A Semi-supervised Approach for Industrial Anomaly Detection via Self-adaptive Clustering

Xiaoxue Ma, Jacky Keung, *Senior Member, IEEE*, Pinjia He, *Member, IEEE*, Yan Xiao, Xiao Yu, and Yishu Li

***Abstract*— With the rapid development of the Industrial Internet of Things (IIoT), log-based anomaly detection has become vital for smart industrial construction that has prompted many researchers to contribute. To detect anomalies based on log data, semi-supervised approaches stand out from supervised and unsupervised approaches because they only require a portion of labeled data and are relatively stable. However, the state-of-the-art semi-supervised approaches still suffer from two main problems: manual parameter setting and unsatisfactory performance with high false positives. We propose AdaLog, an integrated semi-supervised approach based on self-adaptive clustering, for industrial anomaly detection. In particular, the clustering step performs automatic label probability estimation by distinguishing twelve situations so that the label probability of each unlabeled data can be carefully calculated, leading to high accuracy. In addition, AdaLog employs a pre-trained model to learn contextual information comprehensively and a transformer-based model to detect anomalies efficiently. To alleviate class imbalance, an undersampling method is incorporated. The results on three popular datasets demonstrate that AdaLog significantly outperforms three state-of-the-art semi-supervised approaches by 17.8%–2,489.8% on average in terms of F1-score, and is even superior to two supervised approaches in most cases with average improvements of 10.9%–23.8%.**

***Index Terms*— Intelligent anomaly detection, clustering, transformer, deep learning.**

I. Introduction

DUE to the needs of intelligent data processing and analysis [1]–[3] in the Industrial Internet of Things (IIoT), anomaly detection has attracted a lot of attention [4]. A trivial industrial system anomaly can lead to a series of problems, such as data corruption and product performance degradation [5], [6]. To this end, logs have been widely utilized in various reliability enhancement tasks, such as system status checks, monitoring facility identification, and anomaly detection, because logs are essential data for recording system runtime information [7]. In particular, log-based anomaly detection has inspired a line of research using machine learning and deep learning. Generally, existing approaches can be categorized into supervised (e.g., LogRobust [8], NeuralLog [5]), semi-supervised (e.g., DeepLog [9], LogAnomaly [10], and PLELog [11]), and unsupervised approaches (e.g., LogCluster [12]).

Typically, supervised approaches perform better than the others since tons of data with labels (i.e., normal or abnormal) are used during model training. However, these approaches are not practical in the industry because of their high demand for large labeled datasets and the sensitivity to mislabeled data [6], [11]. Regarding unsupervised techniques, non-negligible log data instability can lead to unfavorable performance. Because of frequent log modifications, some incoming log sequences do not appear in the training set [11]. Therefore, effectively deploying unsupervised approaches in the industry can be very challenging.

Compared with supervised approaches, semi-supervised approaches only require a small amount of labeled data in model training. It dramatically reduces the cost of manual labeling and the reliance on labeled data. Compared with unsupervised approaches, semi-supervised approaches know part of the training label information (e.g., the correlations between features and labels), which can be utilized by semi-supervised approaches to maximize the training objective.

Although existing semi-supervised approaches achieve decent accuracy on public log datasets (e.g., HDFS [13]), several crucial problems have been overlooked and remain unresolved, hindering their usage in practice. *First*, real-world logs data are significantly imbalanced, i.e., there are many more normal logs than abnormal logs. For example, only 0.82% of the logs in Thunderbird dataset are anomalous. The highly imbalanced real-world data lead to a situation where anomalies are often identified as normal cases by existing approaches. *Second*, most semi-supervised approaches rely on clustering algorithms that cluster the unlabeled training data into different groups. This process is notoriously sensitive to the parameter settings (e.g., cluster size and minimum samples [11] and similarity threshold [14]), which often requires extensive manual efforts.

This work was supported in part by the General Research Fund of the Research Grants Council of Hong Kong (No. 11208017), the research funds of the City University of Hong Kong (No. 7005028, 7005217 and 6000796), and funding support from other industry projects (No. 9229109, 9229098, 9229029, 9220103, 9220097 and 9440227) (Corresponding authors: Yan Xiao; Xiao Yu).

Xiaoxue Ma, Jacky Keung, and Yishu Li are with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: xiaoxuema3-c@my.cityu.edu.hk; jacky.keung@cityu.edu.hk; yishuli5-c@my.cityu.edu.hk).

Pinjia He is with the School of Data Science, The Chinese University of Hong Kong, Shenzhen 518172, China.(e-mail: hepinjia@cuhk.edu.cn).

Yan Xiao is with the School of Cyber Science and Technology, Sun Yat-Sen University, Shenzhen 518107, China. (e-mail: xiaoyan.hhu@gmail.com).

Xiao Yu is with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China (e-mail: xiaoyu@whut.edu.cn).

Third, the adopted clustering algorithms have a huge impact on the performance of the whole semi-supervised pipeline [11], [14]. Existing clustering algorithms simply predict labels for unlabeled samples without distinguishing different situations (e.g., the correlation between samples in a cluster). This rough labeling often results in severe performance degradation (e.g., lower precision and specificity) in the subsequent steps due to high false positives (FP).

To address these problems, we propose **AdaLog**, a semi-supervised approach based on self-Adaptive clustering for Log-based anomaly detection. AdaLog adopts a self-adaptive clustering method based on K-Means by dividing label probability calculation into twelve situations, which largely enhances the clustering performance. Specifically, AdaLog considers the distance between each unlabeled sample and its cluster centroid and the distance averages of labeled two class samples. In this way, the label probability of each unlabeled data is carefully calculated and used for the subsequent model training. In addition, by employing the Elbow method [15] to compute the sum of squared distances between each log and its corresponding cluster centroid, AdaLog mitigates the burden of manual parameter tuning, allowing for the recommended number of clusters to be automatically determined. To tackle dataset imbalance, AdaLog incorporates an undersampling method before clustering to reduce the imbalanced ratio of normal and abnormal data. Moreover, AdaLog employs a pre-trained BERT model [16] for semantic representation and a Transformer-based model [17] for log sequence classification.

We evaluate the performance of AdaLog with comprehensive experiments on three widely-used log anomaly datasets (i.e., HDFS [13], BGL [18], and Thunderbird [18]). The results demonstrate that AdaLog remarkably outperforms three state-of-the-art (SOTA) semi-supervised approaches with an average improvement (in terms of F1-score) of 2489.8% (DeepLog [9]), 2448.9% (LogAnomaly [10]), and 17.8% (PLELog [11]), respectively. Moreover, AdaLog even outperforms two SOTA supervised approaches in F1-score by 23.8% (LogRobust [8]) and 10.9% (NeuralLog [5]). The ablation studies also indicate the effectiveness of our proposed self-adaptive clustering method and the necessity of the undersampling method.

The main contributions of this paper are as follows:

- We propose AdaLog, a semi-supervised log-based anomaly detection approach that addresses the three main concerns of existing semi-supervised approaches.
- The core of AdaLog is a self-adaptive clustering method with twelve specially-designed situations for label probability calculation.
- The experimental results show that AdaLog outperforms SOTA semi-supervised and supervised methods in F1-score by 17.8%–2489.8% and 10.9%–23.8%, respectively.
- Our implementation is publicly accessible[1].

