Hindawi
Security and Communication Networks
Volume 2022, Article ID 5822124, 13 pages
https://doi.org/10.1155/2022/5822124

Research Article

LogCAD: An Efficient and Robust Model for Log-Based Conformal Anomaly Detection

Chunbo Liu,[1] **Mengmeng Liang**,[2] **Jingwen Hou**,[2] **Zhaojun Gu**,[1] **and Zhi Wang**[3]

[1]*Information Security Evaluation Center, Civil Aviation University of China, Tianjin 300300, China*
[2]*College of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China*
[3]*College of Cyber Science, Nankai University, Tianjin 300350, China*

Correspondence should be addressed to Zhi Wang; zwang@nankai.edu.cn

Received 29 October 2021; Accepted 23 February 2022; Published 30 March 2022

Academic Editor: Hao Peng

Log files are usually semistructured files that record the historical operation information of systems or devices. Researchers often find anomalies by analyzing logs, so as to identify system operation faults and cyberattacks. Traditional classification-based methods, especially deep learning methods, can effectively solve the problem of static log anomaly detection. However, when addressing dynamic unstable logs caused by concept drift and noise, the performance of those methods decreased significantly, and false positives are prone to occur. Retraining model is a choice to solve the log instability problem, but this will greatly increase the computational complexity for deep learning models. The log-based conformal anomaly detection (LogCAD) builds a confidence evaluation mechanism for multiple labels, which can achieve good detection results by making collaborative decisions based on multiple weak classifiers without deep learning. Moreover, LogCAD can be easily extended to dynamic unstable logs. It incrementally updates the trained model with conformal detection results of new samples. Experimental results show that LogCAD can achieve excellent detection results for both dynamic unstable logs and static stable logs. Compared with LogRobust and other deep learning models, it has higher efficiency and wider application scope.

1. Introduction

With the rise of large distributed systems and large supercomputers, these computers generate tens of thousands of logs every day. Logs record the status and running track of the systems. System administrators can use logs to track the running status of the system and discover faults [1–3]. Log-based anomaly detection [4–6] has become an important means for system administrators to maintain the normal operation of the systems. However, due to the large amount of distributed system logs, manual log analysis requires much time and effort. How to extract effective information from massive logs in real time and automatically has become a key research in the field of system maintenance. Moreover, as a result of concept drift or model aging [7, 8], it always occurs in an unstable environment. With the continuous operation of the systems, a variety of running programs will generate logs that are more complex and variable than ever before. But the current hyperplane or classifiers trained from historical data do not conform to the distribution law of new samples, so the classification accuracy will decrease. Therefore, the problem of log instability has a serious impact on the log-based anomaly detection algorithm. We analyze the causes of instability in real log data, mainly including the following aspects: (1) the log statement changes naturally due to the influence of software update and external environment, namely, concept drift, and (2) noise introduced in the process of log data collection and parsing. In addition, although many algorithms, such as Decision Tree (DT [9]), Naïve Bayes (NB [10]), Support Vector Machine (SVM [11]), Logistic Regression (LR [12]), Random Forest (RF [13]), etc., achieve good performance in the actual environment, they still face many challenges in dealing with unstable logs. They often use periodic retraining to deal with unstable problems, which requires manual labeling of all processed samples and is limited by available resources. On the other hand, these

algorithms stay at the coarse-grained level and ignore fine-grained differences between data, such as data differences between the same class and data differences between different classes. In addition, the concepts of models, hyperplanes, and classifiers trained by logs can not change with the distribution of log data and lack the mechanism of dynamic updating previous experience. LogRobust [14] addresses the problem of log instability and solves it by semantic vectorization and attention-based classification. Yet, it would be less efficient in computing and less effective for some data sets (e.g., BGL, discussed more in Section 4.3.1).

In order to enhance efficiency and performance of algorithm while maintaining its robustness, we proposed a dynamic log anomaly detection model with confidence level, called LogCAD (log-based conformal anomaly detection). To avoid the error of a single decision, it adopts multiple algorithms for joint detection. First, the nonconformity measure module is proposed to calculate the statistical p value. Second, we construct a prediction set with confidence level and mark a confidence label by dynamically adjusting confidence level. Third, the results including labels, nonconformity scores, and the confidence of the logs to be detected are fed back to the score set containing the corresponding labels, which is used as the previous experience to calculate the p value in the subsequent detection. Finally, according to the confidence degree and the preset confidence level obtained by collaborative detection, the unstable log is judged whether to be abnormal. Since LogRobust is superior to traditional algorithms in terms of robustness, we compare it with LogCAD in detail as shown in Table 1. Both algorithms are robust, while LogCAD realizes incremental updating without retraining. Moreover, LogCAD can gain excellent accuracy without using deep learning and operate in a natural online manner.

The main contributions of this work are summarized as follows:

(i) We propose LogCAD, a noval robust anomaly detection model, which can be applied to unstable logs with outstanding efficiency and performance

(ii) We design an incremental update mechanism to ensure LogCAD suitable for new unseen log events without retraining. LogCAD can be naturally generalized to the online setting

The rest of this paper is organized as follows: Section 2 introduces the motivation of our work and limitations of existing works. Section 3 depicts the framework and details of the LogCAD model. Section 4 presents the experimental design and results. Section 5 concludes this paper.

2. Preliminaries

2.1. Motivation. The sources of unstable data are mainly noise in log collection and the evolution of log syntax structure. Confronting these unstable data, the traditional algorithms show a sharp decline in the accuracy of anomaly detection, which is extremely detrimental. Therefore, it is urgent to improve the accuracy of anomaly detection through the algorithm that can adapt to the unstable environment.

2.1.1. Noise in Log Collection. The log-based anomaly detection mainly includes the following steps: log collection, log parsing, feature extraction, and anomaly detection. In the above operations, noise is inevitably introduced, mainly for the following reasons: First, human or computer errors occur during log collection. Second, the log parser itself is not accurate [15]. For example, He et al. [16] and Zhu et al. [17] find that existing log parsers are not accurate enough. These log parsers have different parsing effects on different data sets. For example, the overall accuracy of LogSig [18] on HDFS and Proxifier data sets is 70% to 80% higher than that on BGL and Linux log data sets. There are two causes. On the one hand, BGL and Linux involve more log event types, each of which has a longer length than the other data sets, making log parsing more difficult. On the other hand, because LogSig is not good at exploiting the uniqueness of log data; it adopts an aggressive clustering strategy, which groups two clusters if any two log messages are less than a specified threshold. For example, BGL contains a large number of log messages that are "generating core.∗". Although the similarity of "generating Core.2275" and "generating Core.852" is very intuitive, LogSig tends to separate these log messages into different clusters, which results in low log parsing accuracy. More importantly, the anomaly detection classifier is very sensitive to some critical events, but when these critical events are incorrectly resolved by the log parser, it will have a serious impact on the log anomaly detection [16]. Figure 1 shows a case where errors occurred during the parsing of Microsoft online services system logs [14]. Case 1 represents the missing keyword of the original log event, and Case 2 represents the keyword that incorrectly identifies the parameter as a log event.

