A Robustness Evaluation of Concept Drift Detectors against Unreliable Data Streams

Sixiang Wang and Fumio Machida
Department of Computer Science
University of Tsukuba
Tsukuba, Japan
{s2020668@s and machida@cs}.tsukuba.ac.jp

***Abstract*— Machine learning algorithms have been widely used in IoT application systems for predicting labels of input data streams from IoT sensors. Since data streams are not always stationary in real-world scenarios, underlying data distribution changes may cause a deterioration of the prediction performance that is known as concept drift. Existing concept drift detection methods often assume that complete and true labels for detecting the prediction errors are always available immediately after the prediction. However, such an assumption is not realistic in real IoT application systems. This paper experimentally investigates the robustness of six representative concept drift detectors against unreliable data streams containing the data with error labels and the data with no labels. Robustness is evaluated in terms of the average detection delay and precision by increasing the ratios of error-labeled and missing-label data in a synthetic data stream. Our experimental results show that Cumulative Sum (CUSUM), Page-Hinkley (PH), and Drift Detection Method (DDM) achieve relatively stable performances when the ratio of error-labeled data is less than 40%. With respect to the drift detection efficiency, CUSUM can be regarded as the most robust detector among other detectors tested in our experiments.**

Keywords—classifier, concept drift detection, data stream, internet of things, machine learning

I. INTRODUCTION

Exploring insights from the Internet of Things (IoT) [1] is attracting widespread interest in academia and industry due to ubiquitous applications of IoT systems. An extensive network of IoT devices and sensors produce a large volume of data streams in real-time. To derive insights from a vast amount of collected IoT data, machine learning techniques are often used for classifying and predicting the data. For instance, IoT systems deployed in a smart industrial factory may generate a huge quantity of time series data from different sensors [2]. By constructing a machine learning model from the collected data and applying the model to evaluate the production system, one can predict the performance of the system to improve the productivity and capacity of the plant.

Data streams are typically stored in a non-persistent memory and processed on the fly, and hence analytic components such as machine learning models need to adapt to the distribution changes of data streams. Failures to adapt changes may cause an undesirable deterioration of the application performance, such as prediction accuracy. Considering a data classification task assisted by a supervised machine learning method, classification accuracy can degrade when facing the input data distribution change regarding the input features and the class labels. Such a problem is known as *concept drift* [3]. For real-world IoT applications, errors in label prediction may cause fault operations or unacceptable malfunctions. It is essential to detect a concept drift causing degraded predictions timely and accurately.

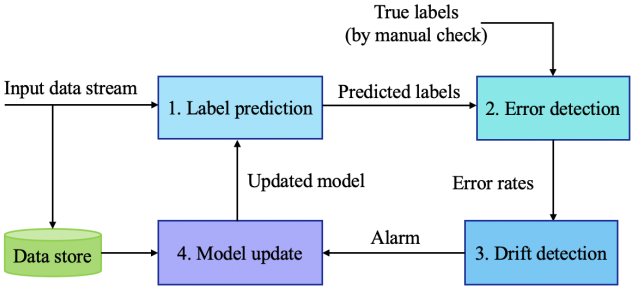

Fig. 1. A concept drift detection procedure

Figure 1 shows a common procedure of concept drift detection for a machine learning-based label prediction system. When a pre-trained machine learning model receives input data from IoT sensors, it predicts the label for the data (e.g., the name of the recognized object). Then, the predicted labels are compared with the true labels to count the prediction errors. In most cases, the true labels are implicitly assumed to be provided by domain experts. A drift detector continuously monitors the error rates and alarms when any symptoms of concept drift are observed. For instance, when input data distribution changes, one can observe the increased number of prediction errors. Once a concept drift is detected, it requires updating the machine learning model with the recently received data to adapt to the data distribution change. In the above detection procedure, the true labels are essential for knowing prediction error rates. However, it is not realistic to assume complete labels are always given in IoT application systems. Real data streams are unreliable in terms of labels of arrival data because the online labeling process may entail manual operations. Therefore, some data may be unlabeled or, even worse, be wrongly labeled. Although unreliable data streams may create significant impacts on the concept drift detection performance, such aspects have not been deeply explored yet.