II. Related Work

A. Anomaly Detection with Supervised Techniques

In existing log-based anomaly detection papers, supervised approaches often achieve better performance compared with semi-supervised and unsupervised approaches because supervised approaches leverage a large amount of labeled training data in their evaluation. Many machine learning-based methods were proposed to detect anomalies in the early years. Liang *et al.* [19] introduced four classifiers to predict failure log events. The decision tree model applied by Chen *et al.* [20] was used to classify logs on eBay's web request logging system, and the regression-based approach presented by Farshchi *et al.* [21] was used to detect application operation failures. Subsequently, deep learning-based methods were proposed to detect log-based anomalies. For example, Zhang *et al.* [22] combined a log template extraction method with TF-IDF to represent templates as vectors and applied an LSTM model for log-based system failure prediction. Similarly, Vinayakumar *et al.* [17] employed a Stacked-LSTM network for log anomaly detection. Wu *et al.* [23] provided an agile solution EdgeLSTM for sequential computation in IoT data, which uses Grid LSTM and multi-class SVM. This method was deployed for anomaly detection applications. To further optimize the LSTM model applied in time-series data analysis, a hyperparameter optimization method was proposed by Wu *et al.* [24] to reduce the time cost and enhance the performance. LogRobust [8] and NeuralLog [5], as state-of-the-art supervised approaches mentioned in Section IV-B, utilize a pre-trained model (i.e., FastText and BERT) for embedding and a deep learning model (i.e., BLSTM and Transformer) combining with attention mechanism for prediction. Supervised approaches require substantial labeled data, which is often infeasible in practice. Different from these approaches, AdaLog is semi-supervised, which only needs a small amount of labeled data yet achieves comparable accuracy as supervised approaches.

B. Anomaly Detection with Semi-supervised and Unsupervised Techniques

Compared with supervised approaches, semi-supervised and unsupervised approaches are more likely to be applied in practice due to their less reliance on labeled data. DeepLog [9] used an LSTM model to learn log patterns from normal log sequences automatically, it can detect anomalies if incoming log sequences cause the model to deviate from normal execution. To learn normal patterns, LogAnomaly [10] considered semantic information by matching log sequences against their generated templates. Wu *et al.* [3] adopted an LSTM model and a Gaussian Bayes model for outlier detection in the IIoT. PLELog [11]) combined the HDBSCAN clustering method for probabilistic label estimation and an attention-based GRU model for log sequence classification. LogCluster [12], an unsupervised approach, considered the weights of log events and grouped log sequences via a hierarchical clustering algorithm. However, the unsupervised approach usually performs worse than those semi-supervised approaches that combine a part of normal labeled data and deep learning techniques. Unlike existing semi-supervised approaches, AdaLog effectively enhances anomaly detection performance because it considers

[1] https://github.com/AdaLog2023/AdaLog

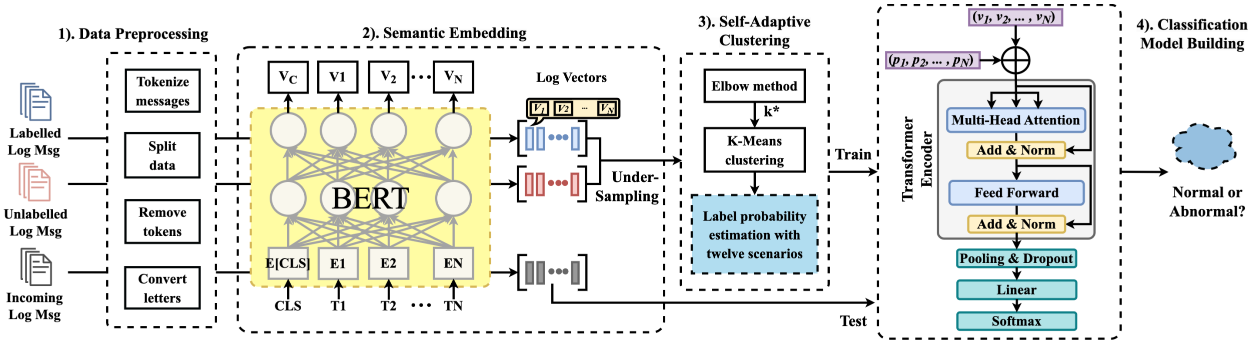


Fig. 1. The four-phrase framework of AdaLog.

various cases of label probability estimation via a self-adaptive clustering method.

III. AdaLog

The workflow of AdaLog is illustrated in Fig. 1, which contains four modules: data preprocessing, semantic embedding and undersampling, self-adaptive clustering, and classification model building.

AdaLog 1) preprocesses the log messages in a standard format (Section III-A). The processed log data is 2) tokenized and transformed into vector representations by a pre-trained BERT model (Section III-B). After that, the training log messages are grouped into log sequences with/without ground truth labels (i.e., labeled or unlabeled), and then are processed through undersampling. 3) The remaining log sequences are collected together for automatic clustering and label probability estimation (Section III-C). AdaLog 4) builds the Transformer-based model for classification (Section III-D) and trains it using the training data with estimated labels. Once deployed, AdaLog first generates log vectors by steps 1) and 2) given a test log sequence. The vectors are then fed into the trained model, from which the predicted output will be normal or abnormal.

A. Data Preprocessing

The first step of AdaLog is preprocessing the raw log messages. To avoid the errors caused by log parsing (e.g., Drain, Spell, AEL, and IPLoM) due to the lack of semantic information and semantic misunderstanding [5], we keep the full information of each log message. The raw log messages are tokenized into word-based tokens and then separated by common delimiters (e.g., commas, semicolons, and white spaces). Inspired by NeuralLog [5], AdaLog removes all non-character tokens, such as numbers, operators, and punctuations, which preserves most of the informative content. In addition, AdaLog converts the capital letters to lower letters to make the overall process case-insensitive. An example of preprocessing is shown in Table I.

TABLE I
An Example of Preprocessing on BGL Log

Original: - 1117958437 2005.06.05 R37-M0-N0-C:J13-U01 2005-06-05-01.00.37.726577 R37-M0-N0-C:J13-U01 RAS KERNEL INFO 1 ddr errors(s) detected and corrected on rank 0, symbol 12, bit 3
Preprocessed: ras kernel info ddr errors s detected and corrected on rank symbol bit

B. Semantic Embedding and Undersampling

To better understand the semantic information of log messages, we adopt a deep learning pre-trained model BERT [16] to extract features and represent log messages as embedding vectors. The architecture of the pre-trained model consists of multi-layers of bidirectional Transformer encoder. Every encoder utilizes the self-attention mechanism to determine the terms they should pay more attention on and draw global dependencies between inputs and outputs. Hence, each input log message from the training set is passed through the first-layer encoder to generate embedding vectors, which serve as the input of the next layer of the encoder. The word embeddings generated by the last-layer encoder of BERT are used for the next step. In this way, each log message can be represented as a fix-length vector, i.e., $V = \{v_1, v_2, ..., v_N\}$, where N is the number of tokens in each log message. Subsequently, the log messages are grouped into log sequences by session or fixed windows.

