Received 16 April 2024, accepted 29 April 2024, date of publication 6 May 2024, date of current version 13 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3397512

RESEARCH ARTICLE

UASDAC: An Unsupervised Adaptive Scalable DDoS Attack Classification in Large-Scale IoT Network Under Concept Drift

SARAVANAN SELVAM[1] **AND**
UMA MAHESWARI BALASUBRAMANIAN[2]**, (Senior Member, IEEE)**
[1]Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Chennai 601103, India
[2]Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Bengaluru 560035, India

Corresponding author: Saravanan Selvam (s_saravanan@ch.amrita.edu)

ABSTRACT Day by day, the number of devices in IoT networks is increasing, and concurrently, the size of botnets in IoT networks is also expanding. Currently, attackers prefer IoT-based botnets to launch DDoS attacks, as IoT devices offer a vast attack surface. Many researchers have proposed machine and deep learning-based classifiers to classify DDoS and benign network traffic in online streams from IoT devices. However, the performance of the traditional machine and deep learning algorithms deteriorates when sudden concept or data drift occurs in the online streams and the volume and velocity of IoT network traffic increases. To address these challenges, we propose UASDAC, an adaptive and scalable data pipeline designed specifically to handle concept drift and detect DDoS traffic in real-time in massive online streams originating from IoT devices. UASDAC incorporates three key components: an online network stream collector for data collection, an online network stream analyzer with an unsupervised drift detector for detecting drift and DDoS traffic, and an online network stream repository for storing streams for future analytics. UASDAC leverages big data technologies to implement all the three components to achieve scalability. Additionally, UASDAC introduces an effective and efficient retraining technique to adapt to novel patterns in online streams in the presence of concept drift. We evaluated the performance of UASDAC in different concept drift scenarios using the benchmark dataset NSL-KDD and the latest IoT dataset IoT23. Our results demonstrate that UASDAC effectively identifies DDoS traffic in the presence of concept drift, achieving an accuracy range of 99.7% to 99.9%.

INDEX TERMS Attacks, big data, botnet, concept drift, DDoS, IoT, machine learning, network, online streams.

I. INTRODUCTION

IoT-based botnet is a collection of infected IoT devices that can be used to launch large-scale DDoS attacks [1], [2], [3]. The size of IoT-based botnets keeps increasing every day. Consequently, the volume of botnet traffic is also increasing. For example, Nokia's threat intelligence report 2023 mentions that the number of IoT devices (bots) engaged in botnet-driven DDoS attacks rose from around 200,000 a year ago to approximately 1 million devices [4]. The main reason for using IoT devices to form a botnet is that IoT devices are generally insecure and can be compromised easily [5]. There are a couple of challenges in detecting DDoS traffic in the massive streaming network traffic of IoT devices. The first challenge is the limitations of IoT devices in terms of memory and processing capacity [6]. Due to these limitations, installing DDoS traffic detection tools locally in IoT devices is not feasible. Hence, the service providers of large-scale IoT networks need to move the network traffic from IoT devices to the cloud for analytics [7]. It demands a scalable data pipeline to collect, analyze, and store the ever-growing network traffic streams from geographically distributed IoT devices [8].

The second challenge is that the supervised machine learning algorithms that detect DDoS network traffic deteriorate performance when concept drift occurs in the online

The associate editor coordinating the review of this manuscript and approving it for publication was Renato Ferrero.

network streams [9]. Concept drift refers to the unpredictable changes in the statistical distribution of training and test instances [10], [11]. Generally, the network traffic originating from the IoT network is ever-changing as the IoT environment is non-stationary and dynamic [12], [13]. Due to this, the supervised machine learning algorithm trained on training instances may result in poor performance when applied to incoming online streams with concept drift. For example, in IoT based patient respiratory monitoring system, the IoT device can be considered a DDoS attack launcher if the number of messages originating from that device is above 20 per second. However, during the COVID-19 pandemic, the benign IoT device that monitors the patient's respiration might generate more than 20 messages per second if the patient is infected with COVID-19. Then, in this scenario, this benign device will be classified as a DDoS device. As a result, the model's False Positive Rate (FPR) increases [14], [15].

Many cybersecurity researchers have identified the importance of concept drift detection in IoT networks and proposed several techniques to find the drift in online network streams [16], [17]. Existing drift detection techniques can be grouped into supervised and unsupervised [18], [19], [20]. Supervised drift detection techniques require true labels of streaming test instances to detect the drift. Meanwhile, unsupervised techniques can detect drifts without knowing the true labels of incoming test instances. In real-life scenarios, class labels of incoming test instances are not known. Hence, supervised techniques are not suitable for detecting drifts in real-life scenarios.

When concept drift occurs, the DDoS traffic classification system must adapt to the novel patterns in the incoming online network streams. Retraining is one of the methods to adapt the DDoS attack classifier to the novel patterns. In the retraining method, existing techniques typically retrain the classifier using either the entire set of streaming test instances that caused the drift or a combination of these instances and the entire old training dataset. When retraining uses only the entire set of streaming test instances causing the drift, the retrained classifier may not recognize the old instances, i.e., old concepts. As a remedy to this, if we retrain with a combination of the entire set of streaming test instances causing the drift and old concepts, the retraining time will increase.

To address the first challenge, many researchers have proposed scalable data pipelines [21], [22], [23]. However, these solutions primarily focus on handling massive streaming IoT traffic and often neglect to consider concept drift. Hence, we propose a scalable and adaptive data pipeline that offers both scalability and adaptability to detect DDoS traffic in the massive streaming IoT traffic in the presence of concept drift. Our proposed data pipeline consists of three key components: an online network stream collector for data collection from IoT devices, an online network stream analyzer with an unsupervised drift detection and a novel adaptation technique, and an online network stream repository for storing streams for future analytics. The proposed data pipeline leverages big data technologies such as Apache Kafka to collect online network streams originating from geographically distributed IoT devices, Apache Structured Streaming to process the massive ingested online IoT network streams, and MongoDB to store the online network streams for future analytics.

To address the second challenge, many researchers have adopted unsupervised and retraining techniques for concept drift detection and adaptation respectively. We adopt a simple yet efficient unsupervised drift detection technique based on the Kolmogorov-Smirnov (KS) test and apply it on prediction probabilities of training instances and incoming test instances and calculate the p-value. If the p-value is less than 0.05, we conclude that there is a drift in the incoming test instances and trigger the drift adaptation component to retrain the classifier. We choose KS test for its simplicity, ease of implementation, non-parametric nature, and suitability for continuous data. After detecting the drift, we employ a novel, effective and efficient retraining technique to make the classifier adaptable to concept drift. Our novel retraining mechanism prepares a retraining dataset using a calculation method that selects an optimal number of instances for every class label from the set of streaming test instances that caused the drift and the old training dataset. This calculation method ensures that the retraining dataset encompasses sufficient examples for each class from the new and old concepts, allowing the classifier to adapt effectively to the evolving data distribution. Further, it reduces the retraining time as it retrains with only the optimal number of instances. This research presents the following significant contributions.

- Developing an unsupervised concept drift detection method based on Kolmogorov-Smirnov test.
- Proposing an efficient and effective retraining mechanism that uses a simple calculation to prepare the retraining dataset for retraining when concept drift occurs.
- Designing and conducting experiments to study the performance of the proposed system in different concept drift scenarios.
- Evaluating the proposed system on the benchmark dataset NSL-KDD [24] and the latest publicly available IoT23 dataset [25].
- Comparing the proposed system with a state-of-the-art existing system, Optimized Adaptive and Sliding Windowing (OASW) [26].

The remainder of this article is organized as follows. Section II discusses the related works. Section III introduces the proposed system in detail. Section IV elaborates the experiments conducted and discusses the evaluations in detail. Section V concludes the article.

II. RELATED WORK

In the preceding section, we identified two key challenges in detecting DDoS traffic from the streaming network traffic of large-scale IoT network in the presence of concept drift. In this section, we present the existing research works in these key challenges.

A. HANDLING OF VAST AMOUNTS OF STREAMING IOT TRAFFIC

In this subsection, we review the latest research articles that mainly focus on using data pipelines to detect DDoS traffic in massive streaming IoT traffic.

Zhou et al. [27] developed a system that consists of three parts namely, collector, messaging system and stream processor. The three parts are implemented with Jpcap, Apache Kafka and Spark Streaming respectively. They compared the performance of three machine learning algorithms in detecting three DDoS attacks – TCP, UDP and ICMP flooding.

Patil et al. [28] presented a novel Spark Streaming and Kafka-based distributed classification system, named by SSK-DDoS, for classifying different types of DDoS attacks and legitimate network flows in real-time. They evaluated SSK-DDoS on CICDDoS2019 dataset and obtained the classification accuracy of 89.05%.

Shih et al. [29] built a Cloudera based big data platform with tools such as Kafka, Spark Streaming, HBase, Hive, and Impala to classify DDoS attacks. They used deep neural network as their classifier and achieved accuracy of 94%.

Shi et al. [30] used Long Short-Term Memory to detect abnormal traffic in CICIDS2017 dataset. They utilized big data processing tools like Apache Kafka, Spark Streaming for online network traffic collection and processing. The detection accuracy reached upto 99%.

Abid et al. [31] introduced a real-time, distributed, fault-tolerant and scalable system for detecting DDoS attacks in Edge-IIoT dataset by leveraging data fusion and big data technologies such as Kafka and Spark Streaming. They achieved the high accuracy of 99.97% and 99.98% in binary and multiclass classification respectively when multilayer perceptron is used.

Alghamdi and Bellaiche [32] introduced a scalable intrusion detection system utilizing multi-staged binary and multi-class classifiers, employing simple and ensemble-based deep learning techniques. They leveraged the Lambda architecture to enhance efficiency by training classifiers at the batch layer and analyzing real-time IoT traffic in the low-latency speed layer. Results show that the ensemble approach gives the high accuracy of over 99.93% for IoT23 dataset.

Yahyaoui et al. [33] designed a data pipeline that collects data from IoT sensors using Kafka and feeds it to Spark Streaming and Apache Flink for processing to detect the intrusions. They used Apache Cassandra to store the results. They compared the performance of Spark Streaming and Apache Flink on KDDCUP99 and N-BaIoT datasets. Their results show that Apache Flink achieved considerable throughput and high detection accuracy.

The modern IoT traffic is ever-growing and ever-changing. The above existing methods proposed scalable data pipeline to handle ever-growing characteristic of streaming IoT traffic. However, they do not address the ever-changing characteristic of the streaming IoT traffic by implementing concept drift detection and adaptation.