2.1.2. Evolution of Log Syntax Structure. As software upgrades and the external environment changes, some of the old log events have been removed from the source code. In the new version, the original log events have been modified. Figure 2 shows the changes to the Micrsoft online services system log events [14]. Case 1 indicates that a new word is added to the original log event to make the log event statement more complete. Case 2 indicates that a new keyword is added to the original log event.

2.2. Limitations of Existing Works. In a dynamically unstable environment, it is not advisable to rely only on previous experience to predict what will happen later. As the system generates more and more complex and variable logs, the model will fail to adapt to the new data distribution, resulting in a sharp decline in prediction accuracy. In this environment, a model that can adapt to environmental changes is urgently needed. However, some log-based anomaly detection algorithms (SVM, LR, DT, etc.) are often made based on whether the new log is normal or abnormal, regardless of how the algorithm determines the choice. For example, hyperplane-based learning models (e.g., SVM) only check the side of the hyperplane where the object lies while ignoring its distance from the hyperplane. In addition, the statistical evaluation tells us how the new log that has been committed belongs to a class compared to the old log in

Table 1: Comparison between LogRobust and LogCAD.

Method	Learning model	Incremental updating	Robustness	Online
LogRobust	Deep learner (Bi-LSTM with attention)	Semi-retraining by dynamically updating parameters	Semantic vectorization	No
LogCAD	Shallow learner (LR, NB, SVM, etc.) + ensembler + conformal predictor	Updating nonconformity score sets without retraining	Dynamically updated nonconformity score sets	Yes

the same class, and whether the new log is a normal or abnormal event compared to other logs in the same event. They ignore the information brought by new logs.

Different from the above algorithm, Conformal Predictor (CP) theory [19] based on statistical evaluation uses past experience to predict the test sample with an additional level of confidence. Therefore, this method makes an association between the new log and the old log. When unstable logs occur, the experience with the new log can be updated to the previous experience without retraining the model, better mitigating the impact of unstable log problems. Conformal prediction [20] provides a statistical p value that could be used to calculate confidence and guide the algorithm to make decisions or evaluations. Adding the new learned experience into the algorithm decision will effectively mitigate the impact of unstable logs.

3. Model Framework

As shown in Figure 3, LogCAD mainly includes three major parts: data preprocessing, conformal prediction, and anomaly detection. Firstly, the original log data is preprocessed, the main purpose is to form the feature vector matrix A of unstructured log through log parser and feature extraction. Secondly, with the guidance of confidence $(1 - \varepsilon)$, we select multiple ensemble algorithms to form a nonconformity measure module, and the conformal score α is obtained. The ensemble algorithms used here are AdaBoost and RF. Finally, we adjust the significance level ε, mark the detected log sequence with a proper label, and attach a confidence degree and then obtain the prediction set. Through voting, the final prediction result and confidence level of the detected log will be obtained. The algorithm automatically selects the nonconformal score of the prediction error sample and adds it to the nonconformal score set in the second step as the experience of the subsequent unstable log sequence detection.

3.1. Preprocessing. The detailed processing of log data is shown in Figure 4.

3.1.1. Log Collection. Modern large-scale systems use logs to record system performance. Each log contains unstructured data such as time stamps, log priorities, system components, and log sequences. Typically, log messages record specific system events with a set of fields. Eight lines of the log were extracted from the log on the Amazon EC2 platform (see Figure 4), with some fields omitted for ease of presentation.

3.1.2. Log Parsing. Logs are unstructured and contain free text. The purpose of log parsing is to extract a set of event templates that can structure the raw log. Each log message is parsed into an event template (constant part) and some specific parameters (variable part), as shown in Figure 4 (i.e., "PacketResponder ∗ for blocking ∗ terminating" in the example). To assess the effectiveness of the automated log parser, we use the parsing accuracy [21] to denote the percentage of successfully parsed log events in the total log messages. After parsing, each log message has an event template this event template corresponds to a set number of messages of the same template. If and only if the log message event templates have ground truth of the corresponding log messages at the same time, the log message is considered resolved properly. Although the log parser Drain, which we adopt, achieves the highest parsing accuracy among the 13 common log parsers [17], it still introduces varying levels of noise. The LogCAD model proposed in this paper can alleviate the log instability caused by the inaccurate parser. See Section 3.3 for details.

3.1.3. Feature Extraction. The main purpose of this step is to extract valuable features from log events that can be fed into the anomaly detection model. The input of feature extraction is the log events generated during the log parsing step, and the output is the event count matrix. In each log sequence, the number of occurrences of each log event is counted into the log parser to form an event count vector. For example, if the event count vector is [0, 2, 0, 7, 6, 2], it means that, in this log sequence, event 2 occurred twice and event 4 occurred seven times. Finally, all event counting vectors are constructed to form event counting matrix A, where the sequence A_{ij} records the occurrence times of event j in the ith log sequence.

3.2. Conformal Prediction. In this section, a brief description of conformal prediction is given. The conformal prediction framework theory is based on the concepts of algorithm randomness and transformational reasoning. Generally speaking, the training set is defined as follows:

$$\{(x_1, y_1), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_n)\} \in X \times Y, \tag{1}$$

$$(x_1, y_1), \ldots, (x_{n-1}, y_{n-1}), (x_n, y_c), \tag{2}$$

where X represents the set of all possible objects, Y represents the set of all possible labels, and each sample (x_i, y_i) consists of an instance (usually an attribute vector) and its label. To predict the label of a new instance, the idea of conformal prediction is to try every possible label as a label

Case 1:

HttpRequestAsync::EnsureBuffer − allocated:* **resize**:*

HttpRequestAsync::EnsureBuffer − allocated:*

Case 2:

Skipping update of with app alias[*]

Skipping update of with app alias[**rtc**]

Figure 1: Misparsing logs in Microsoft online server X [14].

Case 1:

Creating and opening new * channel factory

Creating and opening new * channel factory **for bindingId** *

Case 2:

{ “result”:*” BaseHash” :*” UploadHash” :*” TipHash” :*” NewTipHash” :*}

MiniMerge: {“result”:*”BaseHash”:*”UploadHash”:*” TipHash”:*”NewTipHash”:*}

Figure 2: Evolving log events in the Microsoft online services system [14].