As we mentioned in Section I, class imbalance exists in the log-based anomaly detection datasets. Therefore, we employ an undersampling method [25] to remove a certain proportion of normal log sequences to alleviate this problem. We did not use oversampling since oversampling may lead to overfitting during the model construction process, especially when abnormal data is very limited. To handle the varying degrees of imbalance in the datasets, we empirically design the following heuristic rule: use an undersampling ratio range and then specify three-quarters of the undersampling range as the specific undersampling ratio to conduct undersampling. Finally, we remove normal log sequences from the training dataset according to the order of the collected data. The remaining normal log sequences and all abnormal log sequences are sent to the clustering stage as training data.

C. Self-adaptive Clustering

Typically, the training dataset of semi-supervised approaches consists of labeled and unlabeled data. To utilize labeled data, semi-supervised approaches employ clustering algorithms to form groups of similar samples, based on which we can estimate the label probability of the unlabeled data according to specially-designed different situations.

Existing clustering methods can be classified into four categories. Since the hierarchical-based clustering technique has the drawback of high time complexity, the density-based and grid-based are sensitive to parameter selection, AdaLog uses K-Means [26], a widely-used partition-based method to cluster the samples. The principle of this method is to optimize the squared error distortion between the samples and the centroids to minimize the within-cluster variance by adjusting the centroids.

(1) K-Means Clustering with Elbow Method. As explained above, we choose the effective and straightforward method K-Means for rough clustering. To ensure the most suitable k (i.e., the number of clusters), we use the Elbow method [15] to compute the recommended k^*. The Elbow method is centered around the computation of the sum of squared errors (SSE). The SSE measures the sum of the squared distances between each data sample and its corresponding cluster centroid, which represents the within-cluster variability. The SSE can be regarded as the clustering error of all samples, and serves as an indicator of the clustering quality. When the number of clusters, denoted by k, is smaller than the true number of clusters, increasing k will lead to a significant decrease in SSE due to the increased degree of aggregation within each cluster. However, as k approaches the true number of clusters, denoted by k^*, the rate of decrease in SSE will begin to plateau, resulting in an elbow-shaped curve. The value of k^* corresponding to the elbow point represents the optimal number of clusters for the given dataset. To compute the SSE for each value of k within a range $K = (0, 50)$, we sum the squared distance as shown in the following equation:

$$SSE_k = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2. \tag{1}$$

where C_i represents the i-th cluster, p denotes a sample point belonging to C_i, and m_i indicates the centroid of C_i. Based on our empirical findings, we recommend adopting the first two optimal k^* values for subsequent operations, as they may be very close in certain scenarios. For each chosen k^*, we employ K-Means to cluster our training data on the embedding space. K-Means iteratively computes centroids by calculating the index of the cluster to which each sample is assigned and the distance between each sample and its nearest cluster centroid. When the calculated index of each sample's cluster is its nearest cluster, the iteration stops, and the centroids will no longer change. In this way, the labeled/unlabeled data has its corresponding cluster under each optimal k^*.

(2) Label Probability Estimation. In this step, our aim is to compute the probability P_{normal} for each unlabeled sample being normal under each k^* value, and the final P_{normal} is obtained by taking the average. Compared to other approaches, AdaLog proposes a novel way to leverage the clustering results and existing labeled samples in the clusters by considering twelve specially-design situations. By doing so, existing labeled data can contribute to the label probability estimation of unlabeled data. We discuss twelve scenarios and demonstrate the computation in Table II. In addition, we keep the $P_{normal} = 1$ for labeled normal samples and $P_{normal} = 0$ for labeled anomalous samples.

The twelve scenarios can be classified as five main situations ($\mathscr{S}1$-$\mathscr{S}5$) based on which cluster an unlabeled sample belongs to. We calculate the label probability P_{normal} for each unlabeled sample by comparing the distances d, d_A, and d_N.

$$\begin{aligned} d_N &= \frac{1}{N_n} \sum_{i}^{N_n} d_i, \\ d_A &= \frac{1}{N_a} \sum_{j}^{N_a} d_j, \end{aligned} \tag{2}$$

where d represents the distance from the unlabeled sample to its corresponding cluster centroid. d_N and d_A are the average distances between each labeled normal/anomalous sample in this cluster and the cluster centroid. N_n and N_a indicate the number of labeled normal and abnormal samples, respectively.

The first situation $\mathscr{S}1$ is that the corresponding cluster only contains one known category (normal or abnormal). $\mathscr{S}1.1$ indicates the cluster only has labeled normal samples (i.e., so we have d_N). Therefore, this unlabeled sample is very likely to be normal, and a large probability (P_{normal} is almost closer to 1) is given to this sample. In contrast, $\mathscr{S}1.2$ describes the situation in that only labeled abnormal samples exist in the cluster (i.e., so we have d_A). Accordingly, a small probability (P_{normal} is almost closer to 0) is given to this sample. $\mathscr{S}2$ demonstrates another situation the cluster does not contain any labeled samples. Since there is no preference for the possible estimated categories, we assign a medium probability to the unlabeled sample (i.e., $P_{normal} = 0.5$).

For the $\mathscr{S}3$ and $\mathscr{S}4$ (i.e., the cluster contains two known categories), the average distances d_N and d_A should be considered to decompose the situation. For each sub-situation, we introduce two coefficient parameters P_1 and P_2 for calculating P_{normal} as shown in Table II. P_1 represents the confidence degree in which the sample is more inclined to be normal or abnormal, according to the average distances (i.e., d_N and d_A) from the normal and abnormal samples to the cluster centroid. P_2 refers to the confidence degree that the unlabeled sample is more likely to be normal or abnormal, depending on which category the sample is closer to (i.e., $|d - d_N|$ and $|d - d_A|$). In addition, we utilize constants (i.e., 0.5) to ensure that the P_{normal} is located in the reasonable range. If $d_N < d_A$ (i.e., $\mathscr{S}3$), it demonstrates that the cluster may be more towards the normal category. For $\mathscr{S}3.1$, if the situation satisfies $d < d_N$, the unlabeled sample is closer to the cluster centroid, so it is more likely to be normal (i.e., $P_{normal} \in (0.5, 1)$). The coefficent parameters (P_1 and P_2) expressed as $\frac{d_N}{d_A}$ and $\frac{d_N - d}{d_A - d}$ are used to assess the degree to which the sample tends to be normal. Take the scenario as an example, 1) if the d_N approximates to d_A, it becomes challenging to determine to which category the sample is more likely to belong because known normal and abnormal data are very close in distance within the cluster. Therefore, P_1 as a coefficient cannot excessively bias the sample towards the normal category, even if the sample is closer to the normal

TABLE II
SITUATIONS OF SELF-ADAPTIVE CLUSTERING.