B. DETECTION OF CONCEPT DRIFT IN STREAMING IoT TRAFFIC

In this subsection, we review the latest research articles that primarily focus on concept drift detection and adaptation in the domain of security attack classification in streaming IoT traffic. In our review, we highlight the working of concept drift detection and adaptation process, whether old instances are considered or not for retraining and whether the experimentations are conducted for detecting drift in different scenarios.

L. Yang and A. Shami [26] presented an Optimized Adaptive and Sliding Window (OASW) algorithm to counter concept drifts in IoT streaming network traffic. OASW maintains two windows. The size of the windows is determined based on experiments. When a test instance arrives, both windows slide, and the accuracy of the current window is compared with the accuracy of the previous window. If the accuracy drops below a certain threshold, they conclude drift exists and retrain the classifier. This method works only when the class label of the incoming test instances is known immediately. This work did not consider old concepts for retraining and did not test concept drift detection method in different concept drift scenarios. In the development of the classifier, the authors of [26] chose to employ LightGBM, an ensemble algorithm. LightGBM offers superior generalizability and robustness compared to many other ML algorithms, particularly when dealing with non-linear and high-dimensional data.

Jain and Kaur [34] proposed a drift detection system that declares the drift in the incoming test instances if the prediction accuracy of the model is less than 90% and the False Positive Rate (FPR) is above 10%. When the drift is detected, they retrain the model by replacing the old instances with new instances causing the drift. They employed stacking ensemble technique for classifying attacks. Their ensemble technique involves two levels: Level 0 and Level 1. Level 0 consists of two base learners such as Random Forest and Logistic Regression. The output of Level 0 base learners is combined and given as an input for Support Vector Machine, a Level 1 classifier. The authors of [34] did not design different concept drift scenarios and study the performance of their proposed drift detection mechanism.

Shao et al. [35] utilized Adaptive Windowing (ADWIN) to detect drift in IoT botnet detection problems. Two different configurations are used to explore the effectiveness of adaptive learning strategy in IoT botnet detection: Hoeffding Adaptive Tree and Adaptive Random Forest. They achieved the high accuracy of 99.95% when Adaptive Random Forest is used. They studied the performance of their proposed system for data streams with unbalanced data.

Yang et al. [36] utilized Drift Detection Method (DDM) and ADWIN to detect drift. They proposed the Performance-Weighted Probability Averaging Ensemble technique to adapt to the drifts. The prediction probability of 4 base learners, ARF-ADWIN, ARF-DDM, SRP-ADWIN, and SRP-DDM, are multiplied with those base learners' weights for attack and benign class labels separately. The sum of (prediction probability*weights) is calculated separately for attack and benign class. If the sum of the attack class label is above the benign class label, then the instance is classified into attack. The authors did not design different concept drift scenarios for evaluating the proposed system. However, their proposed system considers old concepts while retraining.

Schwengber et al. [37] proposed a system that assigns class label '0' to the training instances and class label '1' to the test instances. Then, the decision tree is used as a classifier. If the decision tree correctly classifies old and new data as two distinct classes, then it is declared that drift occurred. After drift detection, hierarchical agglomerative clustering distinguishes botnet flow from benign flow. The authors did not evaluate the proposed system in different concept drift scenarios and mention about drift adaptation.

Yang and Shami [38] detected drift using ADWIN and Early Drift Detection Method (EDDM). 6 base learners are used for retraining when drift occurs. 2 of them are leader models. 2 other models are selected from the remaining four models based on performance. Then, the Window-based Performance Weighted Probability Averaging Ensemble (W-PWPAE) model is used for final prediction. Like [36], the authors did not design different concept drift scenarios for evaluating the proposed system. However, their proposed system considers old concepts while retraining.

Qiao et al. [39] used the residual subspace projection method to calculate the anomalous quantity in the incoming instances. When the anomalous quantity increases, the concept drift occurs. If the concept drift is detected, LSTM and CNN-based classifiers are retrained to classify multiclass attacks. However, they did not mention whether they use old instances along with new instances for retraining. They studied the performance of their proposed system for data streams with unbalanced data.

Yang et al. [40] designed a technique that estimates the distance of an incoming sample to all the existing classes in the training set. If any incoming sample exhibits a significant distance from all the existing classes, then that sample is chosen as the candidate drifting sample. Then, the autoencoder improves its learning with the candidate drifting sample. The authors studied the performance of their drift detection technique in the scenario where drifting samples come from a new class that does not exist in the training dataset.

Andresini et al. [41] employed DNN to predict the class label of the incoming instances. If the DNN model assigns the least certain prediction to an incoming instance, then the incoming instance is assigned a pseudo-label and will be given as input to the DNN model for updating. They studied the performance of their proposed system for data streams with unbalanced data.

Korycki and Krawczyk [42] developed a trainable drift detector that is based on Restricted Boltzmann machine. The drift detector is fully trainable and is capable of autonomous adaptation to the current state of a stream, imbalance ratio, and class roles without relying on user-defined thresholds. They assigned larger training weights to minority class and identical instances using a novel training weight formula. They studied the performance of their proposed system in imbalanced data streams.

Liu et al. [43] employed a technique that selects the test instances with low prediction confidence for retraining. Adaptive Random Forest (ARF) and Leverage Bagging are two algorithms used for training the classifier in multiclass imbalance network traffic.

Abdel Wahab [14] utilized PCA to detect the drift in the intrusion data. An online DNN is proposed that dynamically adjusts the sizes of hidden layers based on the Hedge weighing mechanism to learn and adapt as new intrusion data steadily emerges. The authors did not design different concept drift scenarios for evaluating the proposed system.

Xu et al. [44] proposed autoencoder-based incremental learning for anomaly detection in the presence of concept drift in IoT networks. They implemented a dynamically adjusting parameters algorithm to judge the importance of data in real-time and incrementally update the model according to the importance. The proposed system is not tested in different concept drift scenarios.

Jain et al. [45] presented an error-based and data distribution-based methods to detect drift. They measured the severity of concept drift to decide the amount of data needed to retrain the model. They detected anomalies in the streaming network intrusion data using Support Vector Machine classifier. They also employed K-means clustering for sample size reduction of training datasets. They retrained the model by merging testing and training data whenever drift is occurred.

Amin et al. [46] evaluated the performance of drift detection algorithms such as DDM, EDDM, STEPD and ACE by combining them with four classifier algorithms such as 1-Nearest-Neighbor, Naive Bayes, SVM and LSTM: DDM-1NN, DDM-NB, DDM-SVM, DDM-LSTM, EDDM-1NN, EDDM-NB, EDDM-SVM, EDDM-LSTM, STEPD-1NN, STEPD-NB, STEPD-SVM, STEPD-LSTM, ACE-1NN, ACE-NB, ACE-SVM, and ACE-LSTM. They retrain the model only with new data.

Wang et al. [47] proposed DDM-FP-M, a drift detection method, that utilizes error rate and false positive rate to detect concept drift in multi-label class dataset. The proposed DDM-FP-M is tested on the testbed dataset and compared with DDM that uses prequential error rate. The result demonstrated that DDM-FP-M detected the concept drift earlier than DDM. They focused only on detecting the drift in the streaming dataset.

While exploring these research works, we observed the use of different techniques for drift detection and adaptation in online IoT network streams. However, most of these research works have not studied the performance of the proposed techniques in different concept drift scenarios. Further, despite the increasing prevalence of IoT devices generating substantial amounts of streaming network traffic, only limited research articles offered a comprehensive data pipeline for efficiently collecting vast streaming network traffic from IoT devices, employing distributed processing for real-time analysis, and subsequently storing the processed data in a distributed database for future analytics. However, those research articles did not address the handling of concept drift in their data pipeline. Hence, in this work, we present a novel data pipeline that consists of three components, namely, an online network stream collector to collect streaming network traffic from IoT devices, an online network stream analyzer to detect concept drift and DDoS attacks in online network streams, and online network stream repository to store network traffic for future analytics. Our data pipeline also offers an efficient and effective retraining mechanism that uses a simple calculation to prepare the retraining dataset for retraining when concept drift occurs. These components are implemented with big data technologies to offer scalability. Further, we also study the performance of the proposed drift detection component in different concept drift scenarios. Table 1 summarizes the reviewed research articles.

III. PROPOSED SYSTEM

This section presents the details of the proposed data pipeline, concept drift detection, and retraining methods. There are three components in the proposed system, namely, the online network stream collector, online network stream analyzer, and online network stream repository. The workflow of the proposed data pipeline is illustrated in Figure 1.

A. ONLINE NETWORK STREAM COLLECTOR

The main task of Online network stream collector is to collect the streaming network traffic from geographically distributed IoT devices and ingest it into an online network stream analyzer component. This component utilizes tools, namely NFStream [48] and Apache Kafka [49], to complete its task. The roles and responsibilities of these tools are given below.

1) NFSTREAM

It is a Python-based open-source package that can convert packets to network flows. Our online network stream analyzer operates on network flows rather than packets. Hence, NFStream is used to convert packets from IoT devices to network flows. NFStream groups packets based on 5-tuple <Source IP, Destination IP, Source Port, Destination Port, Protocol> and generates flows. It allows us to extract flow-based statistical features such as the number of packets sent from source to destination and vice versa, the number of bytes sent from source to destination, and vice versa. Algorithm 1 explains the process of extracting statistical network flow features from a packet capture file.

2) APACHE KAFKA

Apache Kafka is an open-source distributed data ingestion tool widely used in high-performance stream processing

Algorithm 1 Obtaining Flow-Based Statistical Features From Packet Capture (PCAP) File

```
Input: PCAP file from an IoT device, the required flow-based statistical features
Output: Flow-based statistical features for each flow
1. Open and read PCAP file
2. Group packets based on 5-tuple <Source IP,
   Destination IP, Source Port, Destination Port,
   Protocol> and generate flows
3. for each flow in flows
       Extract flow-based statistical features and store
       them in a CSV file
4. Close PCAP file
```

data pipelines. It can ingest streaming data from the distributed source to the stream processing engine for analytics. We choose Apache Kafka for its high-throughout, fault-tolerant, distributed, and scalable nature, making it well-suited for handling the ever-growing streaming traffic from IoT devices [50]. In our proposed system, Apache Kafka ingests the flow-based statistical features of each flow from NFStream to the online network stream analyzer component in regular time intervals.

B. ONLINE NETWORK STREAM ANALYZER

The primary function of this component is to receive the statistical features of a set of flows from Apache Kafka and classify each flow into either DDoS or Benign at regular intervals. This component can identify and retrain the classifier if there is a drift in the online streams. This component comprises three subcomponents: drift detector, retrain, and classifier. This component is implemented with Apache Spark [51] and Apache Spark Structured Streaming API [52]. We choose Apache Spark for its rapid data processing and analysis capability. This is essential for detecting and adapting to concept drift in high velocity streaming IoT data. Additionally, Spark's support for various machine learning algorithms facilitates the implementation of drift detection and classification models in a distributed environment. The drift detector and classifier subcomponents work together. Hence, we explain them first and then present the working of retrain subcomponent.