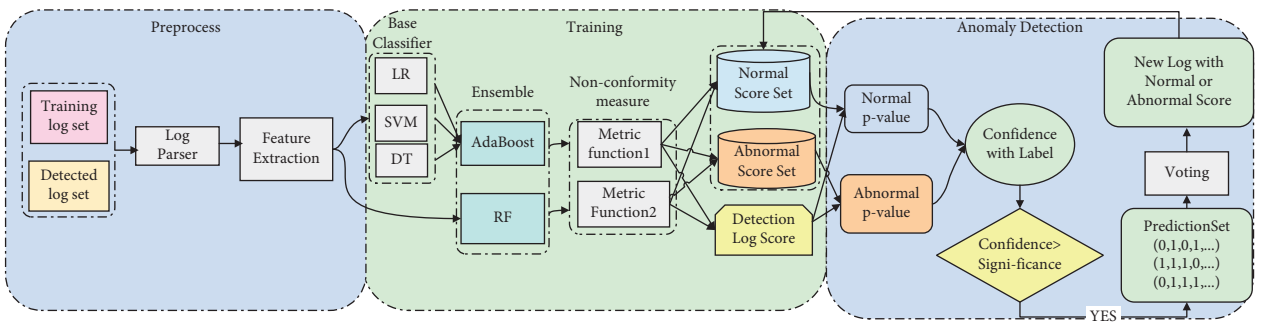

Figure 3: Composition of the LogCAD model.

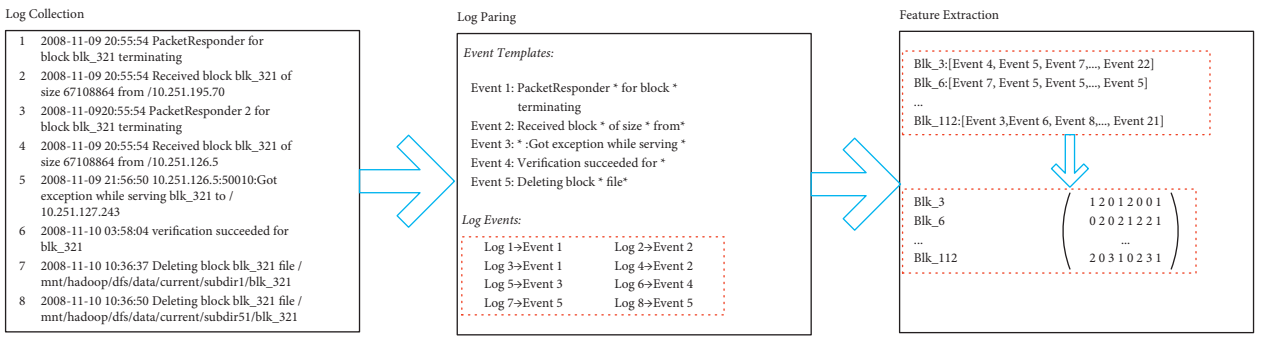

Figure 4: Log data preprocessing.

candidate, and the extended sequence, equation (2), conforms to the randomness hypothesis.

3.2.1. Nonconformity Measure. The design of singular value nonconformity measure is an important step in the conformal predictor algorithm, which is mainly used to measure the degree of nonconformity between a sample and a sample sequence distribution. For unstable logs, the nonconformity measure module can make use of past experience to determine the precise confidence level of the new prediction and consider the relationship between the past data and the detection data. We first choose multiple ensemble algorithms as the measure of nonconformity scoring function to calculate nonconformity scores. The conformal prediction result after ensemble learning is the weighted average of multiple classifiers, which makes the nonconformity score ranking generated by the nonconformity measure function more accurate. The scoring function is denoted as follows:

$$\alpha_i = A_n(\langle(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), (x_n, y_n)\rangle, (x_i, y_i)), i = 1, \ldots, n-1 \quad (3)$$

$$\alpha_i = A_n(\langle(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\rangle, (x_n, y_n)). \quad (4)$$

Here, it means that the data in <·> has nothing to do with the sequence. α_i is the nonconformity score of the i_{th} sample obtained according to (1). The larger it is, the more likely it is to belong to the sample sequence.

We use equation (4) to calculate the nonconformity score for all logs. For the log training data set, each algorithm will get two nonconformity score sets: normal score set and abnormal score set. For log detection sequences, we will also get a log detection score for each ensemble algorithm. When the label $y_c \in y$ is assigned to a new instance x_n, according to equation (5), the randomness degree related to $\{\alpha_1, \alpha_2, \ldots, \alpha_{n-1}, \alpha_n\}$ and (x_n, y_c) of the nonconformal score of a sequence can be calculated, namely, the p value. Here, the value of y_c can be divided into two cases: normal or abnormal. Therefore, for the log sequence to be tested, we get two p values, namely, normal p value and abnormal p value.

$$\text{p value} = \frac{\left|\{i = 1, 2, \ldots., n: \alpha_i \geq \alpha_n\}\right|}{n}. \quad (5)$$

In order to reduce the classification errors of algorithms, we select multiple ensemble algorithms to compose this module. Firstly, we classify training log sequences into normal and abnormal sequences based on their actual labels. Secondly, by ensemble learning nonconformity measure function, we get normal score set and abnormal score set. Finally, by calculating the normal p values, abnormal p values, and given significance levels, we predict the labels of the test log sequence. We establish a new association between the test log sequence that predicted the error and the previous log sequence and add the corresponding score to the normal score set or abnormal score set as the existing experience of subsequent detection, which will avoid the wrong decision caused by the algorithm.

3.2.2. Confidence and Credibility of Point Prediction. The conformal predicting framework provides two key indicators: confidence and credibility. As mentioned above, consider the training set D and test object x_n, try each possible label $y_c \in y$ (normal or abnormal) as the x_n' label, and then calculate the p values. Finally, the corresponding label with the maximum p value is selected as the detection object x_n.

The labels can be measured by confidence and credibility. Confidence is defined as 1 minus the second largest p value, while credibility is defined as the largest p value. Intuitively, high confidence indicates that all other candidates for the detected label are impossible. Low credibility means that the maximum p value is so small that the test instances do not follow the distribution generated by the training set. Note that if the data set is independent and identically distributed, the credibility will not be low.

The four possible outcomes are as follows: (1) High Credibility – High Confidence: this is the best case where the algorithm correctly identifies one sample for one class and only one class. (2) High Credibility – Low Confidence: the test samples are very similar and belong to two or more classes. (3) Low Credibility – High Confidence: the algorithm cannot accurately associate the test sample with any of the existing classes of the data set. (4) Low Credibility – Low Confidence: the algorithm detects a label for the test sample, but it seems to be more similar to another label [7].

3.3. Anomaly Detection. In this section, we train AdaBoost ensemble classifier (AB in short) and Random Forest model with training set $\{(x_1, y_1), \ldots, (x_{n-1}, y_{n-1})\}$. Calculate the nonconformal scores $\alpha_1, \alpha_2 \ldots, \alpha_{n-1}$ of the normal samples and abnormal samples $\{(x_1, y_1), \ldots, (x_l, y_l)\}$ in the training set, respectively. Note: $\alpha_i, i = 1, .., n-1$ is the result of the nonconformity measure function of the classifier after ensemble learning. Then, we design a confidence-guided dynamic adjustment parameters: significance level ε. For each algorithm, we obtain a label with confidence for each log detecting entry. Then, multiple labels given by each algorithm form a prediction set after filtering by the significance level. We can detect anomalies through many rules, such as voting rule, etc. By voting, multialgorithm collaborative decision-making will reduce the misjudgment of a single algorithm and improve the accuracy of the model and the ability to adapt the dynamic change of data distribution. Finally, we use the p value to calculate the credibility and the confidence. The pseudocode

of the above process, namely, LogCAD (log-based conformal anomaly detection), is described as Algorithm 1.

