ is a majority class

Finally, the training weight $w_{tr}$ of sample $s_{tr}$ is calculated in line 11 or line 14. As Algorithm 3 shows, the indistinguishable and minority class samples will be assigned larger training weights.

The final weight of a training sample merits a further explanation here. In step (1) in Fig. 2, we configure a state-of-the-art supervised learning algorithm (ARF or LB) as the initialization learner $L_{init}$. Both ARF and LB use $Poisson(\lambda = 6)$ to increase the number of times the same instance; hence, our sample training weight formulas in lines 11 and 14 do not involve a multiplication by $Poisson(\lambda)$; we let "$w_{tr} \leftarrow 1 + log(1 + f_{dif} + n_{all})$" in line 11 or "$w_{tr} \leftarrow 1 + log(1 + f_{dif} + 1/\mu)$" in line 14 in Algorithm 3. When the current weighted sample is learned by a supervised learner (the ARF or LB algorithm) to update the network traffic classification model $M_{ntc}$ in step (7) in Fig. 2, the weight of this current weighted sample will be multiplied by $Poisson(\lambda = 6)$ according to the weight formula of the ARF or LB algorithm.

### 3.5. Adaptive adjustment mechanism of the label cost budget based on a periodic performance evaluation

To enable the model to achieve the expected classification performance and reduce the label cost budget as much as possible, we propose an adaptive adjustment mechanism for the label cost budget based on periodic classification performance evaluations. The default metric for our periodic performance evaluation is the average $F1$-score of all classes in the current period.

The classes in the current period correspond to the distinct labels in the true-label sliding window $W_t$. Using these true labels and predicted labels cached in the true-label sliding window $W_t$ and the predicted-label sliding window $W_p$, respectively, the average $F1$-score of all classes in the current period can be easily calculated.

Furthermore, based on performance evaluation results, the label cost budget is adjusted dynamically. If the value of the average $F1$-score is below the lower threshold of the benchmark evaluation value, the label cost budget will be increased. Conversely, when the value of the average $F1$-score is above the upper threshold of the benchmark evaluation value, the label cost budget will be reduced appropriately to reduce the label cost.

### 3.6. Analysis of time complexity and spatial complexity

The time complexity and spatial complexity of the proposed MicFoal method are analysed as follows.

(1) Time complexity. Let $T_r$ be the time required to predict an instance and to determine whether to request a true label based on the hybrid label request strategy, $T_a$ be the time required to add a pair of true labels and predicted labels into the sliding window group, $T_t$ be the time required for the classification model to learn a training sample online, and $T_e$ be the computing time required for a periodic performance evaluation. According to Algorithm 1, the length of the evaluation period is $z$, and the label cost budget is $b$. For a network traffic stream with -$n$ data instances, the time complexity is $O(n \times T_r + n \times b \times (T_a + T_t) + T_e \times n/z)$, i.e., $O(n \times (T_r + b \times (T_a + T_t) + T_e / z))$. Because the value of $(T_r + b \times (T_a + T_t) + T_e / z)$ is a constant, the time consumption of the proposed MicFoal method to address an unbounded network traffic data stream is linearly related to the number $n$ of arrival data instances.

(2) Spatial complexity. As the MicFoal works in a one-by-one processing manner, only a small amount of space is needed to address the continuously arriving network traffic data stream. The auxiliary space required by the proposed MicFoal method includes three sliding windows (a true-label sliding window, predicted-label sliding window and random-label sliding window) and a variable least confidence threshold vector $V$. According to Algorithm 1, the size of each sliding window is $z$, and the size of $V$ is the number of classes in the entire network traffic stream. Although the number of classes in the entire network traffic stream can be a few, or several, dozen, or even in the hundreds, it is certainly finite; therefore, we might as well set it to a large constant $C$. Thus, the spatial complexity of MicFoal is $O(3 \times z + C)$, which is finite and does not increase with the number of continuously arriving data instances.

## 4. Experimental evaluation

In this section, we describe the details of an experimental study we performed that verifies the usefulness of the proposed MicFoal framework. Section 4.1 presents eight real-world network traffic datasets and evaluation metrics. Section 4.2 presents parameter sensitivity experiments that test the performance of MicFoal under different parameter settings to obtain the appropriate default parameter values. Section 4.3 shows the contribution of each component of the MicFoal framework. Section 4.4 compares the performance of MicFoal with that of three state-of-the-art methods, and Section 4.5 presents a statistical analysis of all algorithms.

We implemented all learning algorithms except the RAL[2] algorithm (Wassermann et al., 2020) in the Massive Online Analysis (MOA) data stream software suite (Bifet, 2010). The RAL algorithm was implemented in the Python environment by using the Scikit-learn library. The experiments were executed on a computer equipped with a 16-core Intel i7-10700F processor operating at 2.9 GHz, and 48 GB of RAM. All experiments were repeated ten times with the random seeds set from 1 to 10 for all active learning label request strategies, and the results shown are the averages over ten trials. Due to space limitations, the complete detailed results of each test that include the results of the metrics for each class and that of the integral metrics for all classes and the mean and standard deviation of ten experimental results can be found on the website[3].

### 4.1. Experimental setting

#### 4.1.1. Experimental datasets

In our experiments, we use several well-known real-world network traffic datasets that are widely employed in network traffic classification research, such as the Moore (Moore, 2005), NSL-KDD (Tavallaee et al., 2009), BRASIL (Li et al., 2009), NIMS (Alshammari and Zincir-Heywood, 2011) and ISCX-URL (Mamun et al., 2016) datasets. Except for ISCX-URL, all datasets are multiclass imbalance. Details of these experimental datasets are described in Table 2.

The Entry10 dataset originates from the Moore[4] network traffic dataset that was released by Cambridge University and collected from the network traffic of 1,000 users over 24 h through a link. As the original traffic volume is too large, the Moore dataset is divided into

[2] https://github.com/SAWassermann/RAL

[3] https://github.com/sunnyweike/MicFoal

[4] https://www.cl.cam.ac.uk/research/srg/netos/projects/archive/nprobe/data/papers/sigmetrics/index.html

10 subsets (named entry01.weka.allclass to entry10.weka.allclass) according to a random sampling method, and the sampling time of each subset is almost the same (approximately 1,680 s). We select the subset entry10.weka.allclass as an experimental dataset. The original dataset entry10.weka.allclass has 65,036 instances and 11 traffic classes (namely, WWW, MAIL, FTP-CONTROL, FTP-PASV, ATTACK, P2P, DATABASE, FTP-DATA, SERVICES, INTERACTIVE and GAMES). As there are only two instances from class INTERACTIVE and 22 instances from class GAMES, which are too few compared to more than 60,000 instances in total, we remove the two classes INTERACTIVE and GAMES from entry10.weka.allclass to form an experimental dataset named Entry10.

The KDDTrain5 dataset originates from the NSL-KDD[5] dataset, which was released by the University of New Brunswick (UNB) in 2009 and was formed by resampling the well-known KDD'99 dataset to eliminate some inherent problems of the latter mentioned by (Tavallaee et al., 2009). Because the NSL-KDD dataset KDDTrain+ is subdivided into up to 23 network traffic classes and the proportion of instances of some minority classes is too low, we merge the 23 classes into five classes named Normal, DOS, Probing, R2L and U2R and then form an experimental dataset called KDDTrain5.