1) DRIFT DETECTOR AND CLASSIFIER

In online streaming mode, the drift detector and classifier subcomponents receive the flow-based statistical features of a set of online streams in regular intervals. The drift detector finds the existence of drift in the streaming flows. Meanwhile, the classifier subcomponent classifies the flow into either DDoS or benign. Initially, we train the machine learning algorithm with initial training instances, Train_set$_{initial}$, in offline mode and generate an initial model, Initial$_{model}$. This Initial$_{model}$ is used in both drift detector and classifier subcomponents. However, we refer to the Initial$_{model}$ used in the drift detector and classifier subcomponents as DD$_{model}$ and Classifier$_{model}$,

TABLE 1. Summary of drift detection techniques and datasets in reviewed research articles.

Authors	Are scalability and data pipeline addressed?	Is concept drift detection technique employed?	Dataset used
B. Zhou *et al.* [27], 2018	Yes	No	Collected normal packets from campus network and generated DDoS packets using bonesi, a DDoS botnet simulator
N. V. Patil *et al.* [28], 2022	Yes	No	CICDDoS2019
W.-C. Shih *et al.* [29], 2022	Yes	No	KDDCUP99
Z. Shi *et al.* [30], 2019	Yes	No	CICIDS2017
A. Abid *et al.* [31], 2023	Yes	No	Edge-IIoT
R. Alghamdi and M. Bellaïche [32], 2023	Yes	No	IoT23
A. Yahyaoui *et al.* [33], 2021	Yes	No	KDDCUP99, N-BaIoT
L. Yang and A. Shami [26] 2021	No	Accuracy-based	IoTID20, NSL-KDD
M. Jain and Gaur [34], 2021	Only scalability	Accuracy-based	NSL-KDD, CIDDS-2017
Zhou *et al.* [35], 2021	No	ADWIN	Collected real IoT botnet data from lab using MIRAI botnet
L. Yang *et al.* [36], 2021	No	ADWIN and DDM	CICIDS2017, IoTID20
B. H. Schwengber *et al.* [37], 2022	No	Unsupervised	Botnet 2014, CTU-13
L. Yang and A. Shami [38], 2023	No	ADWIN and EDDM	CICIDS2017, IoTID20
H. Qiao *et al.* [39], 2022	No	Residual subspace projection	BOT-IoT, SEA concept, UG-2C-5D
L. Yang et al. [40], 2021	No	Distance-based	Drebin Android malware, IDS2018
G. Andresini *et al.* [41], 2021	No	There is no specific technique to detect drift. However, the instances that yield the least certain predictions are assigned pseudo labels and used in retraining.	CICIDS2017
L. Korycki *et al.* [42], 2021	No	Reconstruction error based on Restricted Boltzmann Machine and Skew-insensitive loss function	24 benchmark data streams: 12 come from real-world domains and 12 were generated artificially using the MOA environment
W. Liu *et al.* [43], 2023	No	There is no specific technique to detect drift. However, the instance whose prediction confidence is less than the threshold is selected for retraining	Moore, NSL-KDD, BRASIL, NIMS, ISCX-URL
O. Abdel Wahab [14], 2022	No	PCA	DS2OS traffic traces
L. Xu *et al.* [44], 2023	No	Statistical method based on Hoeffding's inequality	INSECTS, SwaT, WADI, BATADAL
M. Jain *et al.* [45], 2022	No	Error-based and data distribution-based.	NSL-KDD, CIDDS-2017
M. Amin *et al.* [46], 2023	No	ACE, DDM, EDDM, STEPD	IoT-23, IoT Malware
P. Wang *et al.* [47], 2020	No	Error rate and False positive rate based	Generated a dataset from 54 sensors set up in Intel Berkeley research lab

respectively. In online streaming mode at time-interval T_1, when a set of streaming flows arrives (Test_set_T_1), it is given as an input to both the drift detector and classifier. The drift detector applies the DD_{model} to find the prediction probability of each flow in the Test_set_T_1. At the same time, it also finds the prediction probability of each flow

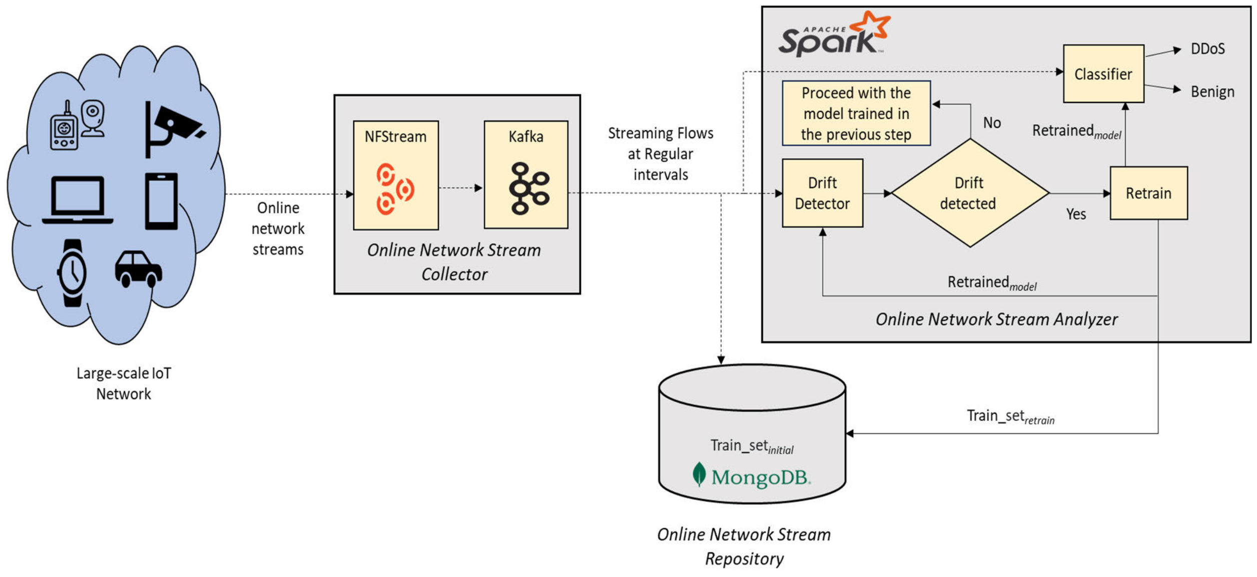

FIGURE 1. **Proposed data pipeline to collect, analyze and store online network traffic streams in the presence of concept drift.**

in the Train_set$_{initial}$. Then, it uses Kolmogorov-Smirnov (KS) test, a non-parametric statistical test, to compare the cumulative distribution function of prediction probabilities of Test_set_T$_1$ and Train_set$_{initial}$ instances. The Kolmogorov-Smirnov test calculates the p-value for each class independently by comparing the prediction probabilities of Train_set$_{initial}$ with Test_set_T$_1$. Then, the drift detector chooses the maximum p-value and checks whether it is lesser than the significance level. If it is lesser than the significance level, then it concludes that there is a drift in the Test_set_T$_1$. Conventionally, the widely used value for significance level of p-value is 0.05 in statistics [53]. Hence, we choose 0.05 for the significance level.

When the drift detector detects the drift, it invokes the retrain subcomponent to retrain the machine learning algorithm to adapt to the novel patterns. The model produced by the retrain subcomponent is called the Retrained$_{model}$. Then, the Retrained$_{model}$ replaces both DD$_{model}$ and Classifier$_{model}$. If the drift detector says there is no drift, it does not invoke the retrain subcomponent. Algorithm 2 explains the working of the drift detector component. In Algorithm 2, Initial$_{model}$ and Train_set$_{initial}$ are used when either the time interval is T$_1$, or there is no drift in the previous time interval. However, Retrained$_{model}$ and Train_set$_{retrain}$ are used instead of Initial$_{model}$ and Train_set$_{initial}$, respectively, after drift is detected at least once.

2) RETRAIN

This subcomponent retrains the machine learning algorithm when drift occurs. Recently, researchers have proposed several techniques to update the model in the event of drift occurrence. Retraining the model is one of the simple and effective techniques that takes the instances causing the drift as input and updates the model to recognize novel patterns. However, there are two challenges that may affect the effectiveness of the retraining technique. The first challenge is that if instances from the previous training are not considered while retraining, the updated model recognizes only the new concepts and fails to recognize old concepts. Due to this, the retrained model's performance will degrade when applied to old concepts. To mitigate this, if we consider all the instances from the set of streaming test instances that caused the drift and the most previous training dataset (MPTS) i.e., old concepts, then the volume of retraining data will increase. Consequently, the time required to retrain the model will also increase. Additionally, the second challenge pertains to retraining time, where the increased volume of retraining data can significantly prolong the retraining process. To tackle these challenges, our retraining technique employs a calculation method that selects an optimal number of instances for every class label. This selection is made from the streaming test instances that caused the drift (Driftset) and the most previous training dataset (MPTS). Next, we present and elucidate the equations used in the proposed work for selecting an optimal number of instances for retraining.