We obtain a prediction set that consists of multiple labels with confidence and can use the voting rule to set a confidence-guided label for the test log entry. Compared with the unimodel algorithms, LogCAD has stronger robustness in dealing with unstable logs. Finally, we set up feedback mechanisms including labels, nonconformal scores, and confidence to mitigate the effects of unstable logs. By means of p values that measure the nonconformity among log entries, the feedback mechanisms establish a relationship between the new log and previous ones. Thus, LogCAD can effectively alleviate the impact of unstable log problems.

4. Experiments and Analysis

4.1. Experimental Design

4.1.1. Data Sets

(1) HDFS Data Set. HDFS data set is a common log-based anomaly detection benchmark [22, 23]. It contains 11,175,629 log messages collected from Amazon EC2. HDFS logs record unique block IDs for each block operation, such as allocate, write, copy, and delete. Each unique block ID can be used to split the log into a set of log sequences, extract feature vectors from the log sequences, and generate 575,061 event count vectors, including 16,838 anomaly samples. We randomly collect 6000 normal log sequences and 6000 abnormal log sequences from the original HDFS data set as a training set.

(2) BGL_100K Data Set. The BGL data contains 100,000 log messages recorded by the BlueGene/L supercomputer system at Lawrence Livermore National Laboratory (LLNL) [24]. BGL logs do not record the identifier of each job executed. Therefore, we must slice the log into a log sequence using either a fixed window or a sliding window and extract the corresponding event count vector. In the BGL_100K data, 2613 log messages are marked as failures, and if there are any failure logs in a log sequence, the log sequence is marked as an anomaly.

(3) Unstable Log Sequences. One of the manifestations of unstable logging is that log events in the log sequence may change during log collection. To simulate unstable log sequences, some log events which do not affect the corresponding abnormal status label are randomly removed from the original log sequence. In a log sequence, a randomly selected unimportant log event is repeated many times, and these composite unstable log sequences are proportionally injected into the original log data.

(4) Synthetic Log Data Set. To demonstrate the effectiveness of our approach to unstable log data, we create unstable test data sets based on the original HDFS and the BGL_100K data set. We primarily simulate a type of log instability, as shown in Figure 5. Based on the experience gained in empirical studies by Zhang et al. [14], unstable log data are synthesized. We believe that composite log data can reflect the unstable nature of real logs.

To synthesize the unstable data set, 51,000 log sequences are randomly collected from the original HDFS data set, including 50,000 normal and 1,000 abnormal logs. The anomaly percentage is 2%, which is close to the anomaly percentage of the original HDFS data set. 100,000 log sequences are randomly collected from the original BGL data set, including 97387 normal and 2613 abnormal, with an anomaly percentage of about 2%. Unstable log sequences are injected into the data set, and a test set is created. Tables 2 and 3 summarize the synthetic HDFS and BGL_100K data sets, respectively.

The experiments are performed on the following platform: Intel(R) Core (TM) i7-6700 CPU @3.40 GHz 3.41 GHz, 16.0 GB RAM, Windows operating system.

4.1.2. Evaluation Metrics.

Accuracy, Precision, Recall, and harmonic mean (F1) values are used as secondary evaluation metrics based on confusion matrix. These metrics are used to evaluate the validity of anomaly detection.

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + FP + FN + TN}, \\ \text{Pr}ecision &= \frac{TP}{TP + FP}, \\ \text{Re}call &= \frac{TP}{TP + FN}, \\ F1 &= \frac{2 * \text{Pr}ecision * \text{Re}call}{\text{Pr}ecision + \text{Re}call}. \end{aligned} \tag{6}$$

As shown above, Precision represents the correct percentage of reported anomalies, Recall denotes the percentage of true anomalies detected, and F1 is the harmonic average of Precision and Recall.

To jointly evaluate the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) in the confusion matrix, Matthews Correlation Coefficient (MCC) is adopted.

$$MCC = \frac{TP * TN + FP * FN}{\sqrt{(TP + FP)(TP + FN)} * \sqrt{(TN + FP)(TN + FN)}}. \tag{7}$$

4.2. Determination of the Best Confidence Level.

In this section, we explore the performance of each metric of LogCAD at different confidence levels (as shown in Figures 6 and 7), so as to determine the optimal confidence level to continue the following experiments. In Figure 6(a), F1, MCC, etc. increase slowly with the increase of confidence level and reach the maximum value between 0.995 and 1. Figure 6(b) describes in detail the change process of F1, MCC, and other indicators in HDFS data set when the confidence level is between 0.995 and 1. It can be observed that when the confidence level is 0.997, the values of F1, MCC, and other metrics reach the maximum. Therefore, we set the confidence level to 0.997 for the HDFS log data set.

Input: Detected log set X, Training log set T, Training normal log set T_1, Training abnormal log set T_2, Significance level ε
Output: Prediction set Γ^{ε}

```
Γε⟵∅
BaseClassifier ⟵ {LR, SVM, DT, NB}
  for all i ϵ BaseClassifier do
    Train ensemble classifier with training log data T
    Normal scores set S_N ⟵Ensemble classifier' non-conformity measure(T1)
    Abnormal score set SA ⟵ Ensemble classifier' non-conformity measure(T2)
    for all log entry x ϵ X do
      p_N(x)⟵normal p value calculated in SN
      pA(x)⟵abnormal p value calculated in S_A
      if p_N(x) ≥ p_A(x) then
        Cred(x) ⟵ pN(x)
        Conf(x) ⟵1 − pA(x)
        if Conf(x) ≥ ε then
          result(x) ⟵(NORMAL, Conf(x))//detection results are pairs of (class label, confidence)
        else
        result(x) ⟵(ABNORMAL, Conf(x))
      else
        Cred(x) ⟵ pA(x)
        Conf(x) ⟵1 − pN(x)
        if Conf(x) ≥ ε then
          result(x) ⟵(ABNORMAL, Conf(x))
        else
          result(x) ⟵(NORMAL, Conf(x))
        Γε⟵ΓεU{result(x)}
        end
end
```

Algorithm 1: Proposed framework of LogCAD.

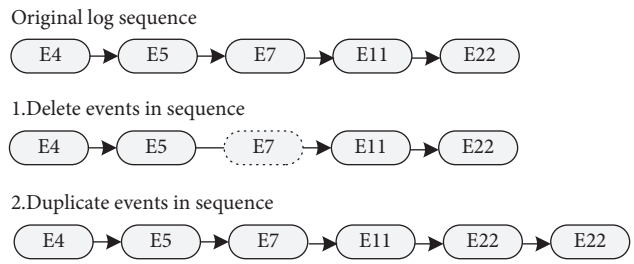

Figure 5: Synthetic log sequence.