The UDP_NoPorts and TCP_NoPorts datasets are both derived from the BRASIL[6] datasets released by Cambridge University in 2008 and were collected from two different sites, namely, Sites A and B. These two sites were both research-centric but conducted research in very different disciplines, and were in two different countries. Sites A and B had more than a thousand local users from a population of researchers, administrators and technical support staff. At Site A, datasets captured on three weekdays in 2003, 2004, and 2006 (denoted by Day1, Day2, and Day3, respectively) were obtained. At Site B, datasets recorded on a weekday in late 2007 (denoted by Site B) were obtained. As the BRASIL datasets were collected from the network traffic of different regions (Site A and Site B) and during different time spans (from 2003 to 2007), we intend to use these datasets to simulate the real-world application scenarios, such as a change in the number of network traffic classes, a change in the multiclass imbalance ratios and a concept drift of traffic classes. Therefore, we combine SiteA.Day1.TCP, SiteA.Day2.TCP, SiteA.Day3.TCP and SiteB.TCP into an experimental dataset called TCP_NoPorts without two port attributes. Because the traffic proportion of class ADMIN is very small ($< 0.0001\%$), we remove that class from the TCP_NoPorts dataset. Similarly, datasets SiteA.Day1.UDP, SiteA.Day3.UDP and SiteB.UDP are combined into an experimental dataset called UDP_NoPorts, without two port attributes, and because the proportion of instances from the GAME class is only 0.00047%, we remove that class from the UDP_NoPorts dataset.

The NIMS1 dataset is derived from the NIMS[7] dataset released by the Network Information Management and Security Group and is mainly used to test the classification performance of machine learning methods for encrypted network traffic data. There are 11 classes in NIMS1: localForwarding, remoteForwarding, scp, sftp, x11, shell, TELNET, FTP, HTTP, DNS and lime.

Datasets URL_All, URL_BestFirst and URL_Infogain are derived from the ISCX-URL[8] dataset released by UNB in 2016 and is mainly used to test the performance of classification of malicious URL attacks. The ISCX-URL dataset includes five classes: Phishing, Malware, Spam, Defacement and Benigns. We choose three datasets from ISCX-URL2016, namely, ISCXURL2016All, ISCXURL2016All_BestFirst and ISCXURL2016All_Infogain, as our multiclass balanced experimental datasets of URL_All, URL_BestFirst and URL_Infogain.

[5] https://www.unb.ca/cic/datasets/nsl.html
[6] https://www.cl.cam.ac.uk/research/srg/netos/projects/brasil/
[7] https://projects.cs.dal.ca/projectx/Download.html
[8] https://www.unb.ca/cic/datasets/url-2016.html

### 4.1.2. Evaluation metrics

The evaluation metrics for multiclass imbalanced network traffic can be divided into two categories: the metrics for each class and the integral metrics (Brzezinski et al., 2019). In our experiment, the metrics for each class include $recall_i$ and $precision_i$, which reflect the performance of the learning model on class $i$. We evaluate the overall performance of the learning algorithms on our experimental datasets by using the following four dedicated skew-insensitive integrated metrics (Branco et al., 2017): average $F_\beta$ ($AvF_\beta$); class balance accuracy ($CBA$), multiclass G-measure ($mGM$), and confusion entropy ($CEN$) (Wei et al., 2010). For these metrics, except for $CEN$, a higher value corresponds to a better classification performance of the learning model. The four skew-insensitive integrated metrics are determined as follows:

$$AvF_\beta = \frac{1}{M}\sum_{i=1}^{M}\frac{(1+\beta^2)\cdot precision_i \cdot recall_i}{\beta^2 \cdot precision_i + recall_i} \tag{4}$$

$$CBA = \frac{1}{M}\sum_{i=1}^{M}\frac{mat_{i,i}}{\max(\sum_{j=1}^{M} mat_{i,j}, \sum_{j=1}^{M} mat_{j,i})} \tag{5}$$

$$mGM = \sqrt[M]{\prod_{i=1}^{M} recall_i} \tag{6}$$

$$CEN = \sum_{j=1}^{M} p_j \cdot CEN_j \tag{7}$$

where $M$ is the number of classes, $\beta$ is the relative importance of $recall_i$ compared with $precision_i$, and $mat_{i,j}$ denotes the number of instances of true class $i$ that are predicted to belong to class $j$, $P_j = (\sum_{k=1}^{M}(mat_{j,k} + mat_{k,j}))/(2 \times \sum_{k,l=1}^{M} mat_{k,l})$,and $CEN_j = -\sum_{k=1,k\neq j}^{M}(P_{j,k}^{j}\log_{2(M-1)}(P_{j,k}^{j}) + P_{k,j}^{j}\log_{2(M-1)}(P_{k,j}^{j}))$, where

$$\begin{aligned} &p_{i,i}^{i} = 0 \\ &P_{i,j}^{i} = mat_{i,j} / \textstyle\sum_{k=1}^{M}(mat_{i,k} + mat_{k,i}), i \neq j, i, j = 1, \ldots, M \end{aligned} \tag{8}$$

## 4.2. Parameter sensitivity and default parameter values

The basic idea of the parameter sensitivity experiments is to set different values for the compared parameters while keeping the other parameters unchanged. To assess parameter sensitivity of the proposed MicFoal method, in this subsection, we report the results of comparative parameter experiments. The compared parameters include the size $z$ of the sliding label window and the labelling cost budget $b$. The parameter sensitivity experiments are carried out on the experimental dataset Entry10. The default parameter settings are $z = 500$ and $b = 0.2$. Table 3 shows the results of the parameter sensitivity experiments.

As shown in the upper part of Table 3, given that the value of $b$ is within {0.15, 0.2, 0.25, 0.3}, when parameter $z$ increases from 300 to 1000, although the values of the four evaluation metrics change, the range of change is very small. The changes in $AvF_\beta$ ($\beta = 1$), $CBA$ and $mGM$ are approximately one percentage point, the changes in $CEN$ are within 0.1 percentage points, and there is no upward or downward trend; therefore, parameter $z$ is not sensitive. The default value for the size of the label sliding window, $z$, is set to 500 for easy comparison with the CALMID algorithm used in the subsequent comparative experiments in which the respective default size is also 500.

As shown in the lower half of Table 3, if $z$ has a value in {300, 500, 700, 1000}, as the labelling cost budget $b$ increases from 0.15 to 0.3, performance will also improve. Thus, increasing the labelling cost budget can help improve performance. In our experiments, the default value of the labelling cost budget $b$ is set to 0.2, as the respective default value in the CALMID algorithm is also 0.2.