In this paper, we experimentally investigate the robustness of the basic concept drift detection methods in terms of delay, precision, and efficiency by testing the detectors with unreliable data streams. We use the synthetic data stream *Circles* [4] for the experiments, which are applied in many data mining frameworks and regarded as a benchmark data stream for concept drift detecting tasks. We synthetically generate the unreliable version of *Circles* in which a part of labels in the data streams is transformed into error-labeled or missing-labeled data by specified ratios. For evaluating the performance of various drift detectors, we use the *Tornado* [5] framework to construct a Naïve Bayes-based classifier and detection models. The objective of our experiments is to evaluate the robustness of individual detectors instead of selecting the best detector in a specific performance measure. To this end, we choose six representative concept drift

detectors and compare the average detection delays and precisions by varying the ratios of unreliable data streams. The experimental results show that the detection delay is increasing, while the precision decreases in most cases as the data stream becomes unreliable. For error-labeled data cases, the impacts on the delay and the precision are significant for all the drift detectors. In an extreme case, where half of the data stream is mixed with error-labeled and missing-label data, all the detectors lose their functionality except the detector using Cumulative Sum (CUSUM). For missing-label data cases, while the precision of drift detection is not much affected, the detection delay increases when the ratio of missing-label data increases. Despite its simple drift detection scheme, we observe that CUSUM achieves a notably stable performance compared with other detectors in most of the unreliable scenarios considered.

The remaining part of the paper is organized as follows. Section II gives an overview of concept drift and introduces six drift detection methods. Section III discusses the robustness issue of existing concept drift detectors. Section IV provides the experiment configuration and results of the robustness evaluation using unreliable data streams. Finally, Section V gives our conclusion and future work.

II. CONCEPT DRIFT DETECTORS

A. Overview of concept drift

1) Definition of concept drift: Concept drift in machine learning and data mining refers to changes in the relationship between input data and output result over time in the underlying distribution. The formal definition of concept drift between two time points can be specified by

$$\exists X: P_{t_0}(X, y) \neq P_{t_1}(X, y), \tag{1}$$

where P_{t_0} denotes the joint distribution at time t_0 between the set of input variables X and the target variable y and P_{t_1} is the joint distribution at time t_1 $(> t_0)$. Changes in data can be characterized as variations in the components of this relation [6]. Moreover, based on a set of input features distribution $P(X)$ and posterior probabilities $P(y|X)$, concept drift is classified into two types [3], namely real concept drift and virtual drift. Real concept drift refers to changes in $P(y|X)$. Such a change can happen either with or without changes in $P(X)$. On the other hand, virtual drift occurs when the distribution of the incoming data changes (i.e., $P(X)$). In this paper, we consider real concept drift that may impact the prediction accuracy, and hence a timely detection is essential.

2) Diversity of concept drift: Considering how data distribution changes, a concept drift can be categorized into different types of drifts such as abrupt drift, gradual drift, incremental drift, and recurring drift [7]. If the original data distribution is suddenly substituted by another one and it affects the prediction accuracy, the drift is called as an abrupt drift. On the other hand, when the samples from the original data distribution decrease gradually and the samples of new data distribution start increasing, the drift can be called as a gradual drift. In this paper, we concentrate on the problem of gradual concept drift detection.

B. Categorization of detectors

As presented in the concept drift detection procedure (in Figure 1), a concept drift detector monitors the error rates of predictions and alarms when the drift is detected by decision rules. The basic drift detection methods can be categorized into three groups as follows [4].

1) Sequential analysis-based approaches validate the prediction results after classifying work and output an alarm of drift while meeting a pre-set threshold. The memoryless detector Cumulative Sum (CUSUM) [8] and its evolving version Page-Hinkley (PH) [8], including Shiryae's Bayesian test that depending on online thresholding are all belonged to this type.

2) Statistical Process Control-based methods normally monitor and control continuous training processes via statistical techniques such as standard deviation of the constantly outputting error rates and the distance between adjacent errors of predictive results to detect concept drift. The Drift Detection Method (DDM) [9] that is known as a benchmark in this categorization, Early Drift Detection Method (EDDM) [10] and Exponentially Weighted Moving Average (EWMA) [11] are members of this group.

3) Sliding-window-based detectors generally implement a fixed reference window to memorize the past information while a sliding detection window over the most recent samples. The prediction result distribution between two windows is compared by Hypothesis testing theory and with the null hypothesis indicating the distribution are equal, in turn, a drift is declared if the null hypothesis is rejected. The Adaptive Windowing (ADWIN) [12], SeqDrift detectors, Drift Detection Methods based on Hoeffding's Bound (HDDM) [13] are some representatives of this family.