Situation			$\boldsymbol{P_{normal}}$
$\mathscr{S}$1. The cluster only has one known category.	$\mathscr{S}$1.1 labeled normal samples d_N		$P = 0.99$
	$\mathscr{S}$1.2 labeled abnormal samples d_A		$P = 0.01$
$\mathscr{S}$2. The cluster does not have any known category. The cluster only has unlabeled samples.			$P = 0.5$
$\mathscr{S}$3. The cluster contains two known categories, and $d_N < d_A$.	$\mathscr{S}$3.1 $d \leq d_N < d_A$		$P = 1 - (\frac{d_N - d}{d_A - d} * \frac{d_N}{d_A} * 0.5)$, $P \in (0.5, 1)$
	$\mathscr{S}$3.2 $d_N \leq d < d_A$	$\mathscr{S}$3.2.1 $d - d_N \leq d_A - d$	$P = 0.5 + (1 - \frac{d - d_N}{d_A - d}) * (1 - \frac{d_N}{d_A}) * 0.5$, $P \in (0.5, 1)$
		$\mathscr{S}$3.2.2 $d - d_N > d_A - d$	$P = 0.5 - (1 - \frac{d_A - d}{d - d_N}) * (1 - \frac{d_N}{d_A}) * 0.5$, $P \in (0, 0.5)$
	$\mathscr{S}$3.3 $d_N < d_A \leq d$		$P = 0.5 - (1 - \frac{d - d_A}{d - d_N}) * (1 - \frac{d_N}{d_A}) * 0.5$, $P \in (0, 0.5)$
$\mathscr{S}$4. The cluster contains two known categories, and $d_A < d_N$.	$\mathscr{S}$4.1 $d < d_A < d_N$		$P = 0 + (\frac{d_A - d}{d_N - d}) * \frac{d_A}{d_N} * 0.5$, $P \in (0, 0.5)$
	$\mathscr{S}$4.2 $d_A \leq d < d_N$	$\mathscr{S}$4.2.1 $d - d_A \leq d_N - d$	$P = 0.5 - (1 - \frac{d - d_A}{d_N - d}) * (1 - \frac{d_A}{d_N}) * 0.5$, $P \in (0, 0.5)$
		$\mathscr{S}$4.2.2 $d - d_A > d_N - d$	$P = 0.5 + (1 - \frac{d_N - d}{d - d_A}) * (1 - \frac{d_A}{d_N}) * 0.5$, $P \in (0.5, 1)$
	$\mathscr{S}$4.3 $d_A < d_N \leq d$		$P = 0.5 + (1 - \frac{d - d_N}{d - d_A}) * \frac{d_A}{d_N} * 0.5$, $P \in (0.5, 1)$
$\mathscr{S}$5. The cluster contains two known categories, and $d_N = d_A$.			$P = 0.5$

P represents the normal probability of each sample, which is calculated by comparing the clustering information of each unlabeled data (d) and labeled data (d_A and d_N).

category. Similarly, Since the distance between this sample and each category is very close, P_2 does not make much effort to push the sample to one of the categories. Consequently, P_1 and P_2 should be very close to 1, resulting in a more neutral label probability for this sample; 2) In contrast, if the d_A is far away from d_N, P_1 and P_2 should be more biased towards 0, which increases the probability of the sample being normal.

For $\mathscr{S}$3.2, when the sample lies between the two categories (i.e., $d_N \leq d < d_A$), two situations may arise. If $(d - d_N) \leq (d_A - d)$ ($\mathscr{S}$3.2.1), this means that although the sample is likely to be in either of the two categories, the sample is closer to the normal category (i.e., $P_{normal} \in (0.5, 1)$). These two coefficient parameters $(1 - \frac{d_N}{d_A})$ and $(1 - \frac{d - d_N}{d_A - d})$ cause the sample to be shifted in the normal direction; else if $(d - d_N) > (d_A - d)$ ($\mathscr{S}$3.2.2), the sample is more likely to be an anomaly (i.e., $P_{normal} \in (0, 0.5)$), and P_1 and P_2 are formulated as $(1 - \frac{d_N}{d_A})$ and $(1 - \frac{d_A - d}{d - d_N})$ to decide the degree to which the sample is biased like an anomaly. $\mathscr{S}$3.3 indicates that the sample is farther from the cluster centroid and even further than the abnormal category (i.e., $d_N < d_A \leq d$), therefore it is more prone to be an anomaly (i.e., $P_{normal} \in (0, 0.5)$). $(1 - \frac{d_N}{d_A})$ and $(1 - \frac{d - d_A}{d - d_N})$ are used to calculate the degree to which the sample is inclined to the abnormal category.

Similarly, the $\mathscr{S}$4 with $d_A < d_N$ is derived from the same calculation idea of the $\mathscr{S}$3. This situation demonstrates that the cluster is more likely to gravitate toward the abnormal category due to the fact that known anomalous data are closer to the cluster centroid. The last situation $\mathscr{S}$5 is that $d_N = d_A$. Since we cannot judge which category of the unlabeled sample is more likely to belong, we assign the $P = 0.5$ to it under this condition. It can be regarded as a special case of the above cases. Furthermore, we list the range of the calculated P_{normal} for each situation in Table II. It also implies that self-adaptive clustering is reasonable and rigorous.

In this way, unlabeled log sequences can be automatically classified into one of the twelve situations, then their corresponding label probabilities of being normal are calculated. We use ($P_{abnormal} = 1 - P_{normal}$) to represent the probability that a sample is an anomaly. The probability pair $(P_{normal}, P_{abnormal})$ for each log sequence is sent to the next classification model.

D. Classification Model Building

Based on the training data with ground-truth labels or label probabilities computed by the clustering method, we finally utilize a transformer-based classification model [17] to detect anomalies. Since a log sequence contains many log messages, relative position information should be incorporated. To this end, a sinusoidal encoder [17] used to generate the embedding p_i at position i is added to the corresponding embedding vector v_i, which is fed into the classification model as illustrated in step 4) of Fig. 1.

The network structure of the transformer encoder solves the limitation of the parallel ability, and better deals with the problem of long dependencies within log sequences. Typically, the transformer encoder stacks several identical layers. Each layer uses multi-head attention and a feed-forward network that contains two fully connected layers, plus layer normalization and residual connection. In this way, the combination of attention scores for each log message is obtained through the fully connected layers [27]. Afterward, the output of this transformer encoder is sequentially connected to a pooling layer, a dropout layer, and a fully connected layer with a softmax function. Since we train with the estimated label probabilities of the log sequences rather than binary labels (i.e., 0 or 1), the transformer-based model will try to optimize the loss between the computed label probabilities and the predicted binary labels. We have such a computation setup for two reasons. The first reason explains why we utilize estimated label probabilities. AdaLog as a semi-supervised method, there is no guarantee that every unlabeled data will be correctly classified by the predictions of the clustering method, and the mislabeled data may bias the training. Hence, we compute the label probabilities of the data to reduce the noise effect to some extent. The second reason explains why we choose the binary labels predicted by the transformer-based model to compute the loss instead of the probabilistic ones. Owing to the binary classification task, the final prediction result should be normal or abnormal (i.e., 0 or 1). If we use both probabilities to calculate the loss simultaneously, some labels of the data are likely to be neutral, making it difficult to judge the boundary between these two categories.