Equation 1 describes the ratio of instances for a specific class in the MPTS relative to the sum of instances for that class in both the MPTS and the Driftset. Equation 2 describes the ratio of instances for a specific class in the Driftset relative to the sum of instances for that class in both MPTS and the Driftset. Equation 3 calculates the total number of instances to be extracted for a specific class in the MPTS. Equation 4 calculates the total number of instances to be selected for a specific class in the Driftset. Equation 5 describes the total instances selected for each class label from both the MPTS and the Driftset. After selecting the optimal number of instances for each class label from the Driftset and MPTS, the retraining process starts with Train_set$_{retrain}$ and

Algorithm 2 Drift Detector Algorithm

```
Input: Initial$_{model}$, Retrained$_{model}$, Train_set$_{initial}$, Train_set$_{retrain}$, Test_set_T$_i$(test set at time interval T$_i$)
Output: True (Existence of drift in Test_set_T$_i$) or False (No drift in Test_set_T$_i$)
1. Start
2. d=0
3. for i=1, 2, 3, 4, 5……,n
    3.1. If the Time interval is equal to T$_1$ or d==0
            Training$_{model}$ = Initial$_{model}$
            Train$_{set}$ =Train_set$_{initial}$
    3.2. If d ==1
            Training$_{model}$ = Retrained$_{model}$
            Train$_{set}$ =Train_set$_{retrain}$
    3.3. Apply Training$_{model}$ on Train$_{set}$ and calculate
         prediction probabilities (pred_prob_train)
    3.4. Apply Training$_{model}$ on Test_set_T$_i$ and
         calculate prediction probabilities
         (pred_prob_test)
    3.5. For each class in Train$_{set}$
            // ks_2samp is Python method for
            // Kolmogorov-Smirnov of 2 samples
            p_value=ks_2samp (pred_prob_train,
                    pred_prob_test)
            Store p_value in p_values set
    3.6. Find maximum p_value (p_value$_{max}$) from
            p_values set
    3.7. If p_value$_{max}$ < 0.05
            Invoke the retrain subcomponent
            d=1
            return True
         else
            return False
```

generates Retrained$_{model}$, as specified in Algorithm 3. Then, this component replaces the old trained model in the drift detector and classifier subcomponents with Retrained$_{model}$.

$$class\ ratioMPTS, class = \frac{InstancesMPTS, class}{InstancesMPTS, class + InstancesDriftset, class} \tag{1}$$

$$class\ ratioDriftset, class = \frac{InstancesDriftset, class}{InstancesMPTS, class + InstancesDriftset, class} \tag{2}$$

$$total\ instancesMPTS, class = class\ ratioMPTS, class \times InstancesMPTS, class \tag{3}$$

$$total\ instancesDriftset, class = classratioDriftset, class \times InstancesDriftset, class \tag{4}$$

$$total\ instancesclass = total\ instancesMPTS, class + total\ instancesDriftset, class \tag{5}$$

C. ONLINE NETWORK STREAM REPOSITORY

This component stores the online streaming flow-based statistical features, Train_set$_{initial}$ and Train_set$_{retrain}$. The online network stream analyzer and retrain subcomponents access this database to perform their task. This database is implemented with MongoDB, a NoSQL database. MongoDB is a widely used document-oriented database well suited to high-performance data pipelines. Further, its rich querying capabilities support efficient retrieval and analysis of historical data for drift detection and analytics. The network administrator can analyze the data stored in this database in real-time to identify and report malicious IoT devices through dashboards.

Algorithm 3 Retrain Algorithm

```
Input: MPTS, Driftset
Output: Train_set$_{retrain}$, Retrained$_{model}$
1. Start
2. Train_set$_{retrain}$ = Empty
3. For Each Class Type
    3.1. Calculate $class\ ratio_{MPTS,Class}$
    3.2. Calculate $class\ ratio_{Driftset,Class}$
    3.3. Calculate $total\ instances_{MPTS,Class}$
    3.4. Calculate $total\ instances_{Driftset,Class}$
    3.5. Calculate $total\ instances_{Class}$
    3.6. Add $total\ instances_{Class}$ to Train_set$_{retrain}$
4. Retrained$_{model}$ = LightGBM(Train_set$_{retrain}$)
5. return Train_set$_{retrain}$, Retrained$_{model}$
```

IV. EXPERIMENTS, EVALUATION, AND DISCUSSION

This section presents and discusses the dataset, the experimental environment, and the performance of the proposed system.

A. EXPERIMENT ENVIRONMENT

PySpark is used to code the proposed system. It is a library that interfaces Python and Apache Spark. The proposed system is executed on a machine with an eight-core Intel i5-1135G7 processor, 8GB RAM, and a 64-bit Ubuntu 22.04.1 LTS operating system. Apache Spark is installed and deployed in the machine in local mode. When the local mode is employed, all the processes of Spark are run on a single machine, optionally using any number of cores on the local system. However, we used a single core to run our experiments.

B. DETAILS ON DATASET

NSL-KDD and IoT23 datasets are used to evaluate the proposed system. This subsection presents the details of both datasets.

1) NSL-KDD DATASET

NSL-KDD is the most preferred publicly available benchmark dataset used by the researchers working in the field of concept drift and network intrusion detection. It is an improved version of the original KDD Cup 1999 dataset [54].

2) IoT23 DATASET

The IoT23 [25] was first made available online in January 2020. IoT23 is a modern unbalanced dataset for malicious IoT device detection. In recent times, it is a popular dataset among the researchers. It comprises 20 malware Packet Capture (PCAP) files and their corresponding labeled flow data. Besides malware captures, it contains three benign PCAP files and their corresponding labeled flow data. IoT23 traffic was captured during 2018 and 2019 in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic.

C. DATASET PREPARATION

To rigorously assess the effectiveness of our proposed technique for detecting and adapting to drift, we meticulously prepared datasets sourced from NSL-KDD and IoT23. The primary objective of this subsection is to provide a comprehensive overview of our dataset preparation process.

1) PREPARATION OF NSL-KDD DATASET

There are 148517 records in the dataset. KDDTrain+.txt contains 125972 records, and KDDTest+.txt contains 22544 records. Each record contains 41 features with label class as 42nd feature. There are 9 categorical and 32 continuous features in the dataset. We converted categorical features into continuous features. The two class labels in the dataset are Benign and Attack. We mapped the Benign label to 0 and Attack to 1. To evaluate the proposed system, we considered the last 10 percent of KDDTrain+.txt as Train_set$_{initial}$. Then, we divided the entire KDDTest+.txt into 9 test sets, namely Test_set1, Test_set2, Test_set3, Test_set4, Test_set5, Test_set6, Test_set7, Test_set8, Test_set9. Each test set consists of 3514 records. It is known that there is a sudden drift in Test_set3 [26].

2) PREPARATION OF IoT23 DATASET

We used labeled flow data from IoT23. Some labeled flow data files are huge and have a greater number of flows. Hence, we have extracted the desired number of flows from those big files. The total number of flows extracted from the original dataset is 15792577. There are 20 features and one class label column in the dataset. We considered ‘orig_ip_bytes’ and ‘orig_pkts’ features to conduct our experiments since these two features scored high feature importance from feature important scoring process. There are many class labels in the dataset. Since our system classifies DDoS flows from benign flows, we extracted only the instances labeled ‘Benign’ or ‘DDoS’ instances. We first randomly sampled 40000 instances for Benign and DDoS separately from the original dataset and generated the training dataset, namely Train_set$_{initial}$. We also randomly sampled another 40000 instances for benign and DDoS separately to generate the dataset, namely Test_set$_{initial}$. This Test_set$_{initial}$ is free from concept drift.

Furthermore, we mapped the Benign label to 0 and DDoS to 1 in the Train_set$_{initial}$ and Test_set$_{initial}$. To conduct drift detection experiments, we synthetically introduced concept drift to the Test_set$_{initial}$ and generated three different test datasets: Test$_{concept1}$, Test$_{concept2}$, and Test$_{concept3}$. Each test dataset exhibits a different concept drift scenario. Table 2 shows how the drift is introduced to Test_set$_{initial}$ and three different test concepts are generated. First, Test$_{concept1}$ was generated by updating the ‘orig_ip_bytes’ column in 5000 randomly sampled instances with a benign class label from the Test_set$_{initial}$. This involved replacing their values with a random selection from the ‘orig_ip_bytes’ column of instances with a DDoS class label in the same Test_set$_{initial}$. Subsequently, Test$_{concept2}$ was derived from Test_set$_{initial}$ by replacing 5000 instances labeled as DDoS with randomly selected 5000 instances labeled as Benign. This dataset was prepared to exhibit the class imbalance feature. Like the Test$_{concept1}$ generation process, Test$_{concept3}$ was generated. However, in this case, we replaced not only the values of the ‘orig_ip_bytes’ column but also the ‘orig_pkts’ column values in the Test_set$_{initial}$.

TABLE 2. Process of preparing Test$_{concept1}$, Test$_{concept2}$ and Test$_{concept3}$ from Test_set$_{initial}$ for the IoT23 dataset for studying the proposed system in different concept drift scenarios.

Generated test concepts from Test_set$_{initial}$	Concept drift introduction process
Test$_{concept1}$	Test$_{concept1}$ is generated by updating the 'orig_ip_bytes' column in 5000 randomly sampled instances with a benign class label from the Test_set$_{initial}$. This involved replacing their values with a random selection from the 'orig_ip_bytes' column of instances with a DDoS class label in the same Test_set$_{initial}$
Test$_{concept2}$	Test$_{concept2}$ is derived from Test_set$_{initial}$ by replacing 5000 instances labeled as DDoS with randomly selected 5000 instances labeled as Benign. This dataset was prepared to exhibit the class imbalance feature
Test$_{concept3}$	Like the Test$_{concept1}$ generation process, Test$_{concept3}$ is generated. However, in this case, we replaced not only the values of the 'orig_ip_bytes' column but also the 'orig_pkts' column values in the Test_set$_{initial}$

D. EXPERIMENTS

This section presents the various experiments conducted to study the performance of the proposed system. We compared the performance of our proposed system with OASW [26], a recent drift detector, and an adapter in the IoT network. Since OASW used LightGBM [55], a highly efficient gradient boosting decision tree, as its machine learning algorithm, we also chose it to train our classifier. OASW algorithm maintains two sliding windows, namely the current and previous windows. The size of these windows is determined through experiments. If the accuracy of the current window is dropping alpha percent, then the warning level is triggered. After that, if the current window's accuracy keeps dropping, the drift alarm is triggered. The design of OASW allows it to detect instances causing the drift. OASW requires the true labels of incoming test instances to detect the drift. Further, the drift detection process in OASW confirms the occurrence of drift when classification accuracy falls below a certain threshold. Conversely, our proposed system does not rely on true labels to detect the drift. It confirms the occurrence of drift by analyzing the distribution of prediction probabilities of training and testing instances. Hence, our system is more suitable for practical scenarios where true labels of incoming test instances are generally unknown.

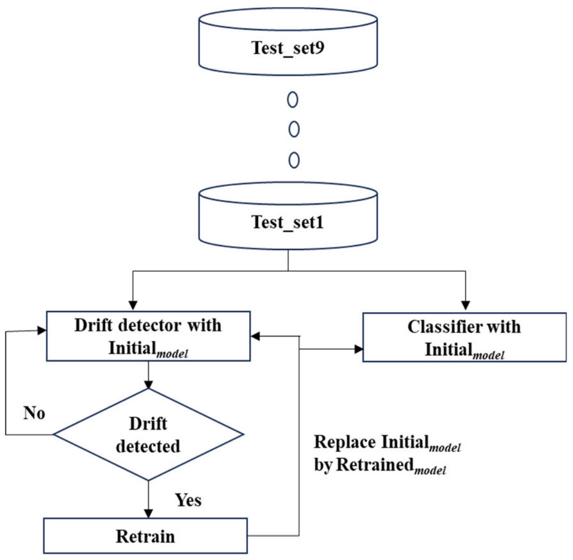
Test_set9
Test_set1
Drift detector with Initial$_{model}$
Classifier with Initial$_{model}$
No
Drift detected
Yes
Retrain
Replace Initial$_{model}$ by Retrained$_{model}$

FIGURE 2. Experiment process on NSL-KDD dataset.