C. Detectors for evaluation

In our experiments in Section IV, we employ two representative detectors from each group since they are basic detectors that performed well under a variety of extensive data stream tests in the previous literature. The basics and characteristics of these detectors are briefly explained below:

1) CUSUM: It is a classical change detection algorithm that gives an alarm when the mean value of the input data is significantly different from zero [8]. The CUSUM test is defined as follows:

$$S_t = max(0, S_{t-1} + (x_t - \delta)) \tag{2}$$

The decision rule: if $S_t > \lambda$ then it alarms and reset $S_t = 0$. Here, λ is a threshold, δ corresponds to the acceptable magnitude of changes, and x_t is the presently obtained value. This formula only detects changes in the positive direction. When negative changes need to be found as well, the min operation should be used instead of the max operation. In this case, a change is detected when the value of S_t becomes below the (negative) value of the threshold. The CUSUM test is memoryless, and its accuracy depends on determining values of parameters δ in formula (2) and λ.

2) PH: A variant of the CUSUM algorithm is the Page-Hinkley test [8], which is typically used for monitoring change detection in signal processing. It allows efficient detection of changes in the normal behavior of a process established by a model. This test considers a cumulative variable m_T, defined as the cumulated difference between the observed values and their mean value until the current moment:

$$m_T = \sum_{t=1}^{T} (x_t - \bar{x}_T - \delta), \qquad \bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t \tag{3}$$

Here δ and x_t are the same parameters as specified in CUSUM, while T is the current time. The minimum value of this variable is computed by:

$$M_T = min(m_{t,} t = 1 \dots T) \quad (4)$$

$$PH_T = m_T - M_T \quad (5)$$

After computing m_T in the range of 1 to T, the minimum value M_T of m_T is updated which is then used to compute the PH statistics by (5). When the value of PH_T becomes greater than a threshold λ, it indicates a drift alert.

3) DDM: This drift detection method employs Binomial distribution [9] that gives the general form of the probability for the random variable representing the number of errors in a sample of n examples.

$$s_i = \sqrt{p_i(1 - P_i)/\mathrm{i}} \quad (6)$$

For each point i in the sequence that is being sampled, the error rate is the probability of incorrectly classifying P_i with standard deviation given by formula (6). Next, store the values of p_i and s_i when $p_i + s_i$ reaches its minimum value during the process, in other words, obtaining p_{min}, s_{min}. Finally, the detection follows conditions below:

a) $p_i + s_i \geqslant p_{min} + 2 \cdot s_{min}$ for the warning level. Beyond this level, the examples are stored in anticipation of potential concept drift.

b) $p_i + s_i \geqslant p_{min} + 3 \cdot s_{min}$ for the drift level. Beyond this level, the concept drift is supposed to be true.

4) EDDM: This approach is developed for improving the detection in the presence of gradual concept drift [10]. The basic idea is to consider the distance between two classification errors instead of only the number of errors. We can calculate the average distance between two errors p_t and its standard deviation s_t. What we store are values of p_t and s_t till $p_t + 2 * s_t$ reaches its maximum value (obtaining p_{max} and s_{max}). This method is assigned with two thresholds:

a) $(p_t + 2 * s_t)/(p_{max} + 2 * s_{max}) < \alpha$ for the warning level. Beyond this level, the examples are stored in advance of a possible change of context.

b) $(p_t + 2 * s_t)/(p_{max} + 2 * s_{max}) < \beta$ for the drift level. Exceeding this level shows a concept drift occurring.

5) ADWIN: This detector holds two sub-windows W with memory to store the average of collecting results. It alarms a drift when observing the distinct change of averages and corresponding expected values [11]. Let n_0 and n_1 be the size of W_0 and W_1, respectively, where $W_0 * W_1 = W$. The difference in average values $\hat{\mu}$ of two windows is bounded by

$$\left|\hat{\mu}_{W_0} - \hat{\mu}_{W_1}\right| \geq \varepsilon \quad (7)$$

$$\varepsilon = \sqrt{\frac{1}{2m} ln \frac{4}{\delta'}} \quad (8)$$

where m is the harmonic mean of n_0 and n_1, $\delta' = \delta / n$, and δ is the confidence level. Once a drift is detected, the sliding window is removed to the end of the window until no notable changes are found.

6) HDDM-A: HDDMA-test and HDDMW-test are a pair of detectors proposed simultaneously [13]. HDDMA compares the moving averages to detect drift, while HDDMW uses weighted moving averages instead. Both approaches use Hoeffding's inequality to set an upper bound to the level of difference between the averages.