Table 2: The synthetic HDFS data set.

Data set	Stable/unstable	#normal samples	#abnormal samples	#total samples
Training	Stable	6000	6000	12000
Testing	Unstable	50000	1000	51000

Table 3: The synthetic BGL_100K data set.

Data set	Stable/unstable	#normal samples	#abnormal samples	#total samples
Training	Stable	58432	1568	60000
Testing	Unstable	38955	1045	40000

In Figure 7(a), when the confidence level is between 0.5 and 0.9, the values of F1 and MCC increase with increasing confidence level, and when the confidence level is between 0.85 and 1, the values of F1 and MCC reach the maximum. Figure 7(b) describes in detail the change process of F1, MCC, and other metrics on the BGL_100K data set when the

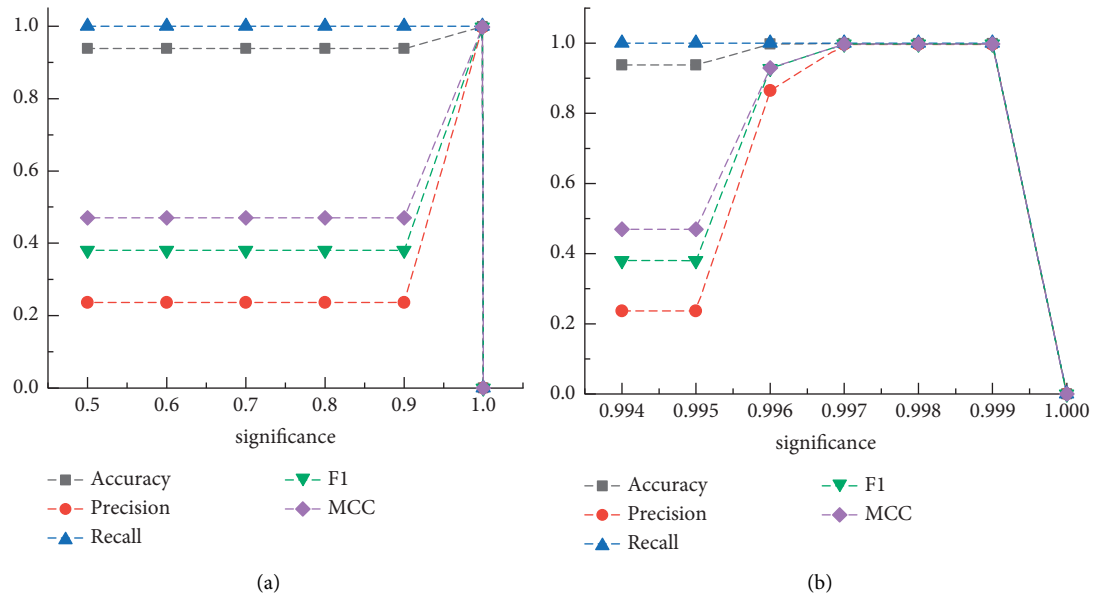

Figure 6: LogCAD's performance on HDFS data set varying with confidence level. (a) Overall changes in metrics. (b) Local changes in metrics.

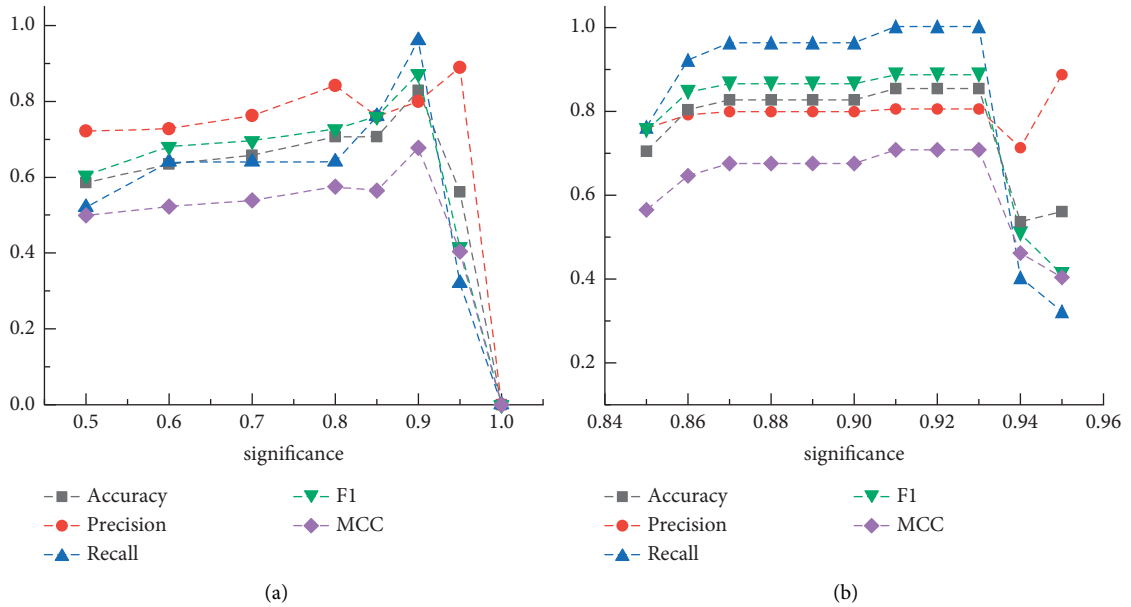

Figure 7: LogCAD's performance on BGL_100K data set varying with confidence level. (a) Over changes in metrics. (b) Local changes in metrics.

confidence level is between 0.9 and 1. It can be observed that when the confidence level is 0.92, the values of F1, MCC, and other indicators reach the maximum. Therefore, on the BGL_100K data set, the confidence level is set to 0.92.

4.3. Synthetic Log Data Set Experiments

4.3.1. Effectiveness of LogCAD. We train LogCAD on the original HDFS and BGL_100K and then test the trained model on the synthetic test set, respectively. We take the injection rate to denote the proportion of unstable log sequences injected into original data sets and set it to 5%, 10%, 15%, and 20%, respectively. We compare LogCAD with three traditional unique decision classifiers (LR, SVM, and NB), CP, and LogRobust proposed by Zhang et al. [14]. The experimental results are shown in Tables 4 and 5. There are three underlying algorithms of CP, which are corresponding LR, SVM, and NB. We only take the one that achieves the best performance, and the more details will be illustrated in Section 4.3.2. As shown in Tables 4 and 5, LogCAD performs the best. LogCAD can maintain relatively high accuracy even

Table 4: Experimental results on HDFS unstable log sequences.