During the training, the best-trained parameters of this model will be saved for anomaly prediction. If a new log sequence comes, it is preprocessed and embedded before being fed into the trained model. Finally, its predicted label (normal or abnormal) will be generated as output with the above steps.

IV. Experimental Settings

A. Datasets

To evaluate AdaLog's performance in detecting anomalies, we use three widely-used datasets: Hadoop Distributed File System (HDFS) dataset [13], Blue Gene/L supercomputer (BGL) dataset [18]), and Thunderbird dataset [18]. In particular, Thunderbird is an extremely imbalanced dataset, which better reflects anomaly detection in practice. Similar to previous works [5], [6], [11], we group log messages of the HDFS dataset by session windows according to the log's identifier (i.e., $block\_id$). For other datasets (i.e., BGL and Thunderbird), log messages are grouped with fixed *window sizes* (i.e., 20, 100, and 200). In Table III, the number of log messages, the grouping strategies, and the corresponding grouped log sequences with abnormal proportions are presented in order.

B. Baselines

Recently, log-based anomaly detection as an interesting and novel research problem has attracted many studies proposing models to detect anomalies. Since supervised methods usually have the best performance and our approach is semi-supervised, we choose the state-of-the-art semi-supervised approaches (i.e., DeepLog [9], LogAnomaly [10], and PLELog [11]) and supervised approaches (i.e., LogRobust [8] and NeuralLog [5]) introduced in Section II as baselines.

C. Parameter Settings

In our experiments, we employ the default values of hyper-parameters that have been widely used in related work [5], [16], [28]. Specifically, we set the layer of the transformer encoder to be 1, the number of attention heads to be 12, and the size of the feed-forward network followed by multi-head self-attention to be 2048. The classification model is optimized by using an improved regularization technique in the Adam optimizer [29], which involves decoupling the weight decay from the gradient-based update. This model was trained with the learning rate of $3e-4$, the dropout rate is 64, the adopted loss function is weighted binary cross-entropy, and the number of epochs is 20. Furthermore, we use PCA for dimensionality reduction so that the number of components for our self-adaptive clustering is 100, the same as PLELog [11].

To make a fair comparison, we adopt the parameter values provided by the original authors. DeepLog [9] and LogAnomaly [10] adopt a two-layer LSTM with 128 neurons. LogRobust [8] utilizes two BLSTM layers with 128 neurons and one attention layer. PLELog [11] employs a single-layer GRU network for prediction, and its clustering parameter settings remain the same as they introduced (that is, the minimum cluster size and the minimum number of samples are 100) to calculate the number of clusters through HDBSCAN. NeuralLog [5] has the same parameter settings as the transformer encoder we described above. In the original work [10], LogAnomaly applied a synonym- and antonym-based approach to representing log templates as semantic vectors. However, the employed template2vec model was trained with domain-specific antonyms and synonyms, some of which were manually added by operators. Furthermore, the model was trained with the index of log events (i.e., sequential and quantitative vectors), thereby ignoring the semantics of the logs. Due to the unavailability of the information, we follow the empirical study [6], using a pre-trained FastText word2vec model [30] to compute semantic vectors for log templates. The semantic vectors obtained from language models are usually more informative.

The log sequences used in our study are continuous and sequential because of their chronological ordering and grouping based on window size or session window. Hence, the conventional k-fold cross-validation method is not applicable in our timing-related problems, considering the time dependencies present in the data. To assess the effectiveness of AdaLog, we perform each experiment 20 times and report the median value in Table IV. These results are more reliable and applicable in practical scenarios.

D. Evaluation Metrics

Log-based anomaly detection is an imbalanced binary classification problem. To evaluate the effectiveness of AdaLog, we refer to previous works and adopt the precision= $\frac{TP}{TP+FP}$, recall= $\frac{TP}{TP+FN}$, and F1-score= $\frac{2\times precision\times recall}{precision+recall}$ metrics for comparison. In addition, Le *et al.*'s work [6] demonstrated that specificity= $\frac{TN}{TN+FP}$ is necessary for evaluation, especially under the imbalanced data distribution. Thus, we introduce this metric in our study. Specifically, TP, TN, FP, and FN refer to the number of true positives (an abnormal log sequence is correctly predicted to be anomalous), true negatives (a normal log sequence is correctly predicted to

TABLE III
THE STATISTICS OF THREE DATASETS

Datasets	Category	Messages	Grouping	Log Sequences	Training Data (anomaly)	Test Data (anomaly)
HDFS	Distributed system	11,175,629	session	575,061	460,048 (2.9%)	115,013 (2.9%)
BGL	SuperComputer	4,747,963	ws=20	237,397	189,918 (9.1%)	47,479 (6.3%)
			ws=100	47,478	37,983 (10.5%)	9,495 (8.6%)
			ws=200	23,738	18,991 (11.6%)	4,747 (10.1%)
Thunderbird	SuperComputer	10,000,000	ws=20	499,998	399,999 (0.4%)	99,999 (0.1%)
			ws=100	99,998	79,999 (1.1%)	19,999 (0.1%)
			ws=200	49,998	39,999 (1.6%)	9,999 (0.3%)

be normal), false positives (a normal log sequence is incorrectly predicted to be abnormal), and false negatives (an abnormal log sequence is incorrectly predicted to be normal), respectively. Precision indicates the percentage of correctly detected anomalies among all detected anomalies; recall refers to the percentage of correctly detected anomalies over all anomalous log sequences; specificity represents the percentage of correctly detected normal ones over all real normal log sequences; and the F1-score considers both precision and recall, which is a comprehensive evaluation metric. The higher the F1-score value, the more accurate the proposed approach is in predicting both categories (i.e., normal and abnormal).

V. RESULTS AND DISCUSSION

In Section V-A, we present a comprehensive evaluation of the performance of our proposed AdaLog approach as well as SOTA approaches. Subsequently, we conduct ablation studies to assess the effectiveness of the self-adaptive clustering method, undersampling technique, and overall methodology in Sections V-B, V-C, and V-D, respectively. In Section V-B, we compare our proposed clustering method against the HDBSCAN clustering method used in PLELog, both without (Table V) and with (Table VI) the classification model. Additionally, we investigate the efficacy of the undersampling method at different sampling ratios in Section V-C. Finally, in Section V-D, we analyze how varying sizes of labeled data impact the performance of AdaLog. These sections provide a detailed analysis of our experimental results, further contributing to the overall efficacy and understanding of the proposed approach.

In Table IV–VIII, the column *M* indicates the evaluation metrics, and P, R, S, and F1 represent precision, recall, specificity, and F1-score, respectively. For each dataset, the grouping methods are abbreviated as *session* or the values of the *window sizes* (i.e., 20, 100, and 200).