III. ROBUSTNESS OF CONCEPT DRIFT DETECTION

The basic concept drift detection methods introduced above assume that a class label for input data can be provided immediately after the prediction [3]. Nevertheless, such an assumption is unrealistic in some practical problems that need to deal with unreliable data streams. Instead of relying on a supervised approach that requires complete labels, some researchers focus on proposing efficient unsupervised or semi-supervised concept drift detectors with sliding window mechanisms [14] [15] [16]. According to [17], the existing unsupervised drift detection methods are classified into two major categories depending on how to construct the detection window, namely batch-based method or online-based method. Unsupervised learning methods in both categories evolve and provide efficient ways to detect concept drifts without relying on complete and true labels.

In contrast to these new approaches, our main question in this paper is how robust are the performances of basic drift detection methods under unreliable data streams. Due to the inclusion of a partially or fully manual labeling process, as shown in Figure 1, the data stream may contain incorrect labels and missing labels. Nonetheless, we may still want to use the basic drift detectors even when their drift detection performances degrade due to unreliable data streams. In other words, we advocate that the *robustness* of the drift detectors against unreliable data streams is another important property of the basic drift detection methods when considering real use cases. This aspect is particularly important in IoT application scenarios since the data analysis platform may not have sufficient resources to process advanced drift detection methods.

In our robustness evaluation, we consider two key performance measures of drift detectors as below.

1) Delay: The detection delay represents the number of samples in the data stream from the drift detection point to the true drift point. In other words, it is a measure to evaluate how fast a drift detection method can detect the drift point. We commonly set a fixed acceptable threshold of samples to determine whether the drift is actually occurring or not (i.e., decision deadline). The lower delay within the threshold indicates a better performance of concept drift.

2) Precision: Supposing the correct drift point is successfully detected within the acceptable interval, we regard it as a True Positive (TP). Furthermore, the False Positive (FP) rate represents the detection result considering incorrect drift point as true, while the worst situation False Negative (FN) rate indicates ignoring or missing the true drift points. Precision can be calculated by $TP/(TP + FP + FN)$.

These measures are essential to quantify how individual drift detectors can detect drifts swiftly and accurately. The performance can deteriorate when the reliability of labels decreases. If the performance is not significantly deteriorated, we can consider the drift detector is *robust* against unreliable data streams. To investigate the robustness of drift detectors, in the following section, we synthetically construct data streams with error labels and missing labels, and observe how the performances are affected by these factors.

IV. EXPERIMENT EVALUATIONS

Our experiments are conducted on *Tornado* [5] framework that provides synthetic data streams, classifiers, and drift detectors. Naïve Bayes classifier is employed as a base learner because it is one of the most efficient and practical classifiers. We created error-labeled data and missing-label data for *Circles* data streams. The ratios of unreliable data are chosen from 0 to 0.5. For a given ratio, unreliable data streams are generated randomly. We generate five groups of these unreliable data streams and five original data streams by different random seeds for experiments to obtain the average detection performance.

A. Unreliable data stream

Synthetic data stream *Circles* are frequently adopted in data mining studies. The data stream includes 100,000 instances with gradual drift. The data contains two attributes x and y which are uniformly distributed in [0,1]. The classification function is given by $(x - x_c)^2 + (y - y_c)^2 = \gamma_c^2$ where (x_c, y_c) is its center and γ is the radius. The instances inside the circle are classified as positive (P), while the instances outside the circle are classified as negative (N). A gradual drift happens when the distribution of input values progressively varies, resulting in classification function changes.

In *Circles*, the actual drift points are desirably determined as the location of 25,000, 50,000, 75,000 for gradual drift setting. In other words, there are three TP drift points in total. The acceptable length (i.e., delay) of concept drift is set to 5000. TABLE I shows the summary of the original *Circles.*

TABLE I. SUMMARY OF ORIGINAL CIRCLES

Data stream	Class	Attribute	Drift points	Acceptable interval	Drift type
Circles	2	2	25,000, 50,000, 75,000	5000	gradual

Our incomplete labels version of *Circles* is applied as input data streams. For unreliable *Circles,* following the ratio of error-labeled data (from 0 to 0.5), we exchange the label *n* with *p* to generate error-labeled data. For unlabeled scenarios, following the ratio of missing-label data (from 0 to 0.5), the labels are modified to an undefined class. For mixed label scenarios, both error-labeled data and missing-label data are generated with half of the given ratio. TABLE II summarizes the configurations of unreliable data streams.