1) EXPERIMENT ON NSL-KDD DATASET

Figure 2 illustrates the experiment process on NSL-KDD dataset. In offline mode, we applied LightGBM on Train_set$_{initial}$ to produce the Initial$_{model}$.Then, we streamed 9 test sets at regular intervals. That is, we streamed Test_set1 at time interval T_1, Test_set2 at T_2 and so on. As each test set is streamed, we applied drift detector subcomponent to calculate p-value. Figure 3 shows the p- value obtained for each test set. It can be seen from Figure 3 that the p-value for Test_set1 and Test_set2 is above 0.05. This indicates that there is no drift in Test_set1 and Test_set2. However, p-value obtained for Test_set3, Test_set4, Test_set5, Test_set6, Test_set7, Test_set8, Test_set9 is below 0.05. This indicates there is a drift in Test_set3 and it continues till Test_set9 if drift adaptation (i.e., retraining) is not performed. When drift is detected at T_3, we retrained the model by following Algorithm 3. Then, the retrained model is applied on Test_set3. As a result, the p-value obtained for Test_set3 is above 0.05. After retraining process, when we applied the retrained model to the subsequent test sets, we observed p-values crossing 0.05.

To compare our proposed system, UASDAC, with OASW, we executed OASW on NSL-KDD dataset as described in [26]. As a result, OASW generated the plot shown in Figure 4. The accuracies obtained by UASDAC is plotted in Figure 5. The average accuracy obtained by UASDAC is 97.69% which is 0.08% above when compared to OASW. To test the performance of UASDAC and OASW when old concepts are coming back, we streamed Train_set$_{initial}$ at T_{10}. As UASDAC retrains the model with the portion of instances causing the drift and the most previous training set when the drift occurs, it did not see any drift in the Train_set$_{initial}$ at T_{10}. Hence, retraining is not performed at T_{10}. However, as OASW considers only the instances causing the drift for retraining, it reports drift in Train_set$_{initial}$. Hence, it retrains the model one more time.

TABLE 3. Total time taken by UASDAC and OASW in drift detection and adaptation on NSL-KDD dataset.

System	Time in Seconds
UASDAC	1.67
OASW	34.32

Table 3 documents the time taken by UASDAC and OASW. It is evident that our proposed system, UASDAC, outperforms OASW in terms of time efficiency. UASDAC not only achieves superior accuracy but also completes the task in a shorter amount of time. This efficiency of UASDAC can be attributed to predict labels for all incoming test instances simultaneously. In contrast, OASW predicts labels instance by instance using River, a streaming Python library, which measures the model's accuracy as it processes each instance in the test dataset. If drift is detected during processing, OASW invokes retraining. This allows OASW to detect multiple drifts in a test dataset and engage in multiple retraining cycles. However, UASDAC identifies only one drift for the entire test dataset, leading to a single retraining cycle. Additionally, UASDAC is implemented on Spark, enabling distributed processing of the dataset. These factors contribute to UASDAC completing its task more quickly than OASW.

2) EXPERIMENT ON IoT23 DATASET

Now, we present the experiments conducted on IoT23 dataset. Figure 6 illustrates the experiment process on IoT23 dataset. In offline mode, we applied LightGBM on Train_set$_{initial}$ to produce the Initial$_{model}$. We applied the Initial$_{model}$ on Test_set$_{initial}$ and obtained the accuracy of 99.96%. Since there is no concept drift in Test_set$_{initial}$, we could achieve the accuracy of 99.96%. However, there will be drop in accuracy when Initial$_{model}$ is applied on three test concepts: Test$_{concept1}$, Test$_{concept2}$ and Test$_{concept3}$.

a: EXPERIMENT1: ONLINE STREAMING OF TEST$_{concept1}$

In this experiment, we investigate the performance of our proposed system under the assumption that the incoming

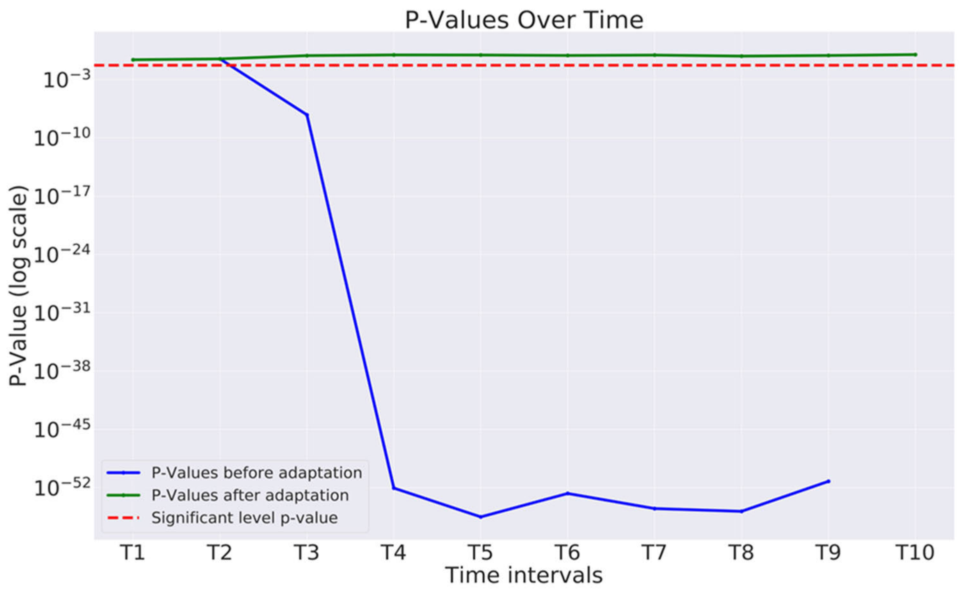

FIGURE 3. P-values obtained by drift detector subcomponent of UASDAC before and after adaptation.

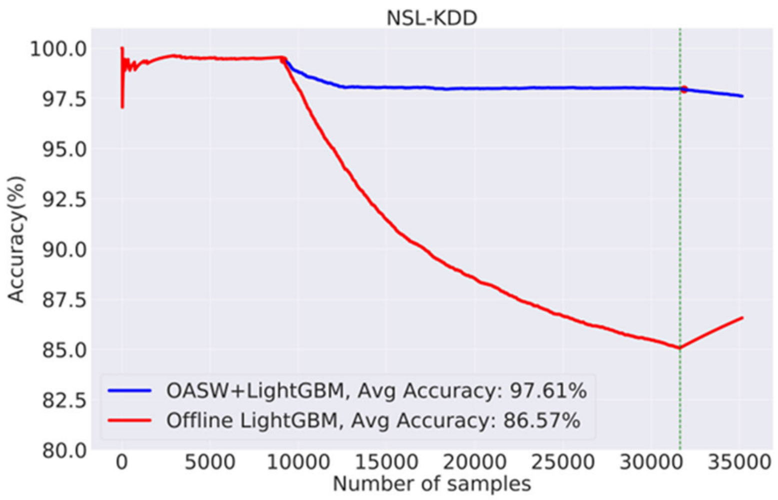

FIGURE 4. Accuracy achieved by OASW. Red dot indicates drift point. Green dashed line indicates the beginning of old concepts.

online stream may exhibit concept drift. To find whether the proposed drift detector finds the concept drift, we streamed $Test_{concept1}$ to the drift detector and classifier subcomponents. It is considered that when $Test_{concept1}$ is streamed, the $Initial_{model}$ is in use in both the drift detector and classifier subcomponents. Then, the drift detector subcomponent applied the Kolmogorov-Smirnov test on $Test_{concept1}$ and $Train\_set_{initial}$ to calculate the p-value. The p-value obtained is closer to zero. Figure 7 shows the obtained p-value. Figure 8 shows the Empirical Cumulative Distribution Function plot (ECDF) for $Test_{concept1}$ and $Train\_set_{initial}$. It indicates a significant difference in the distribution of prediction probabilities between the $Test_{concept1}$ and $Train\_set_{initial}$, leading to concept drift detection. At the same time, the classifier subcomponent predicted the class label for the instances in $Test_{concept1}$ and calculated the accuracy. The accuracy obtained is 93.72%.

Meanwhile, since the drift is detected in $Test_{concept1}$, the drift detector subcomponent invoked the retrain subcomponent. As a result, the retrain subcomponent calculated the number of sample instances to be extracted for each class from the $Test_{concept1}$ and MPTS for retraining. It is crucial to note that $Test_{concept1}$ is considered as Driftset since the instances from $Test_{concept1}$ cause the observed drift. Similarly, $Train\_set_{initial}$ is considered as MPTS.

Table 4 shows the number of instances extracted for each class from MPTS and Driftset for retraining. The number of instances presented in Table 4 constitute

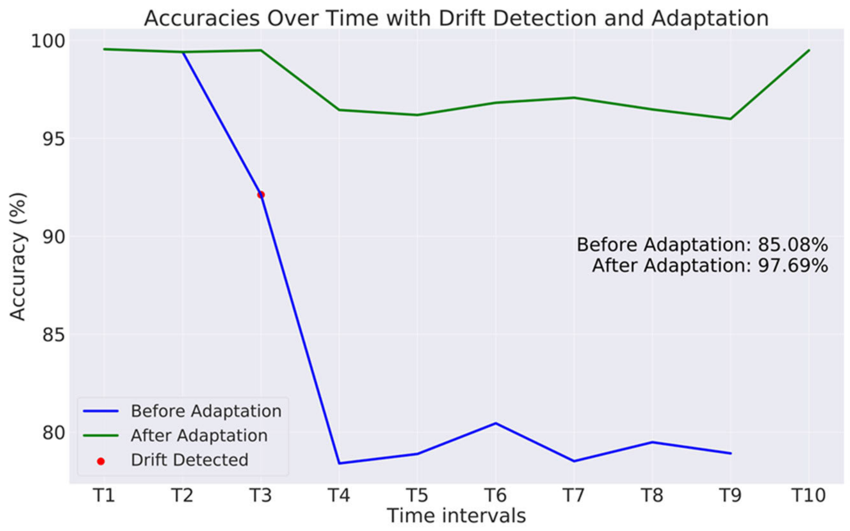

FIGURE 5. Accuracy achieved by UASDAC. Red dot indicates drift point.