Injection rate (%)	Classifier	Accuracy	Precision	Recall	F1	MCC
5	LR	0.999	0.94	0.997	0.967	0.967
	SVM	0.996	0.849	0.994	0.916	0.917
	NB	0.996	0.849	0.998	0.918	0.919
	LogRobust	0.996	0.990	0.930	0.960	0.965
	CP	0.998	0.946	1	0.972	0.972
	LogCAD	**1**	**0.992**	**1**	**0.996**	**0.996**
10	LR	0.998	0.94	0.984	0.961	0.961
	SVM	0.996	0.848	0.989	0.913	0.914
	NB	0.996	0.849	0.993	0.915	0.916
	LogRobust	0.997	0.940	0.990	0.961	0.964
	CP	0.998	0.946	1	0.972	0.972
	LogCAD	**0.999**	**0.996**	**0.992**	**0.994**	**0.994**
15	LR	0.998	0.94	0.984	0.961	0.961
	SVM	0.996	0.848	0.989	0.913	0.914
	NB	0.996	0.849	**0.999**	**0.996**	**0.992**
	LogRobust	0.998	0.980	0.910	0.940	0.960
	CP	0.998	0.945	0.992	0.968	0.968
	LogCAD	**1**	**0.992**	**0.992**	**0.992**	**0.992**
20	LR	0.998	0.939	0.983	0.960	0.960
	SVM	0.993	0.846	0.987	0.911	0.909
	NB	0.995	0.854	0.979	0.912	0.910
	LogRobust	0.998	0.920	0.970	0.950	0.960
	CP	0.998	0.946	0.996	0.97	0.97
	LogCAD	**0.999**	**0.975**	**0.996**	**0.985**	**0.985**

Table 5: Experimental results on BGL_100K unstable log sequences.

Injection rate (%)	Classifier	Accuracy	Precision	Recall	F1	MCC
5	LR	0.463	1	0.12	0.214	0.25
	SVM	0.667	0.833	0.156	0.263	0.32
	NB	0.756	0.857	0.72	0.783	0.622
	LogRobust	0.488	0.833	0.2	0.323	0.328
	CP	0.756	0.857	0.72	0.783	0.622
	LogCAD	**0.854**	0.806	**1**	**0.893**	**0.71**
10	LR	0.631	0.571	0.125	0.205	0.296
	SVM	0.667	0.833	0.156	0.263	0.32
	NB	0.655	0.714	0.156	0.758	0.677
	LogRobust	0.488	0.833	0.2	0.323	0.328
	CP	0.683	0.8	0.64	0.711	0.556
	LogCAD	**0.854**	0.806	**1**	**0.893**	**0.71**
15	LR	0.631	0.571	0.125	0.205	0.296
	SVM	0.667	0.833	0.156	0.263	0.32
	NB	0.655	0.714	0.156	0.758	0.677
	LogRobust	0.463	1	0.12	0.214	0.225
	CP	0.756	0.857	0.72	0.783	0.622
	LogCAD	**0.829**	**0.909**	**0.8**	**0.851**	**0.709**
20	LR	0.631	0.571	0.125	0.205	0.296
	SVM	0.655	0.714	0.156	0.256	0.321
	NB	0.69	0.568	0.781	0.658	0.56
	LogRobust	0.414	1	0.04	0.077	0.126
	CP	0.683	0.75	0.72	0.735	0.55
	LogCAD	**0.756**	**0.8**	**0.8**	**0.8**	**0.613**

when the instability injection rate is 20%. For example, LogCAD remains robust when the injection rate is 20%; i.e., 20% of the original log sequence is replaced by the synthetic log sequence. There are two reasons. First, on HDFS log data set, LogCAD is a multimodel collaborative detection with confidence level of 0.997, which means the probability of prediction error is less than 0.3%. On the BGL_100K data set, LogCAD performs detection at a confidence level of 0.92.

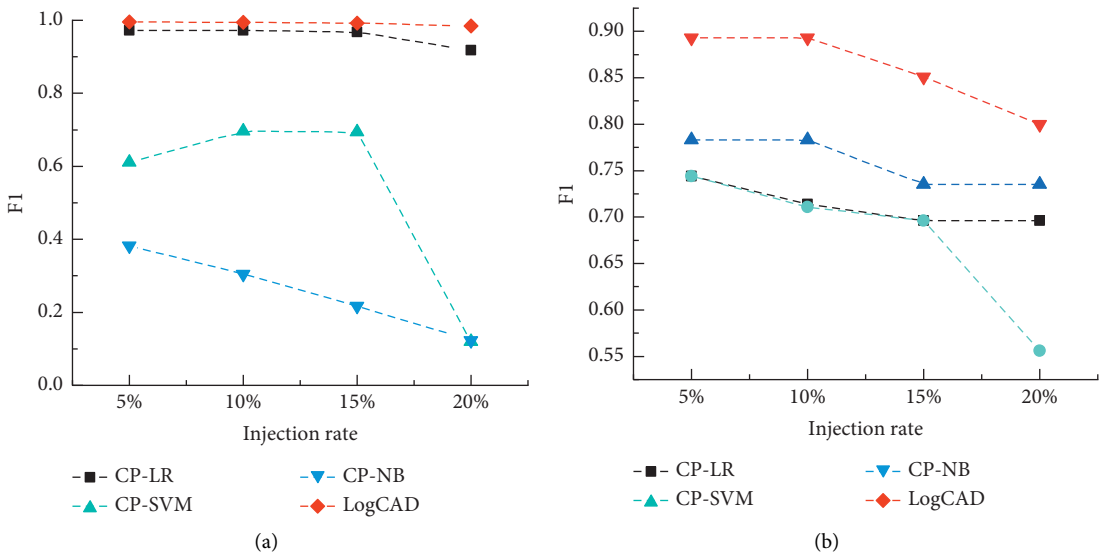

Figure 8: Variation of F1 scores in two data sets with different inject rates. (a) HDFS data set. (b) BGL data set.

Table 6: Effectiveness of multiple models on HDFS log data set.

Injection rate (%)	Classifier	Accuracy	Precision	Recall	F1	MCC
5	CP-AB-LR	0.998	0.950	1	0.973	0.976
	CP-AB-SVM	1	0.992	1	0.996	0.996
	CP-AB-NB	1	0.992	1	0.996	0.996
	LogCAD	1	0.992	1	0.996	0.996
10	CP-AB-LR	0.998	0.96	1	0.98	0.976
	CP-AB-SVM	0.999	0.996	0.992	0.994	0.994
	CP-AB-NB	0.999	0.974	0.997	0.985	0.984
	LogCAD	0.999	0.996	0.992	0.994	0.994
15	CP-AB-LR	1	0.983	0.989	0.986	0.983
	CP-AB-SVM	0.999	0.989	0.987	0.988	0.986
	CP-AB-NB	0.999	0.972	0.991	0.982	0.981
	LogCAD	**1**	**0.992**	**0.992**	**0.992**	**0.992**
20	CP-AB-LR	0.999	0.942	0.982	0.963	0.960
	CP-AB-SVM	0.998	0.947	0.995	0.971	0.969
	CP-AB-NB	1	0.958	1	0.979	0.975
	LogCAD	0.999	**0.975**	0.996	**0.985**	**0.985**

Table 7: Effectiveness of multiple models on BGL_100K log data set.