**Table 2**
Characteristics of real-world network traffic datasets.

| Dataset | #Instances | #Features | #Classes | *IR* | Class distribution |
|---|---|---|---|---|---|
| Entry10 | 65013 | 248 | 9 | 672.05 | number of instances: 54436/6592/81/257/446/624/1773/592/212<br>proportion(%): 83.73/10.14/0.12/0.40/0.69/0.96/2.73/0.91/0.33 |
| KDDTrain5 | 125973 | 41 | 5 | 1295.06 | number of instances: 67343/45927/995/11656/52<br>proportion(%): 53.46/36.46/0.79/9.25/0.04 |
| UDP_NoPorts | 844806 | 7 | 5 | 473.28 | number of instances: 337920/140959/61031/714/304182<br>proportion(%): 40.00/16.69/7.22/0.08/36.01 |
| TCP_NoPorts | 1008323 | 10 | 13 | 9107.14 | number of instances: 846964/59460/28850/6781/572/44985/14581/530/1136/3454/762/155/93<br>proportion(%): 84.00/5.90/2.86/0.67/0.06/4.46/1.45/0.05/0.11/0.34/0.08/0.02/0.01 |
| NIMS1 | 713851 | 22 | 11 | 516.60 | number of instances: 1251/1728/11904/38016/646271/2557/2422/2444/2412/2355/2491<br>proportion(%): 0.18/0.24/1.67/5.33/90.53/0.36/0.34/0.34/0.34/0.33/0.35 |
| URL_All | 36707 | 79 | 5 | 1.18 | number of instances: 7586/6712/6698/7930/7781<br>proportion(%): 20.67/18.29/18.25/21.60/21.20 |
| URL_BestFirst | 36694 | 8 | 5 | 1.18 | number of instances: 7575/6711/6698/7930/7780<br>proportion(%): 20.64/18.29/18.25/21.61/21.20 |
| URL_Infogain | 36417 | 12 | 5 | 1.18 | number of instances: 7311/6707/6693/7930/7776<br>proportion(%): 20.08/18.42/18.38/21.78/21.35 |

**Table 3**
Results of the parameter (*z*, *b*) sensitivity experiments based on Entry10.

| *b* | 0.15 | | | | 0.2 | | | | 0.25 | | | | 0.3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *z* | 300 | 500 | 700 | 1000 | 300 | 500 | 700 | 1000 | 300 | 500 | 700 | 1000 | 300 | 500 | 700 | 1000 |
| $AvF_{\beta=1}$(%) | 84.46 | 85.31 | 85.77 | 84.46 | 87.02 | 86.18 | 86.04 | 87.02 | 87.07 | 87.22 | 86.78 | 87.07 | 87.35 | 87.67 | 87.87 | 87.35 |
| *CBA(%)* | 77.42 | 78.38 | 78.83 | 77.42 | 80.66 | 79.49 | 79.48 | 80.66 | 80.63 | 80.94 | 80.28 | 80.63 | 81.04 | 81.37 | 81.75 | 81.04 |
| *mGM(%)* | 74.28 | 75.62 | 76.55 | 74.28 | 78.77 | 77.11 | 76.80 | 78.77 | 78.68 | 78.92 | 77.94 | 78.68 | 79.16 | 79.62 | 79.99 | 79.16 |
| *CEN(%)* | 2.69 | 2.70 | 2.66 | 2.69 | 2.51 | 2.58 | 2.54 | 2.51 | 2.45 | 2.35 | 2.43 | 2.45 | 2.41 | 2.34 | 2.30 | 2.41 |
| *z* | 300 | | | | 500 | | | | 700 | | | | 1000 | | | |
| *b* | 0.15 | 0.2 | 0.25 | 0.3 | 0.15 | 0.2 | 0.25 | 0.3 | 0.15 | 0.2 | 0.25 | 0.3 | 0.15 | 0.2 | 0.25 | 0.3 |
| $AvF_{\beta=1}$(%) | 84.46 | 87.02 | 87.07 | 87.35 | 85.31 | 86.18 | 87.22 | 87.67 | 85.77 | 86.04 | 86.78 | 87.87 | 84.46 | 87.02 | 87.07 | 87.35 |
| *CBA(%)* | 77.42 | 80.66 | 80.63 | 81.04 | 78.38 | 79.49 | 80.94 | 81.37 | 78.83 | 79.48 | 80.28 | 81.75 | 77.42 | 80.66 | 80.63 | 81.04 |
| *mGM(%)* | 74.28 | 78.77 | 78.68 | 79.16 | 75.62 | 77.11 | 78.92 | 79.62 | 76.55 | 76.8 | 77.94 | 79.99 | 74.28 | 78.77 | 78.68 | 79.16 |
| *CEN(%)* | 2.69 | 2.51 | 2.45 | 2.41 | 2.7 | 2.58 | 2.35 | 2.34 | 2.66 | 2.54 | 2.43 | 2.3 | 2.69 | 2.51 | 2.45 | 2.41 |

### 4.3. Experiment for component contribution analysis

The purpose of the experiments in this subsection is to analyse the contribution of the following components to our MicFoal method: the initial supervised learner, the hybrid label request strategy (H), the sample training weight formula (S) and the adaptive adjustment mechanism for the label cost budget based on periodic performance evaluation (P). We perform our experiments on the Entry10 dataset. The proposed framework is flexible as to the selection of the underlying supervised learner or label query strategies.

First, to verify the flexibility of MicFoal, we select state-of-the-art supervised learners, namely, ARF and leveraging bagging(LB) (Bifet et al., 2010), as two different initial supervised learners. To assess the performance of the proposed hybrid strategy (H), we choose three traditional label request strategies (Zliobaite et al., 2014) for comparison: the uncertainty strategy with a variable threshold (VU), the uncertainty strategy with a random and variable threshold (RV) and the uncertainty strategy with selective sampling (SS). We gradually assemble the abovementioned four components to form two groups (the ARF and LB groups) with 12 active learning algorithms to be considered in the experiments for component contribution analysis. The results of this analysis are shown in Table 4.

For ease of viewing, we highlight the optimal values in bold in Tables 4–8. As shown in Table 4, regardless of whether the initial supervised algorithm is ARF or LB, the active learning algorithm combined with the hybrid strategy, the sample training weight formula and the periodic performance evaluation mechanism performs best in the corresponding comparison group on the four skew-insensitive integrated metrics, which shows the flexibility of MicFoal. Moreover, it is clear that the performance is better for the ARF group than for the LB group under the same configurations of other components; therefore, the default initial supervised learner in MicFoal is set to ARF.

Comparing the six active learning algorithms of each group in detail, we obtain the following findings: (1) the proposed hybrid strategy (H) greatly outperforms the three traditional strategies (RV, SS and VU); (2) adding the sample training weight formula (S) allows the performance of the active learning algorithm with the hybrid strategy to be improved by several percentage points, and (3) the active learning MicFoal algorithm combined with the hybrid strategy (H), the sample training weight formula (S) and a periodic performance evaluation mechanism (P) achieves the best performance.