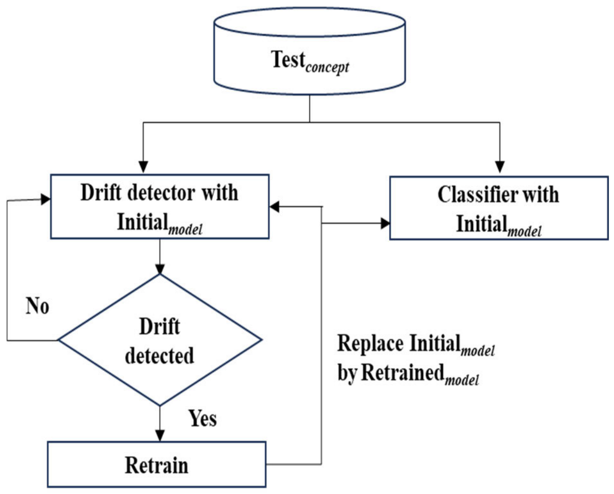
Test$_{concept}$
Drift detector with Initial$_{model}$
Classifier with Initial$_{model}$
Drift detected
No
Yes
Retrain
Replace Initial$_{model}$ by Retrained$_{model}$

FIGURE 6. Experiment process on IoT23 dataset.

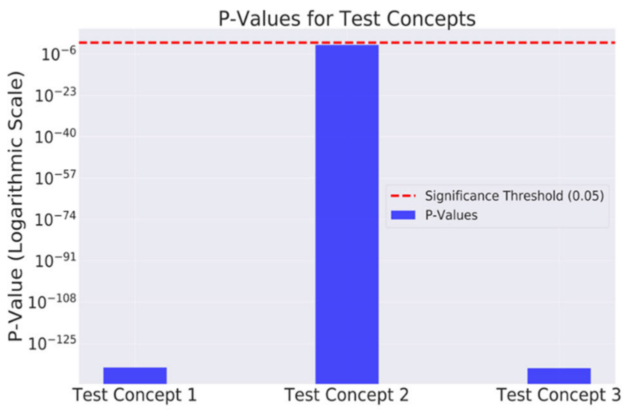

FIGURE 7. p-values obtained for Test$_{concept1}$, Test$_{concept2}$, Test$_{concept3}$.

Train_set$_{retrain}$. After this, the retrain subcomponent creates the Retrained$_{model}$ by retraining LightGBM on

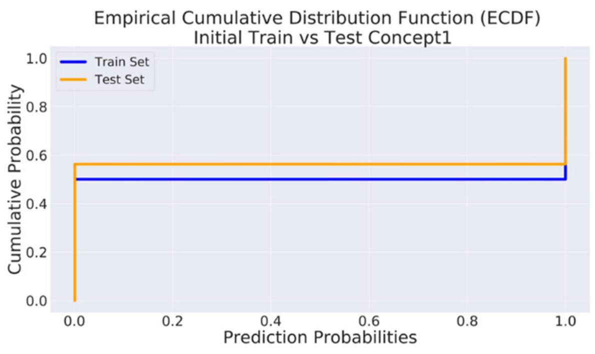

FIGURE 8. ECDF plot to show the distribution of prediction probabilities between the Test$_{concept1}$ and Train_set$_{initial}$.

TABLE 4. Number of instances from MPTS and Driftset for retraining in Experiment1.

Class label (0-Benign, 1-DDoS)	Number of instances selected from MPTS	Number of instances selected from Driftset
0	20000	20000
1	20000	20000

Train_set$_{retrain}$. Subsequently, the Retrained$_{model}$ replaces the Initial$_{model}$ in the drift detector and classifier subcomponents. Then, this Retrained$_{model}$ will be used in subsequent intervals until the next drift occurs.

After replacing the Initial$_{model}$ in the classifier subcomponent with the Retrained$_{model}$, we tested the classifier on Test$_{concept1}$ to verify whether the classifier can recognize the novel patterns in Test$_{concept1}$. During this investigation, we observed that the classifier with the Retrained$_{model}$ improved the accuracy from 93.72% to 99.89%.

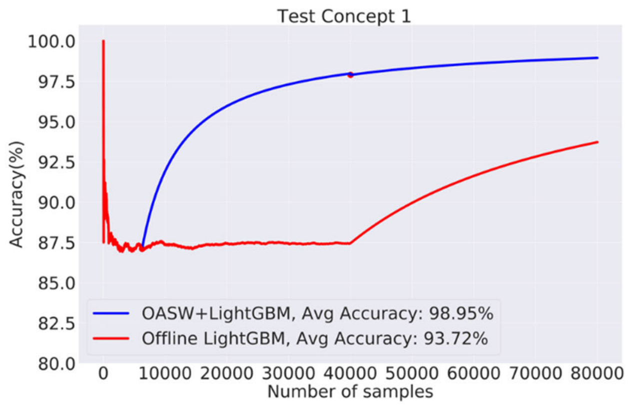

FIGURE 9. OASW plot for Test$_{concept1}$.

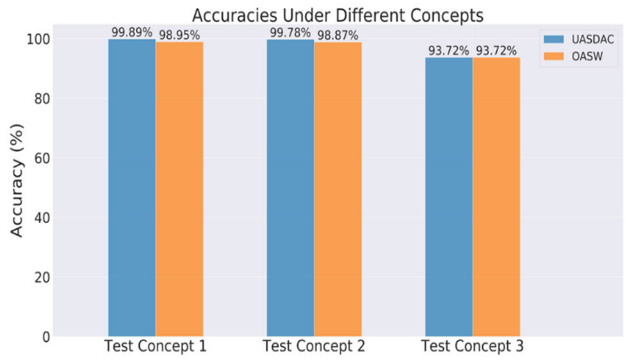

FIGURE 10. Accuracy comparison between UASDAC and OASW in different concept drift scenarios on IoT23 dataset.

To compare our proposed system, UASDAC, with OASW, we presented the Train_set$_{initial}$ and Test$_{concept1}$ to OASW for evaluation. As a result, OASW generated the plot shown in Figure 9.

It can be observed from Figure 9 that offline LightGBM achieved an accuracy of only 93.72%. This accuracy is achieved when drift detection and adaptation are not employed. However, when drift detection and adaptation are implemented, the accuracy is improved from 93.72% to 98.95%.

Figure 10 and Figure 11 show the accuracy and the time taken by UASDAC and OASW respectively for all the experiments on IoT23 dataset. It is evident that our proposed system, UASDAC, outperforms OASW in terms of both accuracy and time efficiency. UASDAC not only achieves superior accuracy but also completes the task in a shorter amount of time. This efficiency can be attributed to UASDAC requiring only a single retraining iteration when drift occurs in the test data set. At the same time, OASW engages in multiple retraining cycles and processes the test data set instance by instance.

b: EXPERIMENT2: ONLINE STREAMING OF TEST$_{concept2}$

In real-life practical scenarios, the incoming network traffic may consist of more instances for one class label than other class labels. This is called the class imbalance problem. Hence, it is essential to study the performance of drift detector and adaptation techniques in imbalanced online test streams. Therefore, this experiment is designed to consider the Test$_{concept2}$, an imbalanced test set, as input to Initial$_{model}$. Our drift detector algorithm successfully detected the drift when applied on Train_set$_{initial}$ and Test$_{concept2}$. The drift detector subcomponent applied the Kolmogorov-Smirnov test on Test$_{concept2}$ and Train_set$_{initial}$ to calculate the p-value.

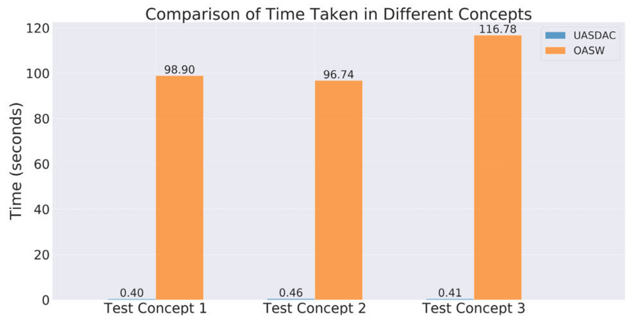

FIGURE 11. Time taken comparison between UASDAC and OASW in different concept drift scenarios on IoT23 dataset.

Figure 7 shows the obtained P-value. The p-value for Test$_{concept2}$ may not be as close to zero as other test concepts, it still falls below the threshold value of 0.05, indicating significant concept drift detection. Figure 12 shows the Empirical Cumulative Distribution Function plot for Test$_{concept2}$ and Train_set$_{initial}$. Though there is not much difference observed in the plot it still indicates a significant difference in the distribution of prediction probabilities between the Test$_{concept2}$ and Train_set$_{initial}$, leading to concept drift detection. At the same time, the classifier subcomponent predicted the class label for the instances in Test$_{concept2}$ and calculated the accuracy. The accuracy obtained is 92.88%. Since the drift is detected, the retrain subcomponent is invoked, and it retrained the model with the Train_set$_{retrain}$. The details about the number of instances used for retraining are presented in Table 5. The number of instances presented in Table 5 constitute Train_set$_{retrain}$. After this, the retrain subcomponent creates the Retrained$_{model}$ by retraining LightGBM on Train_set$_{retrain}$. Subsequently, the Retrained$_{model}$ replaces the Initial$_{model}$ in the drift detector and classifier subcomponents. Then, this Retrained$_{model}$ will be used in subsequent intervals until the next drift occurs.

After replacing the Initial$_{model}$ in the classifier subcomponent with the Retrained$_{model}$, we tested the classifier on Test$_{concept2}$ to verify whether the classifier can recognize the novel patterns in Test$_{concept2}$. During this investigation, we observed that the classifier with the Retrained$_{model}$ improved the accuracy from 92.88% to 99.78%.

For comparison, we presented the Train_set$_{initial}$ and Test$_{concept2}$ to OASW. The plot generated by OASW is shown in Figure 13. The performance comparison of UASDAC and OASW is shown in Figure 10 and Figure 11. Once again, like

TABLE 5. Number of instances from MPTS and Driftset for retraining in Experiment2.

Class label (0-Benign, 1-DDoS)	Number of instances selected from MPTS	Number of instances selected from Driftset
0	18800	23850
1	21200	16450

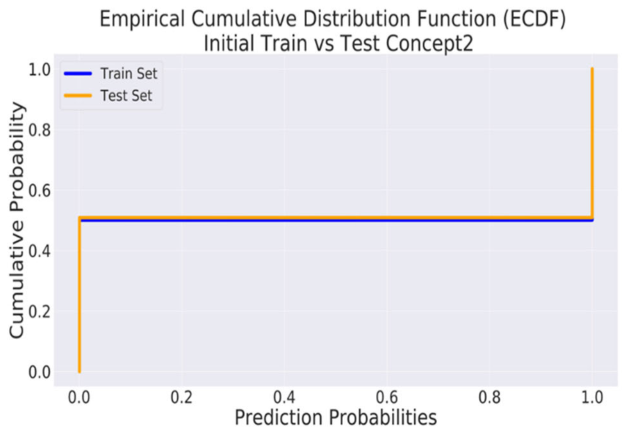

FIGURE 12. ECDF plot to show the distribution of prediction probabilities between the Test$_{concept2}$ and Train_set$_{initial}$.