Injection rate (%)	Classifier	Accuracy	Precision	Recall	F1	MCC
5	CP-AB-LR	0.78	0.864	0.76	0.809	0.648
	CP-AB-SVM	0.805	0.87	0.8	0.833	0.676
	CP-AB-NB	0.756	0.857	0.72	0.783	0.622
	LogCAD	**0.854**	0.806	**1**	**0.893**	**0.71**
10	CP-AB-LR	0.756	0.888	0.8	0.8	0.613
	CP-AB-SVM	0.683	0.75	0.72	0.735	0.55
	CP-AB-NB	0.78	0.786	0.88	0.83	0.624
	LogCAD	**0.854**	0.806	**1**	**0.893**	**0.71**
15	CP-AB-LR	0.732	0.85	0.68	0.756	0.598
	CP-AB-SVM	0.829	0.909	0.8	0.851	0.709
	CP-AB-NB	0.756	0.857	0.72	0.783	0.622
	LogCAD	**0.829**	**0.909**	**0.8**	**0.851**	**0.709**
20	CP-AB-LR	0.732	0.792	0.76	0.776	0.592
	CP-AB-SVM	0.756	0.8	0.8	0.8	0.613
	CP-AB-NB	0.683	0.75	0.72	0.735	0.55
	LogCAD	**0.756**	**0.8**	**0.8**	**0.8**	**0.613**

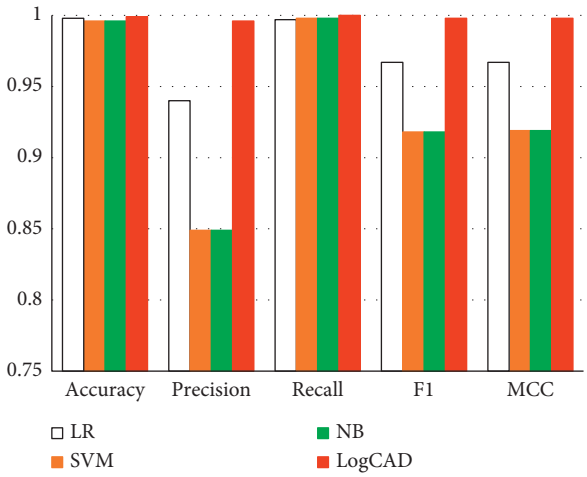

Figure 9: Results on stable HDFS data set

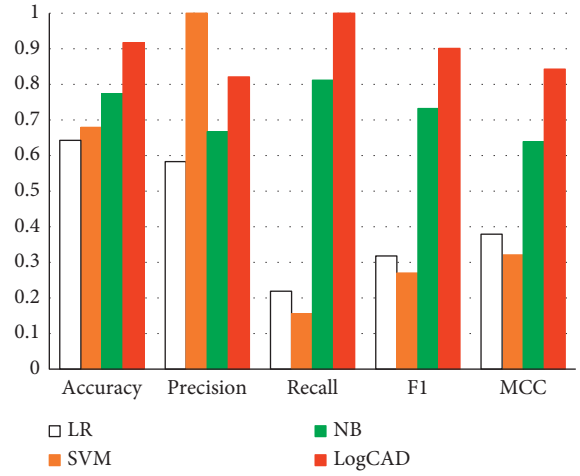

Figure 10: Results on stable BGL_100K data set.

In contrast to the traditional unique decision, the decision process of LogCAD is the nonconformity score generated by multiple single decision classifiers after several iterations, which makes the statistical p value of measuring log nonconformity more accurate. Thus the anomaly detection result is also more accurate to mitigate the problem of various mistakes caused by unique decision classifiers. Second, LogCAD can add misdetected unstable log scores to the previous score set, updating the old logs to better adapt to unstable environments. However, the traditional unique classifiers cannot relearn the previously unseen log sequences to adapt to the dynamically changing logging environment, so they cannot get satisfactory results.

4.3.2. Effectiveness of Ensemble Learning. In order to figure out the influence of ensemble learning on conformal anomaly detection (CAD), Figure 8 reveals in detail the CAD experimental results with or without ensemble learning in two data sets with different injection rates. CP-LR, CP-SVM, and CP-NB are conformal anomaly detection models over different underlying classifiers without ensemble learning, while LogCAD is complemented by ensemble learning.

As can be seen in Figure 8(a), CP-LR has the highest F1 score in conformal prediction models without ensemble learning whenever the injection rate is 5%, 10%, 15%, or 20%, while the F1 score of LogCAD is still higher than that of CP-LR. As shown in Figure 8(b), F1 scores of all classifiers present a downward trend with the increase of injection rate. In conformal prediction models without ensemble learning, CP-NB gains the highest F1 score, while the F1 score of LogCAD is about 10% higher than that of CP-NB on average, achieving a supreme performance.

4.3.3. Effectiveness of Multimodel Fusion. LogCAD is a conformal anomaly detection approach that incorporates multiple underlying learning models. We compare it with unimodel-underlied conformal anomaly detection algorithms to demonstrate its effectiveness. As shown in Table 6, LogCAD is not obviously superior to unimodel algorithms (CP-AB-LR is abbreviated for Conformal Prediction – AdaBoost – Logistic Regression, CP-AB-SVM for Conformal Prediction – AdaBoost – Support Vector Machine, CP-AB-NB for Conformal Prediction – AdaBoost – Naïve Bayes) on the HDFS data set when the unstable data are injected at the low rate of 5% and 10%. It is mainly because unimodel algorithms can also accurately identify anomalies in less unstable data. With the increase of unstable log proportion, LogCAD's advantages are becoming apparent. Its F1 and MCC values are higher than unimodel algorithms'. In Table 7, LogCAD performs best on the BGL_100K data set when the injection rate is from 5% to 20% and gains about 15% improvement in each metric compared to unimodel algorithms. The reason is that multimodel fusion means that multiple decision-makers make decisions collaboratively, which allows for greater avoidance of the possibility of unimodel decision errors.

4.4. Stable Log Data Set Experiments. The HDFS data set is collected on an unmodified Hadoop system [25], and the BGL data is recorded from the Lawrence Livermore National Laboratory (LLNL) BlueGene/L supercomputer system [22], so they do not have log sequence changes in the source code. In addition, all log events of the HDFS and BGL_100K data sets are identified directly from the source code to exclude parsing errors and other processing noise. The original HDFS and BGL_100K data sets are stable data sets. We apply LogCAD to them. The training sets keep the same as shown in Tables 2 and 3. The remaining 51,000 log sequences in the original HDFS data set are used as a test set, 1,000 of which indicate abnormal behavior. The remaining 40,000 log sequences in the original BGL_100K data set are used as test sets, 1,045 of which represent abnormal behavior.

The experimental results on the HDFS data set are shown in Figure 9. It can be seen that the F1 and MCC values of LogCAD are higher than those of traditional classifiers LR, DT, SVM, and NB. Although Recall of traditional classifiers

is very high, their F1 and MCC values are not as high as those of LogCAD.