### 4.4. Comparative experiments

#### 4.4.1. Comparison algorithms

In this subsection, we compare MicFoal with other state-of-the-art methods on the eight real-world network traffic streams listed in Table 2. The comparison algorithms include a supervised learning algorithm and two active learning algorithms, all of which are online ensemble learning methods that can learn from multiclass data streams. The supervised learning algorithm is ARFre, which is the most recent algorithm intended to solve the joint problems of concept drift and a variable multiclass imbalance ratio. One of the comparison active learning algorithms is CALMID[9] (Liu et al., 2021), which is the most recent ensemble active learning method specifically designed to solve the coexisting problems of a variable multiclass imbalance ratio, concept drift and a limited label cost budget. Another comparison active learning algorithm is the most recent RAL method (Wassermann et al.,

[9] https://github.com/sunnyweike/CALMID

**Table 4**
Results (%) of comparison experiments with different configurations of MicFoal components.

| Label request strategy | Sample training weight | Periodic performance evaluation | Initial supervised learner | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | ARF | | | | LB | | | |
| | | | *AvF*$_\beta$ ($\beta$ = 1) | *CBA* | *mGM* | *CEN* | *AvF*$_\beta$ ($\beta$ = 1) | *CBA* | *mGM* | *CEN* |
| RV | | | 64.55 | 55.08 | 42.89 | 5.85 | 57.25 | 44.47 | 12.12 | 8.00 |
| SS | | | 68.29 | 58.07 | 44.34 | 5.62 | 65.18 | 55.71 | 37.67 | 7.69 |
| VU | | | | 49.65 | 0.00 | 5.79 | 42.68 | 37.58 | 11.64 | 8.27 |
| H | | | 82.08 | 74.79 | 69.41 | 3.5 | 75.40 | 66.81 | 62.05 | 6.66 |
| H | S | | 85.08 | 78.47 | 74.94 | 2.65 | 78.64 | 71.39 | 64.68 | 4.74 |
| H | S | P | **86.18** | **79.49** | **77.11** | **2.58** | **80.88** | **74.06** | **69.71** | **4.37** |

**Table 5**
$AvF_\beta(\beta = 1)$ results (%) of the comparative experiments.

| Datasets | ARFre | CALMID | RAL | MicFoal |
|---|---|---|---|---|
| Entry10 | 83.44 | 65.54 | 78.69 | **86.18** |
| KDDTrain5 | 81.26 | 78.43 | 77.51 | **84.42** |
| UDP_NoPort | 96.98 | 94.93 | 92.31 | **97.68** |
| TCP_NoPorts | 81.14 | 73.38 | 74.77 | **89.73** |
| NIMS1 | 93.43 | 97.10 | 95.13 | **98.09** |
| URL_All | **99.83** | 98.02 | 74.29 | 99.79 |
| URL_BestFirst | **91.00** | 81.82 | 73.08 | 89.00 |
| URL_Infogain | 84.51 | 74.24 | 71.94 | **85.11** |
| Avg.rank | 2 | 3.125 | 3.625 | **1.25** |

**Table 6**
*CBA* results (%) of the comparative experiments.

| Datasets | ARFre | CALMID | RAL | MicFoal |
|---|---|---|---|---|
| Entry10 | 79.24 | 56.31 | 72.10 | **79.49** |
| KDDTrain5 | 77.86 | 74.69 | 74.72 | **80.51** |
| UDP_NoPort | 96.16 | 94.47 | 89.05 | **97.29** |
| TCP_NoPorts | 77.22 | 68.91 | 70.57 | **87.12** |
| NIMS1 | 90.10 | 95.22 | 92.65 | **97.37** |
| URL_All | **99.75** | 97.01 | 64.35 | 99.70 |
| URL_BestFirst | **89.82** | 77.15 | 71.03 | 87.87 |
| URL_Infogain | 80.14 | 70.08 | 69.65 | **81.11** |
| Avg.rank | 2 | 3.25 | 3.5 | **1.25** |

**Table 7**
*mGM* results (%) of the comparative experiments.

| Datasets | ARFre | CALMID | RAL | MicFoal |
|---|---|---|---|---|
| Entry10 | 76.69 | 47.90 | 67.44 | **77.11** |
| KDDTrain5 | 65.01 | 64.15 | 35.28 | **71.03** |
| UDP_NoPort | 96.16 | 94.47 | 89.05 | **97.29** |
| TCP_NoPorts | 77.22 | 68.91 | 70.57 | **87.12** |
| NIMS1 | 96.86 | 97.22 | 93.49 | **97.74** |
| URL_All | **99.83** | 97.95 | 76.13 | 99.79 |
| URL_BestFirst | **90.84** | 81.08 | 72.77 | 88.86 |
| URL_Infogain | 83.99 | 73.49 | 71.56 | **84.72** |
| Avg.rank | 1.875 | 3.125 | 3.75 | **1.25** |

**Table 8**
*CEN* results (%) of the comparative experiments.

| Datasets | ARFre | CALMID | RAL | MicFoal |
|---|---|---|---|---|
| Entry10 | 4.05 | 7.75 | 2.84 | **2.58** |
| KDDTrain5 | 2.22 | 4.17 | 2.74 | **1.64** |
| UDP_NoPort | 3.40 | 2.90 | 3.96 | **2.05** |
| TCP_NoPorts | 2.29 | 2.86 | 3.26 | **1.48** |
| NIMS1 | 5.40 | 0.89 | 1.25 | **0.58** |
| URL_All | **0.56** | 4.14 | 31.14 | 0.66 |
| URL_BestFirst | **18.39** | 29.83 | 42.14 | 21.43 |
| URL_Infogain | 27.32 | 39.97 | 43.33 | **26.71** |
| Avg.rank | 2.25 | 3 | 3.5 | **1.25** |

2020), which combines the reward mechanism with the model's uncertainty to tune the sample-informativeness heuristic to better guide the query decisions. In the comparative experiments, the parameters of the comparison algorithms were all set to their respective default values, and the label budget was set to 20% for all active learning algorithms (MicFoal, CALMID and RAL); An exception was the supervised learning algorithm ARFre with a label cost of 100%.

### 4.4.2. *Comparison of recall and precision for each class*

Fig. 3 shows the recall and precision results of a comparison of the proposed MicFoal method with other algorithms for each class of multiclass imbalance experimental datasets. The class name and the proportion of its instances are marked at the top of each axis of radar diagrams to make it easy to determines whether a class is a majority or a minority.

The recall radar diagram on the left of Fig. 3a shows that the recall values of the proposed MicFoal method for the minority classes FTP-CONTROL, FTP-PASV, ATTACK, FTP-DATA and SERVICES are significantly higher than the recall values of the compared active learning algorithms CALMID and RAL. For almost all classes, the recall values of MicFoal are similar to (or, for the minority class P2P, even higher than) the recall values of the supervised learning algorithm ARFre. On the right side of Fig. 3a, the precision radar diagram shows that the proposed MicFoal method outperforms the comparison algorithms for all classes.

Fig. 3b shows that the proposed MicFoal method outperforms the comparison algorithms in terms of the recall and precision metrics for almost all classes, except the U2R class. As the U2R class is a minority class that accounts for only 0.04%, and although the recall values of all algorithms for this class are not high, the proposed MicFoal method is still the best performer, as shown in Fig. 3b. On the right side of Fig. 3b, the precision metric's radar diagram shows that MicFoal performs much better than the compared active learning algorithms CALMID and RAL for the minority class U2R and performs similarly to the supervised learning algorithm ARFre.