TABLE II. SUMMARY OF UNRELIABLE CIRCLES

Data stream	Label type	Ratio	Class
Circles	Original	0	2
Circles	Error	0.1-0.5	2
Circles	Missing	0.1-0.5	3
Circles	Mixed	0.1-0.5	3

B. Configuration of detectors

We use Naïve Bayes as a classifier and employ different detectors to detect the drifts in the generated data stream. To set the relevant specific parameter values for drift detection, we refer to some default parameter values from the literature and adjust the values through the preliminary experiments to establish the base performance. The parameter values used for individual detectors are shown in TABLE III.

TABLE III. CONFIGURATION PARAMETERS OF DETECTORS

Detector	Configuration
CUSUM	min instance = 2600, δ = 0.0001, λ = 50
PH	min instance = 2600, δ = 0.0001, λ = 50
DDM	min instance = 1600
EDDM	min instance = 2600, min errors between instances = 35, warning level = 0.97, drift level = 0.92
ADWIN	δ = 0.002
HDDM-A	warning confidence = 0.002, drift confidence = 0.001

In the preliminary experiments, the configured detectors are tested with the original *Circles*. The drift point detection results are shown in Figure 2. Each rectangle represents the entire input data stream, where each data is processed from left to right. We set three drift points at 25%, 50%, and 75% of the data stream that are represented as the vertical dotted lines. The colored dots on each box show the detected drift points by individual detectors. As can be seen, for the original *Circles*, all detectors successfully detect three drift points after the true drift points with acceptable delays, although ADWIN has two additional false alarms.

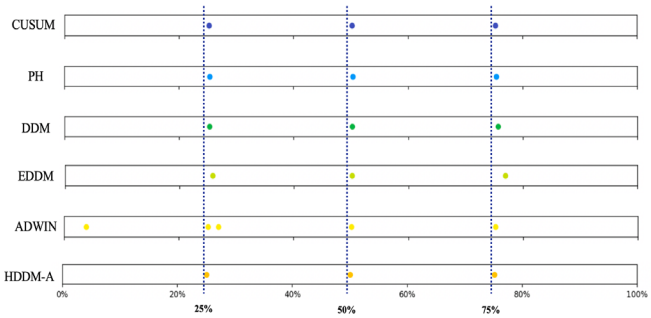

Fig. 2. Original circles drift point detection result

C. Robustness evaluation

To investigate the robustness of individual detectors against unreliable data streams, we observe the changes of detection delays and precision. We conduct five experiments for each data stream scenario and take the average values of the delays and the precisions. The results of the experiments are summarized in Figure 3 to Figure 8. For error-labeled data stream, Figure 3 and Figure 4 show the changes of the delay and the precision, respectively. As the ratio of error-labeled data increases, the delay of detectors generally increases. A lower delay indicates a swift detection of the concept drift. As can be seen in Figure 3, the delay of EDDM increases sharply compared to other detectors. When the error label ratio is larger than or equal to 0.3, CUSUM achieves the lowest delay among other detectors. On the other hand, the precision of detectors declines when the error label ratio becomes higher except ADWIN whose precision steadily increases until the ratio reaches from 0 to 0.3. The reason could be that the average difference of two sub-windows of ADWIN is easier to break the threshold in terms of a small range of error label rate. What stands out is that the precision of EDDM rapidly declines to 0. The precisions of PH and DDM are stable till the ratio 0.3, but then dramatically decrease as the ratio increases. The precision of CUSUM drops steadily over the domain but achieves the best precision among other detectors at the error label ratio 0.5.

Figure 5 and Figure 6 show the delay and precision values in missing-labeled data streams, respectively. We observe the fluctuating performance of EDDM both in the delay and the precision. EDDM detects drift depending on the distance