The experimental results on the BGL_100K data set are shown in Figure 10. It can be seen that NB performs best in traditional classifiers. Compared with NB, the F1 and MCC values of LogCAD are significantly higher.

The above experimental results prove that LogCAD can be effectively applied not only to unstable log data sets, but also to stable log data sets.

5. Conclusion

Many log data-based anomaly detection methods have been proposed to automatically identify anomalies in large-scale systems [25–27]. However, when facing the problem of instability in logs, these methods usually make use of periodic retraining. Retraining too many times will exhaust resources and make systems too slow to learn new information in time. In addition, they usually require domain knowledge. So they are inefficient and too time-consuming. Therefore, this paper proposes a new anomaly detection method called LogCAD for unstable logs. It ties together the new logs and training log data sets and incrementally updates historical experience for decision-making in unstable logs, avoiding periodic retraining. LogCAD selects multiple ensemble algorithms and makes decisions together based on the confidence, rather than relying on a single algorithm, effectively reducing single decision errors. Excellent results are obtained on HDFS and BGL_100K log data sets, and better performance is achieved in Accuracy, Recall, and F1 value, which verifies the effectiveness of LogCAD algorithm in dealing with unstable log problem.

Data Availability

The research data supporting the results of this study can be available from https://zenodo.org/record/3227177#.XvVGE20zbIU.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

Acknowledgments

The author thanks the Chinese University of Hong Kong for providing the HDFS log data. This work was supported by the National Science Foundation of China under Grants 61872202 and 61601467; the Civil Aviation Safety Capacity Building Foundation of China under Grants PESA2019073, PESA2019074, and PESA2020100; the Natural Science Foundation of Tianjin under Grant 19JCYBJC15500; Key Research Program of the Chinese Academy of Sciences under Grant no. KFZD-SW-440; 2019 Tianjin New Generation AI Technology Key Project under Grant 19ZXZNGX00090; and Tianjin Key Research and Development Plan under Grant 20YFZCGX00680.

References

[1] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Fast Entropy Based Alert Detection in Super Computer Logs," in *Proceedings of the 2010 International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 52–58, IEEE, Chicago, Illinois, USA, June 2010.

[2] A. Oprea, Z. Li, T. F. Yen, and S. H. Chen, "Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data," in *Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, Rio de Janeiro, Brazil, June 2015.

[3] W. Xu, L. Huang, A. Fox, and D. Patterson, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, pp. 117–132, Big Sky Montana USA, October 2009.

[4] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[5] C. Liu, Y. Ren, M. Liang, and L. Pan, "Detecting Overlapping Data in System Logs Based on Ensemble Learning Method," *Wireless Communications And Mobile Computing*, vol. 2020, Article ID 8853971, 2020.

[6] Z. Gu, Y. Ren, and C. Liu, "Intranet log anomaly detection model based on conformal prediction," *Netinfo Security*, vol. 20, no. 3, pp. 45–50, 2020.

[7] R. Jordaney, K. Sharad, and S. K. Dash, "Transcend: detecting concept drift in malware classification models," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*, pp. 625–642, VANCOUVER, BC, CANADA, August 2017.

[8] D. Hu, Z. Ma, X. Zhang, and P. Li, "The Concept Drift Problem in Android Malware Detection and its Solution," *Security and Communication Networks*, vol. 2017, Article ID 4956386, 2017.

[9] M. Chen, A. X. Zheng, J. Lloyd, and M. J. Jarden, "Failure diagnosis using decision trees," in *Proceedings of the International Conference on Autonomic Computing*, New York, NY, USA, May 2004.

[10] W. Peng, T. Li, and S. Ma, "Mining logs files for data-driven system management," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 1, pp. 44–51, 2005.

[11] Y. Liang, Y. Zhang, and H. Xiong, "Failure prediction in IBM BlueGene/L event logs," in *Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007)*, Omaha, NE, USA, October 2007.

[12] P. He, J. Zhu, S. He, and J. Li, "Towards automated log parsing for large-scale log data analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 931–944, 2017.

[13] S. Lal and A. Sureka, "LogOpt: static feature extraction from source code for automated catch block logging prediction," in *Proceedings of the 9th India Software Engineering Conference*, Goa, India, February 2016.

[14] X. Zhang, Y. Xu, and C. Xie, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 807–817, Tallinn, Estonia, August 2019.

[15] X. Xie, Z. Wang, X. Xiao, and Y. Lu, "A Confidence-Guided Evaluation for Log Parsers Inner Quality," *Mobile Networks and Applications*,vol. 26, no. 6, pp. 1–12, 2020.

[16] P. He, J. Zhu, and S. He, "An Evaluation Study on Log Parsing and its Use in Log Mining," in *Proceedings of the 2016 46th*

Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 654–661, IEEE, Toulouse, France, June 2016.

[17] J. Zhu, S. He, J. Liu, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 121–130, IEEE, Montreal, QC, Canada, May 2019.

[18] L. Tang, T. Li, and C. S. Perng, "LogSig: generating system events from raw textual logs," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pp. 785–794, Glasgow, UK, October 2011.

[19] V. Vovk, V. Fedorova, I. Nouretdinov, and A. Gammerman, "Criteria of efficiency for conformal prediction," *Lecture Notes in Computer Science*, Springer, Cham, pp. 23–39, 2016.

[20] Y. Ren, Z. Gu, Z. Wang, and X Du, "System log detection model based on conformal prediction," *Electronics*, vol. 9, no. 2, Article ID 232, 2020.

[21] M. Du and F. Li, "Spell: Streaming Parsing of System Event Logs," in *Proceedings of the 2016 IEEE 16th International Conference On Data Mining (ICDM)*, pp. 859–864, IEEE, Barcelona, Spain, December 2016.

[22] J. Breier and J. Branišová, "Anomaly detection from log files using data mining techniques," *Lecture Notes in Electrical Engineering*, 2015.

[23] W. Xu, *System Problem Detection Mining Console*, University of California, Berkeley, 2010.

[24] A. Oliner and J. Stearley, "What Supercomputers Say: A Study of Five System Logs," in *Proceedings of the37th Annual IEEE/IFIP International Conference On Dependable Systems And Networks (DSN'07)*, pp. 575–584, IEEE, Edinburgh, UK, June 2007.

[25] W. Xu, L. Huang, and A. Fox, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd on Operating Systems Principles*, pp. 117–132, 2009.

[26] T. Jia, Y. Li, and Z.-H. Wu, "Survey of state-of-the-art log-based failure diagnosis," *Journal of Software*, vol. 31, no. 7, pp. 1997–2018, 2020.

[27] C. Liu, L. Pan, and Z. Gu, "Valid probabilistic anomaly detection models for system," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8827185, 2020.

[28] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003)*, pp. 119–126, IEEE Cat. No. 03EX764, 2003.

[29] A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "A lightweight algorithm for message type extraction in system application logs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 11, pp. 1921–1936, 2012.

[30] Q. Fu, J. G. Lou, and Y. Wang, "Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, pp. 149–158, IEEE, Miami, Florida, USA, December 2009.