In Fig. 3c–e, we observe similarity to Fig. 3a and b; namely, the prediction performance of the proposed MicFoal method for most minority classes is clearly superior to that of compared active learning algorithms CALMID and RAL, and its performance is similar to or even better than that of the supervised learning algorithm ARFre. Due to space restrictions, we do not separately discuss Fig. 3c–e.

Fig. 3f–h show the recall and precision results for a comparison of the proposed MicFoal method with other algorithms for each class of the following three balance datasets: URL_All, URL_BestFirst and URL_Infogain. A comparison of MicFoal with the supervised learning algorithm ARFre in Fig. 3f–h indicates a consistency with Fig. 3a–e. From Fig. 3g and h, we note that MicFoal outperforms CALMID and RAL. In Fig. 3f, the performance of ARFre is only slightly better than that of MicFoal, and accordingly the line representing the results for ARFre is hidden below that for MicFoal. Additionally MicFoal has a slight performance advantage over CALMID, but MicFoal is notably better than RAL.

### 4.4.3. *Comparison of integral metrics*

For all algorithms, we compare the values of integral metrics of $AvF_\beta(\beta = 1)$, *CBA*, *mGM* and *CEN*; the details are presented in Tables 5–8.

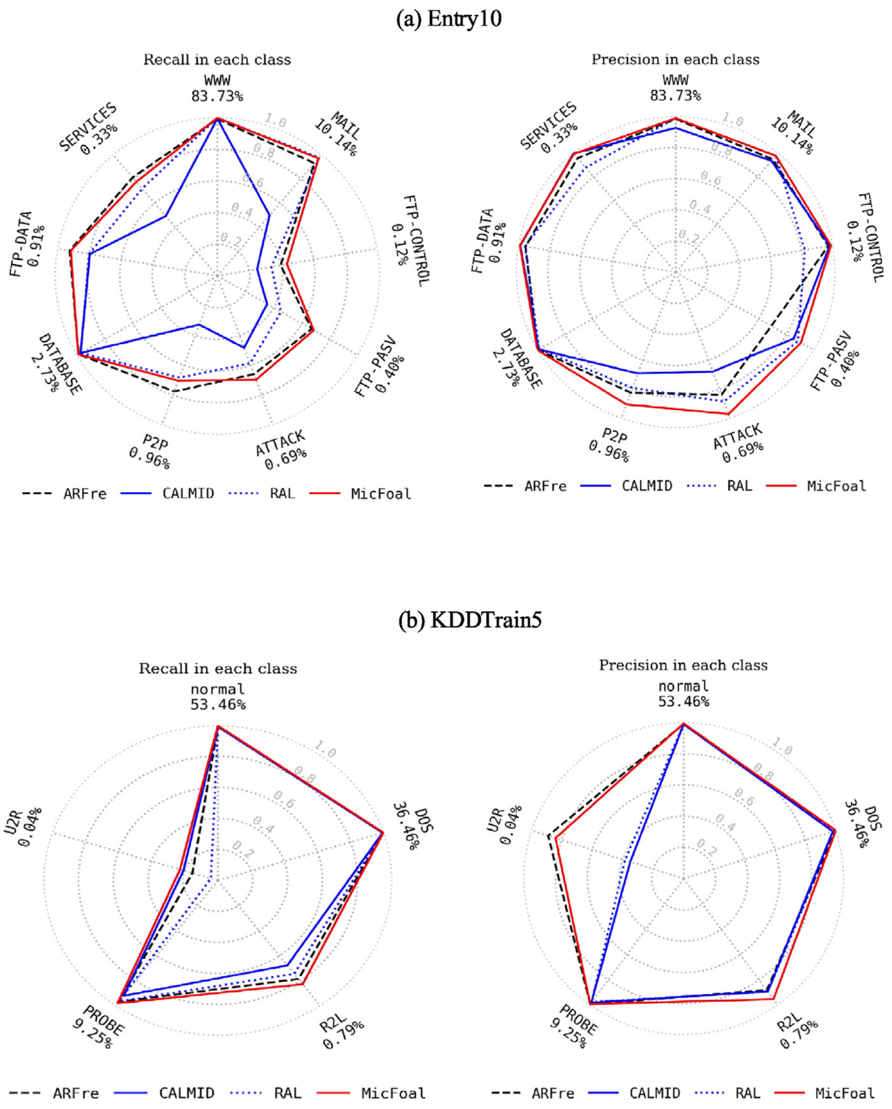

**Fig. 3.** Recall and precision results of each algorithm based on each class of experimental datasets.

As shown in Tables 5–8, the proposed MicFoal method achieves the best performance in terms of all integral metrics ($AvF_\beta(\beta = 1)$, *CBA*, *mGM* and *CEN*) on all imbalanced datasets and the balanced dataset URL_Infogain, while on the remaining two balanced datasets URL_All and URL_BestFirst, the performance of MicFoal is also similar to that of the supervised learning algorithm ARFre. Moreover, MicFoal is far superior to the compared active learning methods CALMID and RAF on all datasets.

Accordingly, MicFoal scores the first average ranking on all datasets, which indicates that MicFoal can not only distinguish minority class instances but also avoid misclassifying other instances.

### 4.5. Statistical analysis

This subsection presents the results of statistical tests calculated from the values of $AvF_\beta(\beta = 1)$, *CBA*, *mGM* and *CEN* of the different algorithms on all experimental datasets. The algorithms involved in statistical testing include the MicFoal method proposed in this paper and the three algorithms introduced in Section 4.4.1.

Algorithm rankings in Tables 5–8 show that MicFoal on average ranks the first in terms of each integral metric. To test whether there is a significant difference between our MicFoal algorithm and the comparison algorithms, we performed statistical tests; the results are listed in Table 9.

Table 9 shows the results of Iman-Davenport statistical tests. The critical values are at the level of 0.05, and can be found in the F-distribution table with 3 and 21 degrees of freedom. For the integral metrics $AvF_\beta$ ($\beta = 1$), *CBA*, *mGM* and *CEN*, the Iman-Davenport values are all larger than their critical values, which indicates that significant differences exist among the rankings of the algorithms. Next, we conduct Nemenyi tests. The critical difference (CD) calculated by the Nemenyi post-hoc test is 1.7624. The Nemenyi test results for $AvF_\beta$ ($\beta = 1$), *CBA*, *mGM* and *CEN* are plotted in Fig. 4.

As shown in Fig. 4, we can conclude that the performance of MicFoal does not differ significantly from that of the compared supervised algorithm ARFre, while MicFoal significantly outperforms the comparison active learning algorithms in terms of integral metrics $AvF_\beta$ ($\beta = 1$), *CBA* and *mGM*. However, Fig. 4d indicates that in terms of the *CEN* metric, MicFoal does not differ significantly from ARFre and CALMID. As shown in Table 8, the resulting values of the *CEN* metric on all imbalanced datasets are too small (less than 0.10), which limits discrimination among them. Therefore, although the *CEN* metric is a dedicated skew-insensitive integrated metric, it may not be suitable for evaluating performance on network traffic datasets with large class imbalances and even dynamic increases or decreases in the number of classes.