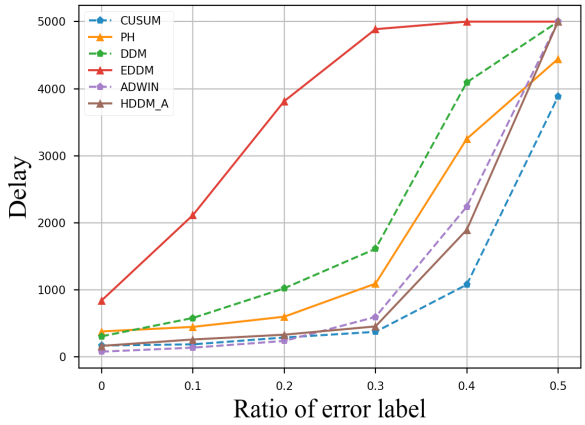

Fig. 3. The delay of drift detections for error-labeled data streams

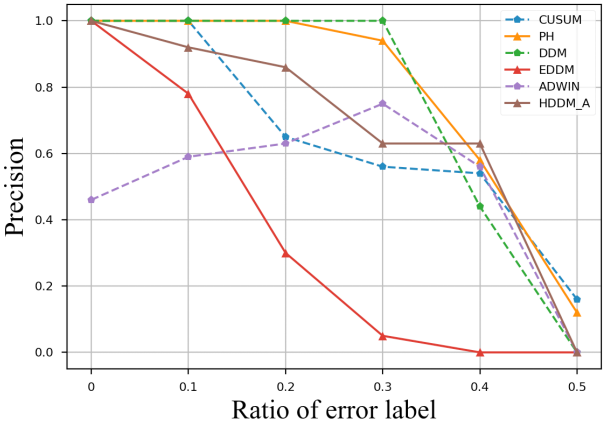

Fig. 4. The precision of drift detections for error-labeled data streams

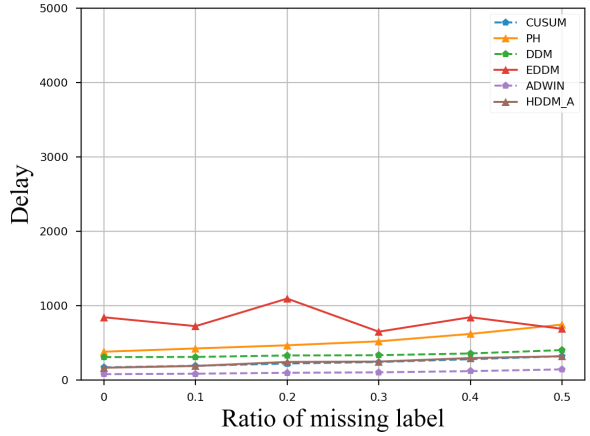

Fig. 5. The delay of drift detections for missing-label data streams

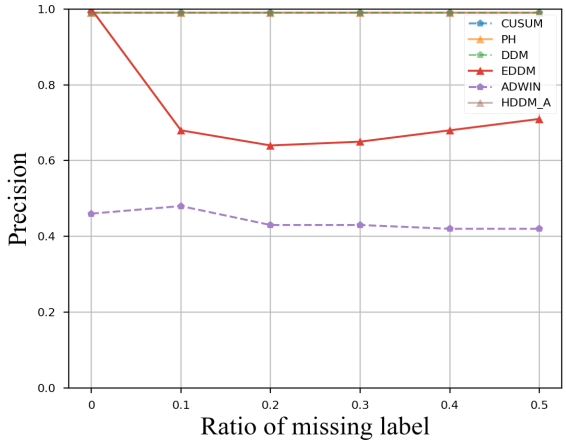

Fig. 6. The precision of drift detections for missing-label data streams

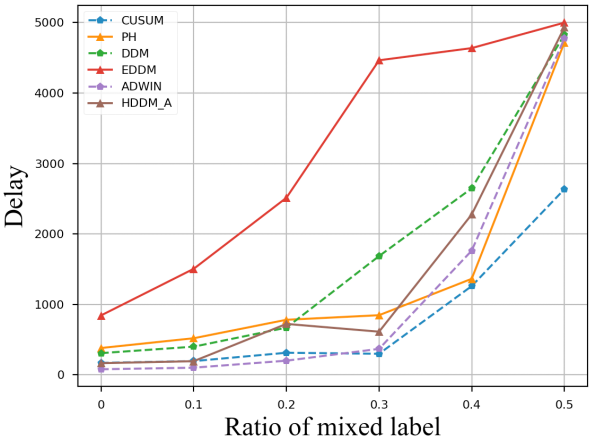

Fig. 7. The delay of drift detections for mixed label data streams

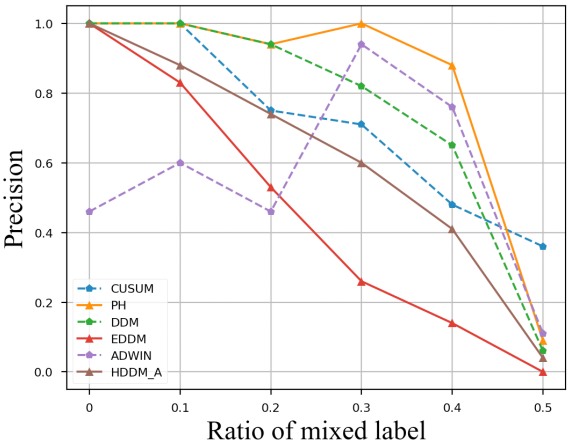

Fig. 8. The precision of drift detections for mixed label data streams

between adjacent errors, and thus, the random generation of unlabeled data might cause the high variance of such distance. Aside from EDDM, the delays of other detectors are gradually increasing by the increased ratio of missing-labeled data, while the precisions are not significantly changed. In particular, CUSUM, PH, DDM, and HDDM-A hold steady performances in both delay and precision, while ADWIN stays an unsatisfied precision performance in the domain.

Figure 7 and Figure 8 illustrate the results of mixed label data streams. For the results of the delay, the performance of EDDM rapidly reaches the maximum acceptable delay interval, which is clearly distinct from other detectors. We also notice that the delay performance oscillations of CUSUM, PH, ADWIN, and HDMM-A are reasonable until the ratio rises to 0.4. We observe that CUSUM outperforms other detectors in the delay when the ratio reaches 0.5. For the results of the precision, PH and DDM outperform the other detectors before the ratio exceeding 0.5. However, when the ratio reaches 0.5, the precisions of all detectors considerably deteriorate except CUSUM. As a result, despite a relatively simple scheme adopted, CUSUM is considered a preferable concept drift detector in terms of robustness against unreliable data streams.

D. Efficiency

From the results, we notice that error labels have a significant impact both on the delay and the precision, while missing labels have a relatively small impact. When the ratio of error-labeled data increases, the detection delay prolongs while the precision decreases. To consider this joint impact on

drift detection performance, we compute the *efficiency* of drift detectors by

$$e_i = p_i \cdot \left(\frac{D - d_i}{D}\right), \quad (9)$$