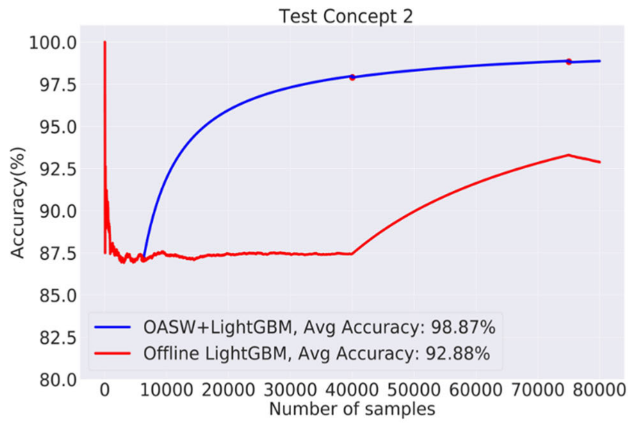

FIGURE 13. OASW plot for Test$_{concept2}$.

in experiment 1, our system UASDAC outperformed OASW in terms of both accuracy and time efficiency. To measure the robustness of the retrained model, we evaluated the performance of the retrained model with the test set that consists of 35000 benign instances and 45000 DDoS instances. The evaluation generated the accuracy of 99.97%.

c: EXPERIMENT3: ONLINE STREAMING OF TEST$_{concept3}$

The scenario for this experiment is the same as experiment 1. However, we streamed Test$_{concept3}$ to the drift detector and classifier subcomponents. As in experiment 1, the drift detector subcomponent successfully detected the drift. The p-value obtained is closer to zero. Figure 7 shows the obtained p-value. Figure 14 shows the Empirical

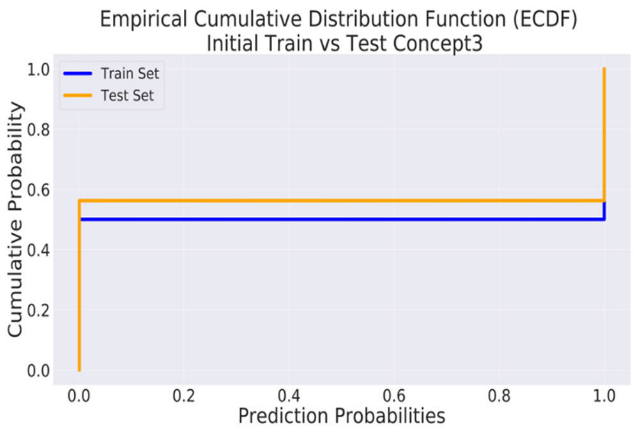

FIGURE 14. ECDF plot to show the distribution of prediction probabilities between the Test$_{concept3}$ and Train_set$_{initial}$.

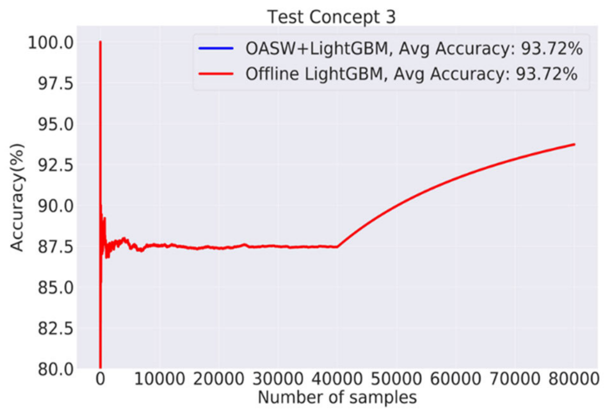

FIGURE 15. OASW plot for Test$_{concept3}$.

Cumulative Distribution Function plot for Test$_{concept3}$ and Train_set$_{initial}$. It indicates a significant difference in the distribution of prediction probabilities between the Test$_{concept3}$ and Train_set$_{initial}$, leading to concept drift detection. At the same time, the classifier subcomponent predicted the class label for the instances in Test$_{concept3}$ and calculated the accuracy. The accuracy obtained is 93.71%. However, OASW failed to detect the drift in Test$_{concept3}$. This inability of OASW can be observed in the plot given in Figure 15. Figure 15 shows the same accuracy for offline LightGBM and online LightGBM. Also, it can be observed that there are no detected drift points in the plot. Since our drift detector found the drift in Test$_{concept3}$, we prepared Train_set$_{retrain}$ as explained in experiment1. However, in this case, we used Test$_{concept3}$ instead of Test$_{concept1}$ while preparing the Train_set$_{retrain}$. After retraining with Train_set$_{retrain}$, when we applied the classifier with the Retrained$_{model}$, we observed that the classifier could not recognize the new patterns in the Test$_{concept3}$. This is because 5000 instances of the benign class share identical values for the features 'orig_ip_bytes' and 'orig_pkts' with instances labeled as DDoS. As a result, retraining does not improve the classifier's performance, unlike in Experiment 1.

V. CONCLUSION

This research article introduced the UASDAC, an adaptive and scalable system, which classifies DDoS traffic and Benign traffic in massive online IoT network streams in the presence of concept drift. The system employs Kolmogorov-Smirnov test, an unsupervised technique, to detect concept drift in online network streams in regular intervals. If the drift is detected, the system retrains the classifier model by selecting an optimal number of instances from both the set of instances causing the drift and the most previous training. The performance of the system is evaluated on NSL-KDD and IoT-23 datasets in different concept drift scenarios. Notably, UASDAC outperformed OASW in terms of accuracy, achieving an improvement of 0.08% in NSL-KDD dataset and 0.91% to 0.94% in IoT23 dataset. In terms of time efficiency, our UASDAC showcased remarkable performance gains over the existing method in both datasets. For NSL-KDD, our technique completed tasks in a mere 1.67 seconds, whereas the OASW required a significantly longer time of 34.32 seconds. In IoT23, the efficiency gains were even more pronounced, with our method taking only 0.45 seconds compared to the substantial 116.78 seconds taken by OASW.

Currently, UASDAC assumes that drifting instances come from classes present in the training dataset. However, in scenarios where new types of attack classes emerge, the drifting instances may belong to classes not included in the training dataset. UASDAC is not equipped to handle this type of concept drift in its current form. Therefore, for future research, we aim to enhance UASDAC to accommodate multiclass attack scenarios from real-world settings. In addition, future work will focus on exploring the effectiveness of alternative machine learning models to further enhance UASDAC's adaptability to concept drift in IoT networks.

ACKNOWLEDGMENT

The authors would like to thank Amrita Vishwa Vidyapeetham for facilitating the research.

CONFLICT OF INTEREST

The authors hereby declare the absence of any competing interest.

REFERENCES

[1] H. Chunduri, T. G. Kumar, and P. V. S. Charan, "A multi class classification for detection of IoT botnet malware," in *Proc. Intl. Conf. Comput. Sci. Commun. Secur.*, May 2021, pp. 17–29, doi: 10.1007/978-3-030-76776-1_2.

[2] P. R. K. Pranav, S. Verma, S. Shenoy, and S. Saravanan, "Detection of botnets in IoT networks using graph theory and machine learning," in *Proc. 6th Int. Conf. Trends Electron. Informat. (ICOEI)*, Tirunelveli, India, Apr. 2022, pp. 590–597, doi: 10.1109/ICOEI53556.2022.9777117.

[3] R. Vinayakumar, M. Alazab, S. Srinivasan, Q.-V. Pham, S. K. Padannayil, and K. Simran, "A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4436–4456, Jul. 2020, doi: 10.1109/TIA.2020.2971952.

[4] *Nokia Threat Intelligence Report Finds Malicious IoT Botnet Activity Has Sharply Increased*. Accessed: Dec. 26, 2023. [Online]. Available: https://www.nokia.com/about-us/news/releases/2023/06/07/nokia-threat-intelligence-report-finds-malicious-iot-botnet-activity-has-sharply-increased/

[5] T. Tu, J. Qin, H. Zhang, M. Chen, T. Xu, and Y. Huang, "A comprehensive study of mozi botnet," *Int. J. Intell. Syst.*, vol. 37, no. 10, pp. 6877–6908, Mar. 2022, doi: 10.1002/int.22866.

[6] A. A. Cook, G. Misirli, and Z. Fan, "Anomaly detection for IoT time-series data: A survey," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6481–6494, Jul. 2020, doi: 10.1109/JIOT.2019.2958185.

[7] G. Wang, M. Nixon, and M. Boudreaux, "Toward cloud-assisted industrial IoT platform for large-scale continuous condition monitoring," *Proc. IEEE*, vol. 107, no. 6, pp. 1193–1205, Jun. 2019, doi: 10.1109/JPROC.2019.2914021.

[8] E. Zeydan and J. Mangues-Bafalluy, "Recent advances in data engineering for networking," *IEEE Access*, vol. 10, pp. 34449–34496, 2022.

[9] G. V. Arbex, K. G. Machado, M. Nogueira, D. M. Batista, and R. Hirata, "IoT DDoS detection based on stream learning," in *Proc. 12th Int. Conf. Netw. Future (NoF)*, Coimbra, Portugal, Oct. 2021, pp. 1–8, doi: 10.1109/NoF52522.2021.9609940.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, Mar. 2014, doi: 10.1145/2523813.

[11] A. G. Menon and G. Gressel, "Concept drift detection in phishing using autoencoders," in *Proc. Symp. Mach. Learn. Metaheuristics Algorithms, Appl.* Singapore: Springer, Feb. 2021, pp. 208–220, doi: 10.1007/978-981-16-0419-5_17.

[12] H. Mehmood, P. Kostakos, M. Cortes, T. Anagnostopoulos, S. Pirttikangas, and E. Gilman, "Concept drift adaptation techniques in distributed environment for real-world data streams," *Smart Cities*, vol. 4, no. 1, pp. 349–371, Mar. 2021, doi: 10.3390/smartcities4010021.

[13] P. Pradeep, S. Krishnamoorthy, and A. V. Vasilakos, "A holistic approach to a context-aware IoT ecosystem with adaptive ubiquitous middleware," *Pervas. Mobile Comput.*, vol. 72, Apr. 2021, Art. no. 101342, doi: 10.1016/j.pmcj.2021.101342.

[14] O. Abdel Wahab, "Intrusion detection in the IoT under data and concept drifts: Online deep learning approach," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 19706–19716, Oct. 2022, doi: 10.1109/JIOT.2022.3167005.

[15] K. Rajora and N. S. Abdulhussein, "Reviews research on applying machine learning techniques to reduce false positives for network intrusion detection systems," *Babylonian J. Mach. Learn.*, vol. 2023, pp. 26–30, May 2023, doi: 10.58496/bjml/2023/005.