A. Detection Accuracy

As shown in Table IV, in terms of F1-score, compared with **all semi-supervised approaches**, our approach AdaLog performs best on all datasets, **improving existing approaches by 2489.8% (DeepLog), 2448.8% (LogAnomaly), and 17.8% (PLELog)** on average. On HDFS dataset, all semi-supervised approaches perform well, and AdaLog outperforms these three approaches by 6.2% on average. On BGL dataset, regardless of window size, DeepLog and LogAnomaly have a relatively low F1-score because of the poor precision; that is, they are likely to mislabel normal log sequences as anomalies. AdaLog is superior to these two approaches by 237.0% and 221.2% on average, respectively. In contrast, PLELog performs better but still 21.8% worse than AdaLog on average. Every dataset has a class imbalance problem, especially Thunderbird dataset, where abnormal log sequences in the test set only account for 0.1%–0.3% of the entire test set under different window sizes. Beneficial from the undersampling method, AdaLog outperforms other approaches most apparently on this dataset with various window sizes, where AdaLog improves DeepLog, LogAnomaly, and PLELog by 5569.5%, 5491.7%, and 17.7% on average, respectively.

TABLE IV
PERFORMANCE OF DIFFERENT APPROACHES ON THREE DATASETS. LOGROBUST AND NEURALLOG ARE SUPERVISED, WHILE THE OTHERS ARE SEMI-SUPERVISED.

Model	M	HDFS	BGL			Thunderbird		
		session	20	100	200	20	100	200
Deep Log	P	0.835	0.128	0.166	0.192	0.004	0.017	0.005
	R	0.994	0.995	0.988	0.987	0.938	0.963	1.000
	S	0.994	0.539	0.530	0.528	0.899	0.922	0.005
	F1	0.908	0.227	0.285	0.322	0.008	0.033	0.010
Log Anomaly	P	0.886	0.136	0.176	0.203	0.004	0.025	0.000
	R	0.893	0.970	0.985	0.985	0.938	0.963	1.000
	S	0.961	0.581	0.562	0.559	0.891	0.950	0.005
	F1	0.966	0.239	0.299	0.336	0.008	0.050	0.009
PLE Log	P	0.893	0.592	0.595	0.862	0.429	0.826	0.692
	R	0.979	0.882	0.880	0.844	0.688	0.704	0.360
	S	0.996	0.958	0.968	0.985	1.000	1.000	1.000
	F1	0.934	0.708	0.710	0.853	0.528	0.760	0.474
Log Robust	P	0.961	0.616	0.696	0.684	0.377	0.318	0.289
	R	1.000	0.969	0.968	0.963	0.876	1.000	0.960
	S	0.989	0.959	0.960	0.949	0.999	0.997	0.994
	F1	0.980	0.753	0.810	0.800	0.531	0.482	0.444
Neural Log	P	0.992	0.976	0.975	0.906	0.977	1.000	0.833
	R	0.979	0.979	0.867	0.791	0.935	0.379	0.200
	S	1.000	0.998	0.985	0.991	1.000	1.000	1.000
	F1	0.986	**0.977**	0.918	0.845	**0.956**	0.550	0.323
Ada Log	P	0.988	0.970	0.883	0.875	0.912	1.000	0.818
	R	0.997	0.940	0.961	0.847	0.674	0.621	0.360
	S	1.000	0.998	0.988	0.986	1.000	1.000	1.000
	F1	**0.993**	0.955	**0.920**	**0.861**	0.775	**0.766**	**0.500**

Even though AdaLog is a semi-supervised approach, its performance (F1-score) on all datasets is even **23.8%** and **10.9%** better than the **supervised approaches LogRobust and NeuralLog** on average. On HDFS dataset, AdaLog outperforms LogRobust by 1.3%. On BGL and Thunderbird datasets, AdaLog improves LogRobust by 21.8% and 39.2% for three window sizes, separately. Compared with NeuralLog, even though AdaLog performs worse on BGL and Thunderbird datasets at $ws = 20$, it is on average 31.7% better in other cases and 10.9% better on all datasets. In cases where there is a significant imbalance between the two classes in the training set, AdaLog is found to exhibit suboptimal performance relative to NeuralLog. Adding more anomalous data in the supervised approaches may help alleviate overfitting and improve overall performance.

TABLE V
COMPARISON OF THE EFFECTIVENESS OF DIFFERENT CLUSTERING APPROACHES ON THE TRAINING SET

Cluster	M	HDFS	BGL			Thunderbird		
		session	20	100	200	20	100	200
Self-adaptive	P	0.889	0.815	0.786	0.795	0.049	0.088	0.110
	R	0.973	0.967	0.922	0.893	0.881	0.765	0.717
	S	0.996	0.978	0.971	0.970	0.939	0.915	0.908
	F1	**0.930**	**0.884**	**0.849**	**0.841**	**0.093**	**0.157**	**0.190**
HDBS-CAN	P	0.796	0.784	0.672	0.666	0.015	0.044	0.046
	R	0.898	0.993	0.976	0.892	0.883	0.886	0.583
	S	0.993	0.973	0.944	0.941	0.798	0.795	0.807
	F1	0.844	0.876	0.796	0.762	0.030	0.084	0.085

TABLE VI
COMPARISON OF THE EFFECTIVENESS OF DIFFERENT CLUSTERING METHODS ON THE TEST SET

Cluster	M	HDFS	BGL			Thunderbird		
		session	20	100	200	20	100	200
Self-Adaptive	P	0.982	0.942	0.896	0.929	0.900	0.913	0.500
	R	0.994	0.962	0.866	0.800	0.587	0.724	0.240
	S	1.000	0.996	0.991	0.993	1.000	1.000	1.000
	F1	**0.988**	**0.952**	**0.880**	**0.859**	**0.711**	**0.808**	**0.324**
HDBS-CAN	P	0.985	0.848	0.824	0.212	-	0.010	-
	R	0.848	0.960	0.920	0.858	0.000	0.138	0.000
	S	1.000	0.988	0.982	0.642	1.000	0.980	1.000
	F1	0.912	0.901	0.869	0.340	-	0.019	-

B. Clustering Effectiveness

To compare the clustering algorithms employed in AdaLog and PLELog, we adopt the same methods of preprocessing and generating word embeddings (i.e., BERT). In this experiment, we do not perform undersampling to ensure that differences in comparisons are entirely due to clustering. The results on the training set are demonstrated in Table V, and Table VI indicates the comparison on the test set with the transformer-based classification model for anomaly prediction.

In Table V, we compare our self-adaptive clustering method with HDBSCAN employed in PLELog on the training set. On the whole, our clustering method can more accurately label unlabeled data on all datasets. We can find that both clustering methods perform much better on both HDFS and BGL datasets than Thunderbird dataset. On both datasets, the recall of two clustering methods achieves high values, while the precision calculated by AdaLog is higher than the HDBSCAN adopted by PLELog. On HDFS dataset, our self-adaptive clustering method outperforms HDBSCAN in terms of F1 by 10.2%. The improvements of our clustering method over HDBSCAN by 6.0% on average on BGL dataset. Because of the severe imbalance problem on Thunderbird dataset, two clustering methods have difficulty in correctly classifying log sequences. Overall, on all datasets, our self-adaptive clustering method **outperforms HDBSCAN in the term of F1-score with 64.1%** on average.