where p_i and d_i are the precision and the delay of detector i, respectively, and D is the acceptable length threshold (i.e., is equal to 5000 in our experiment). The efficiency values vary from 0 to 1, and a higher value corresponds to the higher efficiency of drift detection (i.e., more accurate detection with shorter delay). For mixed-labeled data set, the computed efficiencies are shown in TABLE IV.

TABLE IV. EFFICIENCY RESULT

	CUSUM	PH	DDM	EDDM	ADWIN	HDDM-A
Original	**0.97**	0.92	0.94	0.83	0.45	**0.97**
Mixed (0.1)	**0.96**	0.90	0.92	0.58	0.59	0.85
Mixed (0.2)	0.70	0.79	**0.81**	0.27	0.44	0.63
Mixed (0.3)	0.67	0.83	0.55	0.03	**0.87**	0.53
Mixed (0.4)	0.36	**0.64**	0.30	0.01	0.50	0.22
Mixed (0.5)	0.17	0.01	0.00	0.00	0.00	0.00

When the ratio of error-labeled and missing-labeled data is relatively small (<0.4), CUSUM, PH, and DDM achieve comparable efficiency. However, when the ratio exceeds 0.4, all the detectors are no more viable for practical use except CUSUM. In conclusion, from our experimental observations, CUSUM is considered as a relatively robust drift detector among the other detectors in terms of detection delay, precision, and efficiency.

V. CONCLUSION

In this paper, we evaluated the robustness of basic concept drift detectors by synthetically generated unreliable data streams that can be observed in real IoT data streams. The evaluation results show that error-labeled data stream has a more significant impact on the performance of detectors than missing-label data cases. We also find that CUSUM-based drift detection method can achieve relatively stable performance compared with other detectors under unreliable data streams. In the extreme mixed-label case at the ratio of 0.5, only CUSUM still can detect the drifts. Moreover, CUSUM is light and memoryless, which must be suitable for heavy detection tasks in IoT scenarios with limited computing memory environments. The findings from the experiments are limited to our configurations, and thus they may not be easily generalized for other types of data streams. Nevertheless, the observations from our experiments have several implications to research and practice for IoT applications dealing with unreliable data streams with concept drift. Our future work will explore different types of data streams for experiments and develop a robust concept drift detector that can cope with unreliable data streams.

REFERENCES

[1] K. Ashton, That "internet of things" thing, RFID Journal, Vol. 22, No. 7, pp. 97–114, 2009.

[2] D. Nallaperuma, D. De Silva, D. Alahakoon, and X. Yu, A cognitive data stream mining technique for context-aware IoT systems, In Proc. of the 43rd Annual Conference of the IEEE Industrial Electronics Society, pp. 4777–4782, 2017.