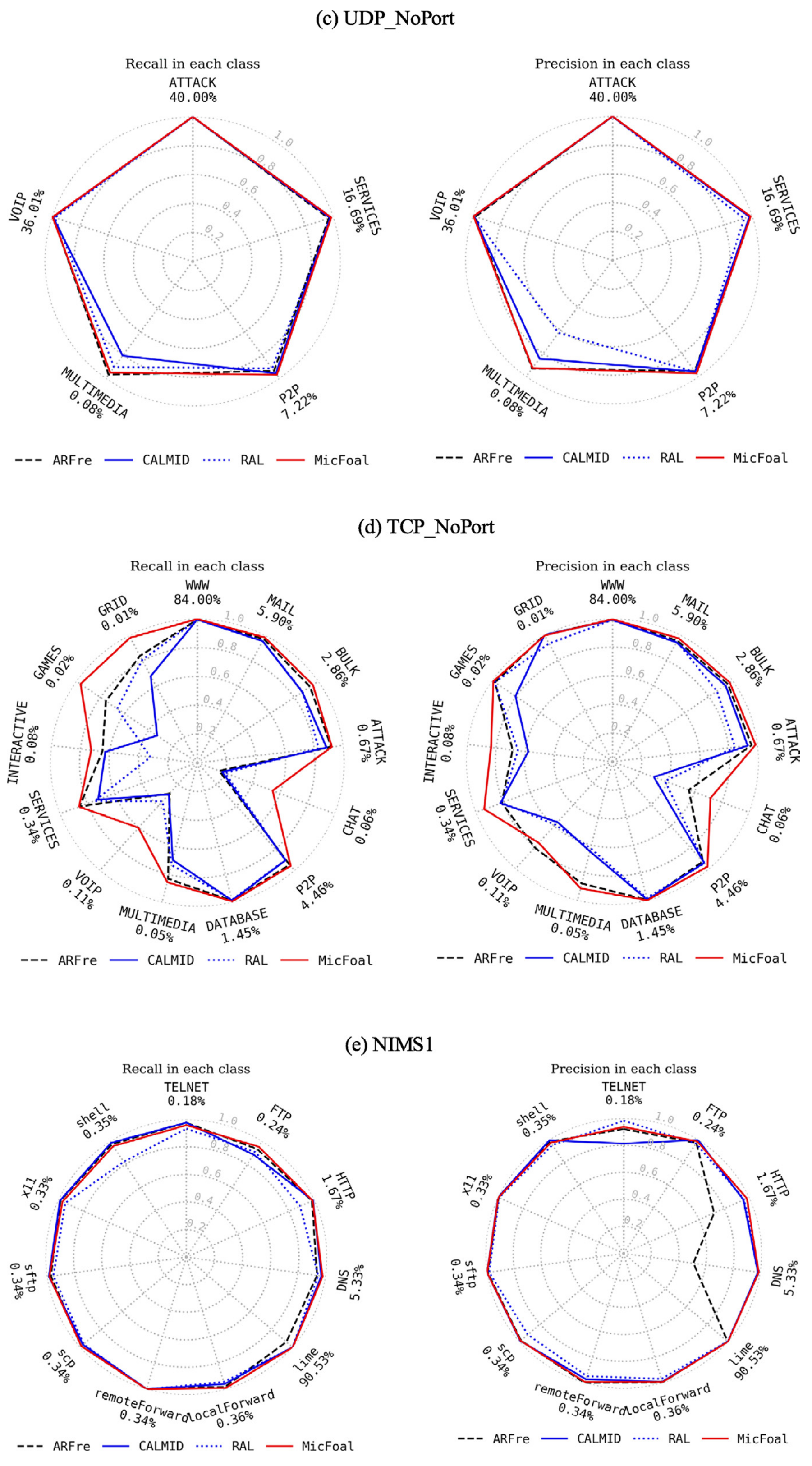

**Fig. 3.** (*continued*).

## 5. Conclusions

By focusing on the complex challenges of variable multiclass imbalance, concept drift, a limited label cost budget and one-by-one processing, we proposed a novel online network traffic classification framework based on active learning. The proposed uncertain label request strategy based on the variable least confidence threshold vector can tackle the mixed problems in which some old network traffic classes may no longer appear, while some new network traffic classes appear with new network applications, and a few traffic classes monopolize the