In Table V, we directly classify unlabeled data on the training set into normal and abnormal based on label probabilities. Nonetheless, to predict the labels of samples in the test set, we use the Transformer-based classification model for training and validation. Similarly, the results displayed in Table VI also illustrate that our clustering method is far better than HDBSCAN in terms of all evaluation metrics. When combining the HDBSCAN clustering method and the same classification model, the results are not satisfactory. Even on the Thunderbird dataset, the value of TP is 0 (i.e., no anomaly can be correctly predicted), which results in a recall of 0 and makes precision and F1-score invalid values. The reason is that the label probability calculated by HDBSCAN has a large error. Incorporating the classification model, the self-adaptive clustering **outperforms HDBSCAN by 864.1%** (on average) in terms of **F1-score** on three datasets.

C. Undersampling Effectiveness with Different Ratios

In Table VII, we summarize the raw ratios of normal and abnormal data, the undersampling ratio ranges, and the specific undersampling ratios used in AdaLog (i.e., 75% of the range as it is practice-recommended on all three datasets). Table VIII shows the comparison of AdaLog with (75% of the range) or without undersampling. For the sake of rigor, we choose five values within the undersampling ratio ranges to observe the effect of the undersampling ratio on AdaLog's performance. The trend is illustrated in Fig. 2.

TABLE VII
STATISTICS ON THE UNDERSAMPLING RATIO OF DATASETS

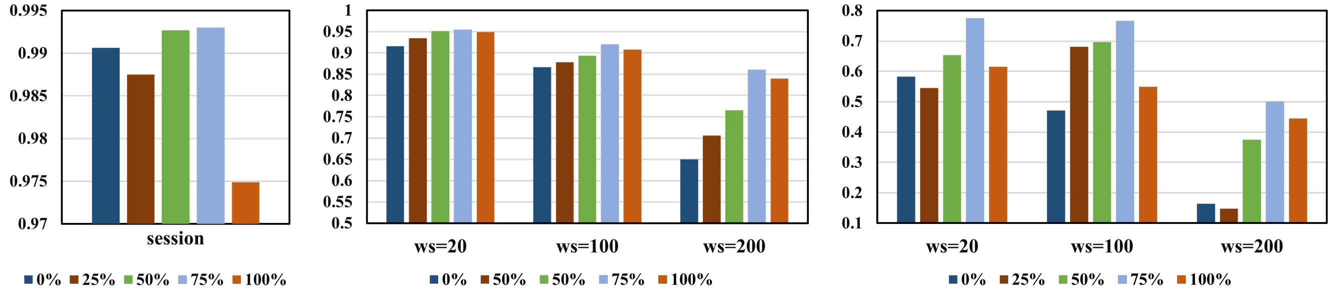
Dataset	Group	Original Ratio	Undersampling Ratio Range	Ratio (75%)
HDFS	session	33.5:1	15:1-30:1	26:1
BGL	20	10.0:1	5:1-9:1	8:1
	100	8.5:1	4:1-8:1	7:1
	200	7.6:1	3:1-7:1	6:1
Thunderbird	20	282:1	140:1-280:1	245:1
	100	93:1	45:1-90:1	79:1
	200	63:1	30:1-60:1	53:1

TABLE VIII
RESULTS OF ADALOG WITHOUT OR WITH UNDERSAMPLING

Ada Log	M	HDFS	BGL			Thunderbird		
		session	20	100	200	20	100	200
W/o	P	0.982	0.942	0.896	0.929	0.900	0.913	0.500
	R	0.994	0.962	0.866	0.800	0.587	0.724	0.240
	S	1.000	0.996	0.991	0.993	1.000	1.000	1.000
	F1	0.988	0.952	0.880	0.859	0.711	**0.808**	0.324
W/h	P	0.988	0.970	0.883	0.875	0.912	1.000	0.818
	R	0.997	0.940	0.961	0.847	0.674	0.621	0.360
	S	1.000	0.998	0.988	0.986	1.000	1.000	1.000
	F1	**0.993**	**0.955**	**0.920**	**0.861**	**0.775**	0.766	**0.500**

The results shown in Table VIII demonstrate that the performance with undersampling outperforms without undersampling except for only one case (i.e., on Thunderbird dataset with $ws = 100$). The possible reason is that the percentage of anomalies in the training set is much larger than that in the test set, and the use of undersampling may enlarge the difference between the two and lead to a little deviation in prediction. Overall, compared with the model without undersampling, the one with undersampling improves the performance on average by 0.5% (HDFS), 1.7% (BGL), and 19.4% (Thunderbird), respectively.

To explore whether this chosen undersampling ratio is reasonable for each dataset, we conduct experiments on the



(a) HDFS

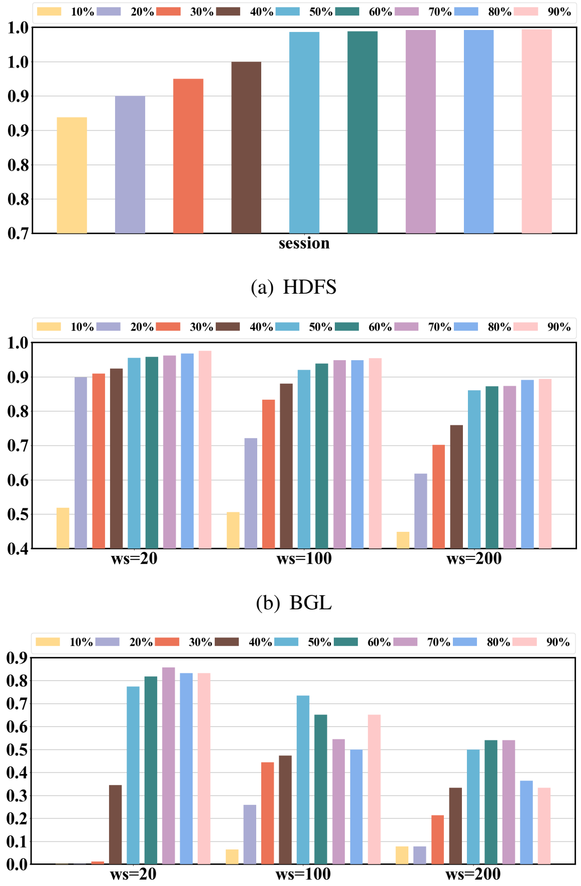
(b) BGL

(c) Thunderbird

Fig. 2. The effect of the undersampling ratio on AdaLog's performance in F1-score. The percentages 0% and 100% indicate the minimum and maximum values of the undersampling range.

undersampling ratio within a set range shown in Table VIII. We take different undersampling values, i.e., 0% (minimum), 25%, 50%, 75%, and 100% (maximum), as the undersampling ratio. The performance trend is illustrated in Fig. 2. The horizontal and vertical axis refers to the percentages of the ratio range and the performance in terms of F1-score, respectively. On the HDFS dataset, performance drops slowly from 0.991 (min) to 0.992 (25%), and rises to 0.993 (75%), then drops sharply to 0.975 at a 30:1 ratio of the two classes (max). Unlike HDFS, the performance of the BGL dataset shows a steady upward trend until reaching 75% of the ratio range. The up and down trends are flatter when the window size is 20 or 100, and the corresponding ranges of the variation are 0.033 and 0.054, respectively. In comparison, the change (i.e., 0.211) is more significant in the case of $ws = 200$. The performance climbed from 0.650 (min) to 0.765 (50%), followed by a sharp rise to 0.861 (75%), and then a slow decline to 0.840 (max).