[3] J. Gama, A survey on concept drift adaptation, ACM computing surveys (CSUR), Vol. 46, No. 4, pp. 1–37, 2014.

[4] A. Pesaranghader, and H. Viktor, Fast hoeffding drift detection method for evolving data streams, Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 96–111, 2016.

[5] A. Pesaranghader, H. Viktor, and E. Paquet, Reservoir of diverse adaptive learners and stacking fast hoeffding drift detection methods for evolving data streams, Machine Learning, Vol. 107, No. 11, pp. 1711–1743, 2018.

[6] H. Wang, and Z. Abraham, Concept drift detection for streaming data, Inte'l Joint Conference on Neural Networks (IJCNN), pp. 1–9, 2015.

[7] D. Brzezinski, and J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, IEEE Transactions on Neural Networks and Learning Systems, Vol. 25, No. 1, pp. 81–94, 2013.

[8] E. Page, Continuous inspection schemes, Biometrika, pp. 100–115, 1954.

[9] J. Gama, Learning with drift detection, Brazilian symposium on artificial intelligence. Springer, Berlin, Heidelberg, pp. 286–295, 2004.

[10] G. Baena, Early drift detection method, Fourth international workshop on knowledge discovery from data streams, pp. 77–86, 2006.

[11] G.Ross, Exponentially weighted moving average charts for detecting concept drift, Pattern recognition letters, Vol. 33, No. 2, pp. 191–198, 2012.

[12] A. Bifet and R. Gavalda, Learning from time-changing data with adaptive windowing, In Proc. of SIAM international conference on data mining, Society for Industrial and Applied Mathematics, pp. 443–448, 2007.

[13] I. Frias-blanco, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Transactions on Knowledge and Data Engineering, Vol. 27, No. 3, pp. 810–823, 2014.

[14] S.Bashir, S.Petrovski, and D.DoolanBashir, A framework for unsupervised change detection in activity recognition, International Journal of Pervasive Computing and Communications, Vol. 13, No. 2, pp. 157–175, 2017.

[15] D. ReisFlach, P. Flach, S. Matwin, and G. Batista, Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test, In Proc. of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1545–1554, 2016.

[16] Y.Koh, Cd-tds: Change detection in transactional data streams for frequent pattern mining, International Joint Conference on Neural Networks (IJCNN), pp. 1554–1561, 2016.

[17] R. Gemaque, A. Costa, R. Giusti, and E. Dossantos, An overview of unsupervised drift detection methods, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, Vol. 10, No. 6, pp. 1381, 2020.

[18] R. Mohawesh, S.Tran, R. Ollington and S. Xu, Analysis of concept drift in fake reviews detection, Expert Systems with Applications, Vol. 169, 2021.

[19] S. Dongre, L. Malik and A. Thomas, Detecting concept drift using HEDDM in data stream, International Journal of Intelligent Engineering Informatics, Vol.7, No. 2, pp. 164–179, 2019.

[20] A. Ghani, N. Laila, I. Aziz and M. Mehat, Concept Drift Detection on Unlabeled Data Streams: A Systematic Literature Review, IEEE Conference on Big Data and Analytics, pp. 61–65, 2020.

[21] J. Perez, S. Roberto and G. Silas, Statistical Tests Ensemble Drift Detector, IEEE Symposium Series on Computational Intelligence, pp. 1021–1028, 2020.

[22] E. Baburoglu, A. Durmisoglu and T. Dereli, Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection, Expert Systems with Applications, Vol. 163, 2021.

[23] Y. Yuan, Z. Wang and W. Wang, Unsupervised concept drift detection based on multi-scale slide windows, Ad Hoc Networks, Vol. 111, 2021.

[24] O. Mahdi, E. Ali and J. Cao, Fast reaction to sudden concept drift in the absence of class labels, Applied Sciences, Vol. 10, No. 2, pp. 606, 2020.

[25] O. Mahdi, E. Ali and J. Cao, Fast reaction to sudden concept drift in the absence of class labels, Applied Sciences, Vol. 10, No. 2, pp. 606, 2020.

[26] H. Hu, M. Kantardzic and T. Sethi, No Free Lunch Theorem for concept drift detection in streaming data classification: A review, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, Vol. 10, No. 2, pp. e1327, 2020.

[27] O. Gozuacik and F. Can, Concept learning using one-class classifiers for implicit drift detection in evolving data streams, Artificial Intelligence Review, pp. 1–23, 2020.