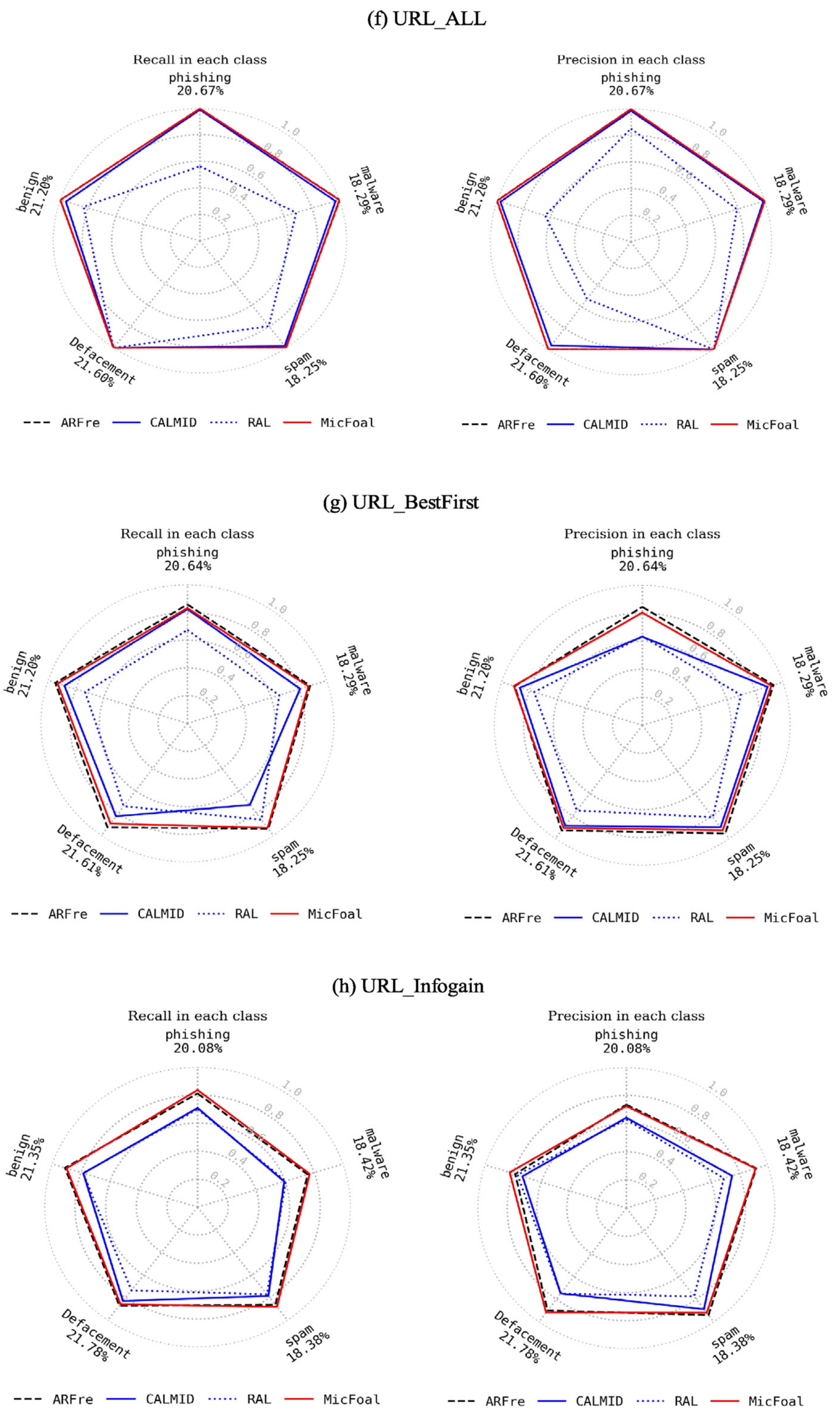

**Fig. 3.** (*continued*).

**Table 9**
Results of the Iman-Davenport tests.

| Indicators | Iman-Davenport values | Critical value ($\alpha$ = 0.05) | *p values* | Significant differences ? |
|---|---|---|---|---|
| $AvF_\beta(\beta = 1)$ | 15.857 | 3.072 | 1.29e−05 | Yes |
| *CBA* | 14.538 | 3.072 | 2.38e−05 | Yes |
| *mGM* | 21.718 | 3.072 | 1.22e−06 | Yes |
| *CEN* | 9.4706 | 3.072 | 3.71e−04 | Yes |

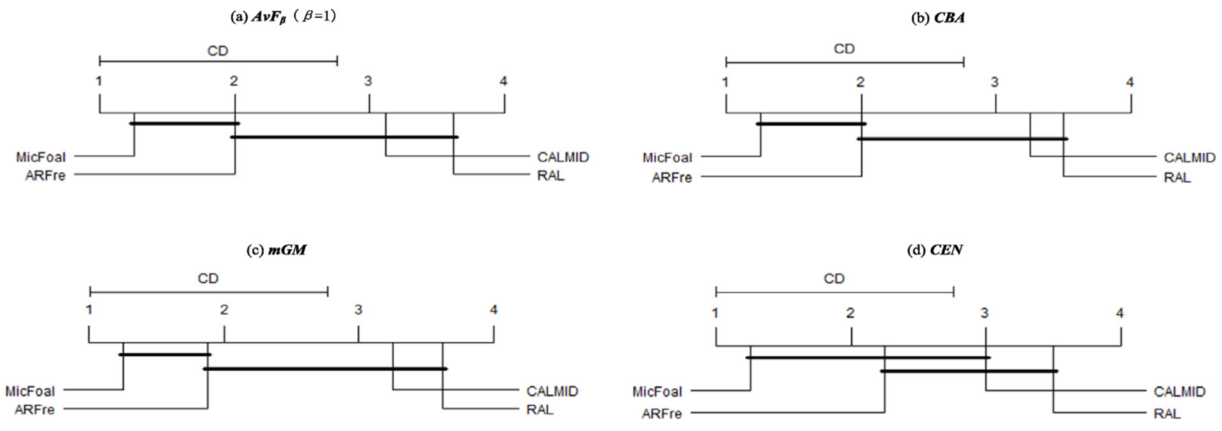

**Fig. 4.** Nemenyi test with a 95% confidence level on AvF$_\beta$($\beta$=1), CBA, mGM and CEN.

entire network traffic for a period of time. To address the problem of concept drift, we integrated active learning to blend the knowledge of human supervision with supervised learners that are based on initially limited labelled data but that have not designed a concept drift detector for multiclass imbalance cases.

In our future research, we plan to improve MicFoal in two ways. Inspired by Korycki and Krawczyk (2021), we intend to integrate the concept drift detector for multiclass imbalanced data streams proposed in Korycki and Krawczyk (2021) into our framework. Additionally, if concept evolution occurs in instances with high prediction confidence according to the classification model, it will be difficult for MicFoal to identify the concept evolution phenomenon in time. Thus, we intend to integrate an advanced detection method for concept evolution based on clustering in our framework.

## CRediT authorship contribution statement

**Weike Liu:** Conceptualization, Formal analysis, Methodology, Writing – original draft, Editing, Visualization. **Cheng Zhu:** Supervision, Project administration, Funding acquisition, Writing – review & editing. **Zhaoyun Ding:** Validation, Project administration, Writing – review & editing. **Hang Zhang:** Investigation, Software, Validation. **Qingbao Liu:** Conceptualization, Methodology, Validation, Writing – review & editing.

## Declaration of competing interest

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

## Data availability

Data will be made available on request.

## Acknowledgement

This work was supported by the National Natural Science Fund of China under grant no. 71971212.

## References

Alshammari, R., Zincir-Heywood, A.N., 2011. Can encrypted traffic be identified without port numbers, IP addresses and payload inspection? Comput. Netw. 55, 1326–1350.

Bifet, A., 2010. MOA : Massive Online Analysis Learning Examples.

Bifet, A., Holmes, G., Pfahringer, B., 2010. Leveraging bagging for evolving data streams. In: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part I.

Branco, P., Torgo, L., Ribeiro, R.P., 2017. Relevance-Based Evaluation Metrics for Multi-Class Imbalanced Domains.

Brzezinski, D., Stefanowski, J., Susmaga, R., Szczech, I., 2019. On the dynamics of classification measures for imbalanced and streaming data. In: IEEE Transactions on Neural Networks and Learning Systems, pp. 1–11.

Carela-Español, V., Barlet-Ros, P., Bifet, A., Fukuda, K., 2015. A streaming flow-based technique for traffic classification applied to 12 + 1 years of internet traffic. Telecommun. Syst. 63, 191–204.

De La Torre Parra, G., Rad, P., Choo, K.-K.R., 2019. Implementation of deep packet inspection in smart grids and industrial Internet of Things: Challenges and opportunities. J. Netw. Comput. Appl. 135, 32–46.

Deka, R.K., Bhattacharyya, D.K., Kalita, J.K., 2019. Active learning to detect DDoS attack using ranked features. Comput. Commun. 145, 203–222.

Ditzler, G., Polikar, R., 2013. Incremental Learning of Concept Drift from Streaming Imbalanced Data. IEEE Educational Activities Department.

Divakaran, D.M., Su, L., Liau, Y.S., Thing, V.L.L., 2015. SLIC: Self-Learning Intelligent Classifier for network traffic. Comput. Netw. 91, 283–297.

Dong, S., 2021. Multi class SVM algorithm with active learning for network traffic classification. Expert Syst. Appl. 176, 114885.

Elnawawy, M., Sagahyroon, A., Shanableh, T., 2020. FPGA-based network traffic classification using machine learning. IEEE Access 8, 175637-175650.