[16] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comparative study on online machine learning techniques for network traffic streams analysis," *Comput. Netw.*, vol. 207, Apr. 2022, Art. no. 108836, doi: 10.1016/j.comnet.2022.108836.

[17] A. Adnan, A. Muhammed, A. A. A. Ghani, A. Abdullah, and F. Hakim, "An intrusion detection system for the Internet of Things based on machine learning: Review and challenges," *Symmetry*, vol. 13, no. 6, p. 1011, Jun. 2021, doi: 10.3390/sym13061011.

[18] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019, doi: 10.1109/TKDE.2018.2876857.

[19] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. M. dos Santos, "An overview of unsupervised drift detection methods," *WIREs Data Mining Knowl. Discovery*, vol. 10, no. 6, p. e1381, Nov. 2020, doi: 10.1002/widm.1381.

[20] V. M. Nidhi, V. Gupta, and R. Vig, "Methods to investigate concept drift in big data streams," in *Knowledge Computing and Its Applications: Knowledge Manipulation and Processing Techniques*, vol. 1. Singapore: Springer, Feb. 2018, pp. 51–74, doi: 10.1007/978-981-10-6680-13.

[21] A. Naghib, N. Jafari Navimipour, M. Hosseinzadeh, and A. Sharifi, "A comprehensive and systematic literature review on the big data management techniques in the Internet of Things," *Wireless Netw.*, vol. 29, no. 3, pp. 1085–1144, Nov. 2022, doi: 10.1007/s11276-022-03177-5.

[22] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A survey," *Int. J. Inf. Manage.*, vol. 45, pp. 289–307, Apr. 2019, doi: 10.1016/j.ijinfomgt.2018.08.006.

[23] M. G. Yaseen and A. S. Albahri, "Mapping the evolution of intrusion detection in big data: A bibliometric analysis," *Mesopotamian J. Big Data*, vol. 2023, pp. 138–148, Dec. 2023, doi: 10.58496/mjbd/2023/018.

[24] *Datasets | Research | Canadian Institute for Cybersecurity | UNB*. Accessed: Jan. 5, 2024. [Online]. Available: http://nsl.cs.unb.ca/NSL-KDD/

[25] S. Garcia, A. Parmisano, and M. J. Erquiaga, Jan. 20, 2020, "IoT-23: A labeled dataset with malicious and benign IoT network traffic (version 1.0.0)," *Zenodo*, doi: 10.5281/zenodo.4743746.

[26] L. Yang and A. Shami, "A lightweight concept drift detection and adaptation framework for IoT data streams," *IEEE Internet Things Mag.*, vol. 4, no. 2, pp. 96–101, Jun. 2021, doi: 10.1109/IOTM.0001.2100012.

[27] B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu, and Z. Li, "Machine-learning-based online distributed denial-of-service attack detection using spark streaming," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6, doi: 10.1109/ICC.2018.8422327.

[28] N. V. Patil, C. R. Krishna, and K. Kumar, "SSK-DDoS: Distributed stream processing framework based classification system for DDoS attacks," *Cluster Comput.*, vol. 25, no. 2, pp. 1355–1372, Jan. 2022, doi: 10.1007/s10586-022-03538-x.

[29] W.-C. Shih, C.-T. Yang, C.-T. Jiang, and E. Kristiani, "Implementation and visualization of a netflow log data lake system for cyberattack detection using distributed deep learning," *J. Supercomput.*, vol. 79, no. 5, pp. 4983–5012, Oct. 2022, doi: 10.1007/s11227-022-04802-y.

[30] Z. Shi, J. Li, C. Wu, and J. Li, "DeepWindow: An efficient method for online network traffic anomaly detection," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City, IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 2403–2408, doi: 10.1109/HPCC/SmartCity/DSS.2019.00335.

[31] A. Abid, F. Jemili, and O. Korbaa, "Real-time data fusion for intrusion detection in industrial control systems based on cloud computing and big data techniques," *Cluster Comput.*, vol. 27, no. 2, pp. 2217–2238, Jun. 2023, doi: 10.1007/s10586-023-04087-7.

[32] R. Alghamdi and M. Bellaiche, "An ensemble deep learning based IDS for IoT using lambda architecture," *Cybersecurity*, vol. 6, no. 1, p. 32, Mar. 2023, doi: 10.1186/s42400-022-00133-w.

[33] A. Yahyaoui, H. Lakhdhar, T. Abdellatif, and R. Attia, "Machine learning based network intrusion detection for data streaming IoT applications," in *Proc. 21st ACIS Int. Winter Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distributed Comput. (SNPD-Winter)*, Jan. 2021, pp. 51–56, doi: 10.1109/SNPDWinter52325.2021.00019.

[34] M. Jain and G. Kaur, "Distributed anomaly detection using concept drift detection based hybrid ensemble techniques in streamed network data," *Cluster Comput.*, vol. 24, no. 3, pp. 2099–2114, Feb. 2021, doi: 10.1007/s10586-021-03249-9.

[35] Z. Shao, S. Yuan, and Y. Wang, "Adaptive online learning for IoT botnet detection," *Inf. Sci.*, vol. 574, pp. 84–95, Oct. 2021, doi: 10.1016/j.ins.2021.05.076.

[36] L. Yang, D. M. Manias, and A. Shami, "PWPAE: An ensemble framework for concept drift adaptation in IoT data streams," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6, doi: 10.1109/GLOBECOM46510.2021.9685338.

[37] B. H. Schwengber, A. Vergütz, N. G. Prates, and M. Nogueira, "Learning from network data changes for unsupervised botnet detection," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 1, pp. 601–613, Mar. 2022, doi: 10.1109/TNSM.2021.3109076.

[38] L. Yang and A. Shami, "A multi-stage automated online network data stream analytics framework for IIoT systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2107–2116, Feb. 2023, doi: 10.1109/TII.2022.3212003.

[39] H. Qiao, B. Novikov, and J. O. Blech, "Concept drift analysis by dynamic residual projection for effectively detecting botnet cyber-attacks in IoT scenarios," *IEEE Trans. Ind. Informat.*, vol. 18, no. 6, pp. 3692–3701, Jun. 2022, doi: 10.1109/TII.2021.3108464.

[40] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," in *Proc. 30th USENIX Secur. Symp. (USENIX)*, 2021, pp. 2327–2344.

[41] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "INSOMNIA: Towards concept-drift robustness in network intrusion detection," in *Proc. 14th ACM Workshop Artif. Intell. Secur.* New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 111–122, doi: 10.1145/3474369.3486864.

[42] L. Korycki and B. Krawczyk, "Concept drift detection from multi-class imbalanced data streams," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Chania, Greece, Apr. 2021, pp. 1068–1079, doi: 10.1109/icde51399.2021.00097.

[43] W. Liu, C. Zhu, Z. Ding, H. Zhang, and Q. Liu, "Multiclass imbalanced and concept drift network traffic classification framework based on online active learning," *Eng. Appl. Artif. Intell.*, vol. 117, Jan. 2023, Art. no. 105607, doi: 10.1016/j.engappai.2022.105607.

[44] L. Xu, X. Ding, H. Peng, D. Zhao, and X. Li, "ADTCD: An adaptive anomaly detection approach towards concept-drift in IoT," *IEEE Internet Things J.*, vol. 10, no. 18, pp. 15931–15942, Sep. 2023, doi: 10.1109/JIOT.2023.3265964.

[45] M. Jain, G. Kaur, and V. Saxena, "A K-means clustering and SVM based hybrid concept drift detection technique for network anomaly detection," *Expert Syst. Appl.*, vol. 193, May 2022, Art. no. 116510, doi: 10.1016/j.eswa.2022.116510.

[46] M. Amin, F. Al-Obeidat, A. Tubaishat, B. Shah, S. Anwar, and T. A. Tanveer, "Cyber security and beyond: Detecting malware and concept drift in AI-based sensor data streams using statistical techniques," *Comput. Electr. Eng.*, vol. 108, May 2023, Art. no. 108702, doi: 10.1016/j.compeleceng.2023.108702.

[47] P. Wang, N. Jin, and G. Fehringer, "Concept drift detection with false positive rate for multi-label classification in IoT data stream," in *Proc. Int. Conf. U.K.-China Emerg. Technol. (UCET)*, Aug. 2020, pp. 1–4, doi: 10.1109/UCET51115.2020.9205421.

[48] Z. Aouini and A. Pekar, "NFStream," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108719, doi: 10.1016/j.comnet.2021.108719.

[49] Z. Wang, W. Dai, F. Wang, H. Deng, S. Wei, X. Zhang, and B. Liang, "Kafka and its using in high-throughput and reliable message distribution," in *Proc. 8th Int. Conf. Intell. Netw. Intell. Syst. (ICINIS)*, Nov. 2015, pp. 117–120, doi: 10.1109/ICINIS.2015.53.

[50] T. P. Raptis and A. Passarella, "A survey on networked data streaming with Apache Kafka," *IEEE Access*, vol. 11, pp. 85333–85350, 2023, doi: 10.1109/ACCESS.2023.3303810.

[51] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016, doi: 10.1145/2934664.

[52] M. Armbrust, T. Das, J. Torres, B. Yavuz, S. Zhu, R. Xin, A. Ghodsi, I. Stoica, and M. Zaharia, "Structured streaming: A declarative API for real-time applications in Apache Spark," in *Proc. Int. Conf. Manage. Data*, May 2018, pp. 601–613, doi: 10.1145/3183713.3190664.

[53] P. Porwik and B. M. Dadzie, "Detection of data drift in a two-dimensional stream using the Kolmogorov–Smirnov test," *Proc. Comput. Sci.*, vol. 207, pp. 168–175, Jan. 2022, doi: 10.1016/j.procs.2022.09.049.

[54] (Oct. 2007). *KDD Cup 1999*. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[55] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.

SARAVANAN SELVAM received the Master of Engineering degree in computer science from Sathyabama University, Chennai, India. He is currently an Assistant Professor with the Department of Computer Science and Engineering, Amrita School of Computing, Chennai. He has 19 years of teaching experience. His research interests include large-scale cybersecurity analytics, design and development of big data technology-based applications, and streaming data analytics.

UMA MAHESWARI BALASUBRAMANIAN (Senior Member, IEEE) received the B.E. degree in computer science and engineering from Bharathidasan University, in 1993, the M.E. degree in computer science and engineering from Anna University, Chennai, India, in 2004, and the Ph.D. degree in computer science and engineering from Amrita Vishwa Vidyapeetham, India, in 2020. She is currently an Assistant Professor with the Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru, India. Her current research interests include digital twin, machine/deep learning, the IoT applications in healthcare, agriculture, network security, overlay networks, P2P video streaming, application layer multicasting in wired, and wireless networks.