Erman, J., Mahanti, A., Arlitt, M., Cohen, I., Williamson, C., 2007. Offline/realtime traffic classification using semi-supervised learning. Perform. Eval. 64, 1194–1213.

Fahad, A., Almalawi, A., Tari, Z., Alharthi, K., Al Qahtani, F.S., Cheriet, M., 2019. SemTra: A semi-supervised approach to traffic flow labeling with minimal human effort. Pattern Recognit. 91, 1–12.

Ferreira, L.E.B., Gomes, H.M., Bifet, A., Oliveira, L.S., 2019. Adaptive random forests with resampling for imbalanced data streams. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–6.

García, S., Zhang, Z.-L., Altalhi, A., Alshomrani, S., Herrera, F., 2018. Dynamic ensemble selection for multi-class imbalanced datasets. Inform. Sci. 445–446, 22–37.

Gomes, H.M., Bifet, Read, Barddal, Enembreck, Pfharinger, Holmes, Abdessalem, 2017. Adaptive random forests for evolving data stream classification. Mach. Learn..

Gómez, S.E., Hernández-Callejo, L., Martínez, B.C., Sánchez-Esguevillas, A.J., 2019. Exploratory study on class imbalance and solutions for network traffic classification. Neurocomputing 343, 100–119.

Guo, Y., Li, Z., Li, Z., Xiong, G., Gou, G., 2020. FLAGB: Focal Loss based Adaptive Gradient Boosting for imbalanced traffic classification. In: 2020 International Joint Conference on Neural Networks (IJCNN).

Hoens, T.R., Chawla, N.V., 2012. Learning in non-stationary environments with class imbalance. In: Sigkdd Explorations.

Iliyasu, A.S., Deng, H., 2020. Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks. IEEE Access 8, 118–126.

Khanchi, S., Zincir-Heywood, N., Heywood, M., 2018. Streaming Botnet traffic analysis using bio-inspired active learning. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–6.

Korycki, L., Krawczyk, B., 2021. Concept drift detection from multi-class imbalanced data streams. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 1068–1079.

Koziarski, M., Woźniak, M., Krawczyk, B., 2020. Combined cleaning and resampling algorithm for multi-class imbalanced data with label noise. Knowl.-Based Syst. 204.

Li, W., Canini, M., Moore, A.W., Bolla, R., 2009. Efficient application identification and the temporal and spatial stability of classification schema. Comput. Netw. 53, 790–809.

Liu, Q., Liu, Z., 2012. A comparison of improving multi-class imbalance for internet traffic classification. Inf. Syst. Front. 16, 509–521.

Liu, S.M., Sun, Z.X., 2014. Active learning for P2P traffic identification. Peer-to-Peer Netw. Appl..

Liu, W., Zhang, H., Ding, Z., Liu, Q., Zhu, C., 2021. A comprehensive active learning method for multiclass imbalanced data streams with concept drift. Knowl.-Based Syst. 215.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G., 2020b. Learning under concept drift: A review. IEEE Trans. Knowl. Data Eng. 1.

Lu, N., Lu, J., Zhang, G., Lopez de Mantaras, R., 2016. A concept drift-tolerant case-base editing technique. Artificial Intelligence 230, 108–133.

Mamun, M.S.I., Rathore, M.A., Lashkari, A.H., Stakhanova, N., Ghorbani, A.A., 2016. Detecting malicious URLs using lexical analysis. Netw. Syst. Secur. 467–482.

Masud, M.M., Chen, Q., Khan, L., Aggarwal, C., Gao, J., Han, J., Thuraisingham, B., 2010. Addressing concept-evolution in concept-drifting data streams. In: 2010 IEEE International Conference on Data Mining, pp. 929–934.

Mirza, B., Lin, Z., 2016. Meta-cognitive online sequential extreme learning machine for imbalanced and concept-drifting data classification. Neural Netw. 80, 79–94.

Mirza, B., Lin, Z., Liu, N., 2015. Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. Neurocomputing 149, 316–329.

Moore, A., 2005. Discriminators for use in flow-based classification.

Odena, A., Olah, C., Shlens, J., 2016. Conditional Image Synthesis with Auxiliary Classifier GANs.

Pacheco, F., Exposito, E., Gineste, M., Baudoin, C., Aguilar, J., 2019. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. IEEE Commun. Surv. Tutor. 21, 1988–2014.

Peng, L., Zhang, H., Chen, Y., Yang, B., 2017. Imbalanced traffic identification using an imbalanced data gravitation-based classification model. Comput. Commun. 102, 177–189.

Shahraki, A., Abbasi, M., Taherkordi, A., Jurcut, A.D., 2021. Active Learning for Network Traffic Classification: A Technical Study.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6.

Torres, J.L.G., Catania, C.A., Veas, E., 2019. Active learning approach to label network traffic datasets. J. Inf. Secur. Appl. 49.

Trittenbach, H., Englhardt, A., Böhm, K., 2021. An overview and a benchmark of active learning for outlier detection with one-class classifiers. Expert Syst. Appl. 168.

Vu, L., Bui, C.T., Nguyen, Q.U., 2017. A deep learning based method for handling imbalanced problem in network traffic classification. In: Proceedings of the Eighth International Symposium on Information and Communication Technology, pp. 333–339.

Vu, L., Van Tra, D., Nguyen, Q.U., 2016. Learning from imbalanced data for encrypted traffic identification problem. In: Proceedings of the Seventh Symposium on Information and Communication Technology, pp. 147–152.

Wang, S., Leandro, L., Xin, Minku, Yao, 2017. A systematic study of online class imbalance learning with concept drift. IEEE Trans. Neural Netw. Learn. Syst..

Wang, P., Li, S., Ye, F., Wang, Z., Zhang, M., 2019. PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification using CGAN.

Wang, S., Minku, L.L., Xin, Y., 2013. A learning framework for online class imbalance learning. Comput. Intell. Ensemble Learn..

Wang, S., Minku, L.L., Yao, X., 2015. Resampling-based ensemble methods for online class imbalance learning. IEEE Trans. Knowl. Data Eng. 27, 1356–1368.

Wang, S., Minku, L.L., Yao, X., 2016. Dealing with multiple classes in online class imbalance learning. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence.

Wassermann, S., Cuvelier, T., Mulinka, P., Casas, P., 2020. Adaptive and reinforcement learning approaches for online network monitoring and analysis. IEEE Trans. Netw. Serv. Manag. 1.

Wei, H., Sun, B., Jing, M., 2014. BalancedBoost: A hybrid approach for real-time network traffic classification. In: Proceedings International Conference on Computer Communications & Networks ICCCN.

Wei, J.M., Yuan, X.J., Hu, Q.H., Wang, S.Q., 2010. A novel measure for evaluating classifiers. Expert Syst. Appl. Int. J. 37, 3799–3809.

Widmer, G., Kubat, M., 1996. Learning in the presence of concept drift and hidden contexts. Mach. Learn. 23, 69–101.

Yi-peng, W., 2013. Networkprotocolidentificationbasedonactive learningand SVM algorithm.

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

Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G., 2014. Active learning with drifting streaming data. IEEE Trans. Neural Netw. Learn. Syst. 25, 27–39.