On Thunderbird, the most imbalanced dataset, the variation range widens to 0.352 ($ws = 200$). When the window size is 20 or 200, the trend of change is similar. There is a modest decrease and then a considerable increase, followed by a minor decay. At the peak points, the percentage of the ratio range is 75%. When $ws = 100$, the performance continues to rise to 0.766 as the percentage increases from min to 75%, after which the performance decreases to 0.549 (max). To sum up, for any dataset, within the range of undersampling ratios, 75% is a very reasonable undersampling ratio.



(a) HDFS

(b) BGL

(c) Thunderbird

Fig. 3. The effect of the labeled training data size on AdaLog's performance in F1-score.

D. Overall Effectiveness with Different Labeled Ratios

As a semi-supervised approach, AdaLog utilizes a part of labeled training data to predict the unknown ones. The operation benefits the model will be less sensitive to the data, thereby keeping stability. Fig. 3 represents how the performance of the proposed AdaLog in F1-score changes when the labeled proportion (represented as lp) ranges (i.e., 10% to 90% in 10 percentages).

On HDFS dataset, the performance of AdaLog continues to rise steadily, from 0.869 to 0.997. Especially when lp increases from 50% to 90%, the performance gap of AdaLog is tiny, and the improvement is only 0.4%. This result represents that AdaLog on HDFS dataset maintains outstanding performance even with the proportion of labeled data changes. On BGL dataset, the changing trend of AdaLog's performance is relatively consistent in different window sizes. The performance change of AdaLog under different window sizes is basically a sharp improvement first (especially the lp changes from 10% to 20%) and then a slow increase (i.e., 50%–90%). When $ws = 20$, the F1-score dramatically increases from 0.519 to 0.899 (an improvement of 73.2%) as lp changes from 10% to 20%, then gradually grows to 0.955 when lp is 50%. Similarly, under $ws = 100$, the F1-score has an improvement of 42.5% with the changes of lp from 10% to

20%. Afterward, this value rises by 27.6% until lp increases to 50%. When $ws = 200$, the F1-score greatly improves from 0.449 to 0.861 (an improvement of 91.8%) with the change in lp from 10% to 50%. When the labeled ratio changes from 50% to 90%, the performance of AdaLog is slightly improved by 2.2% ($ws = 20$), 3.7% ($ws = 100$), and 3.8% ($ws = 200$), respectively. The experimental results demonstrate that as the amount of labeled data increases, the performance of AdaLog on BGL dataset also steadily improves. However, when the proportion of labeled data changes from 50% to 90%, the performance does not change much.

AdaLog has an unusual behavior on the Thunderbird dataset compared to the other two datasets. In Table IV, supervised methods generally perform better than semi-supervised methods since the former use more semantic information for training. Then on the Thunderbird dataset, due to the severe imbalanced problem, supervised methods will likely overfit anomalies. Once some anomalies that the model has not seen appear, the model is likely to predict them as normal logs. Fig. 3(c) clarifies this point intuitively. When the window size is 20, the performance of AdaLog spikes from 0.001 ($lp = 10\%$) to 0.775 ($lp = 50\%$), then slowly rises to 0.857 ($lp = 70\%$) before dropping slightly to 0.833 ($lp = 90\%$). When the window sizes are 100 and 200, the performances of AdaLog change significantly. They rise sharply to 0.735 ($lp = 50\%$) and 0.541 ($lp = 60\%$), respectively, then drop magnificently. The experimental findings suggest that augmenting the amount of training data is not invariably conducive to enhanced performance, and the data category should be considered. In instances of pronounced class imbalance, such a practice may engender training drift, yielding diminished recall and F1-score. In summary, AdaLog performs well as a semi-supervised method with 50% labeled training data, even better with extreme imbalance.

VI. Conclusion and Future Works

We propose an efficient semi-supervised method AdaLog to enhance the performance of anomaly detection. To improve the accuracy of data labeling, AdaLog utilizes a self-adaptive clustering method to calculate the label probability of unlabeled data more accurately, which considers twelve specially-designed situations based on the distance of labeled data. To alleviate the class imbalance problem, AdaLog employs an undersampling method to improve the performance of AdaLog. Furthermore, we adopt a pre-trained model for word embedding and a Transformer-based model for prediction. Experimental results show that AdaLog achieves superior performance than five SOTA semi-supervised and supervised approaches for industrial log-based anomaly detection.

Studies [31]–[33] have pointed out that data-driven deep learning models are vulnerable to perturbations in input data, making them susceptible to adversarial attacks. Although leveraging the transformer, a deep learning model, for anomaly prediction, it is noteworthy that AdaLog is semi-supervised and employs a self-adaptive clustering technique to estimate label probabilities on unlabeled data. This enables subsequent training steps to be less sensitive and less prone to deviation. However, to further enhance the robustness of the proposed approach, it is imperative to integrate advanced techniques, such as adversarial ones, to resist external attacks. Incorporating adversarial training or defense mechanisms, as well as techniques like input perturbation, may be promising directions to explore in future work. Strengthening the overall effectiveness and resilience of AdaLog would allow for its expansion to other applications within the IIoT domain.

References

[1] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Integrated generative model for industrial anomaly detection via bidirectional lstm and attention mechanism," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 541–550, 2021.

[2] M. Cinque, C. Esposito, and A. Pecchia, "Security log analysis in critical industrial systems exploiting game theoretic feature selection and evidence combination," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3871–3880, 2019.

[3] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, "Lstm learning with bayesian and gaussian processing for anomaly detection in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5244–5253, 2019.

[4] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for iot time-series data: A survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[5] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 492–504.

[6] V.-H. Le and H. Zhang, "Log-based anomaly detection with deep learning: How far are we?" in *Proceedings of the 44th International Conference on Software Engineering (ICSE)*. IEEE, 2022, pp. 1356–1367.

[7] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–37, 2021.

[8] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *et al.*, "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 (ESEC/SIGSOFT FSE)*, 2019, pp. 807–817.

[9] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1285–1298.

[10] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun *et al.*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 4739–4745.

[11] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "Semi-supervised log-based anomaly detection via probabilistic label estimation," in *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1448–1460.

[12] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE)*. IEEE, 2016, pp. 102–111.

[13] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP)*, 2009, pp. 117–132.

[14] M. Wurzenberger, F. Skopik, M. Landauer, P. Greitbauer, R. Fiedler, and W. Kastner, "Incremental clustering for semi-supervised anomaly detection applied on log data," in *Proceedings of the 12th International Conference on Availability, Reliability and Security (ARES)*, 2017, pp. 1–6.

[15] F. Liu and Y. Deng, "Determine the number of unknown targets in open world based on elbow method," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 5, pp. 986–995, 2020.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[18] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2007, pp. 575–584.

[19] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in ibm bluegene/l event logs," in *7th IEEE International Conference on Data Mining (ICDM)*. IEEE, 2007, pp. 583–588.

[20] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *1st IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 2004, pp. 36–43.

[21] M. Farshchi, J.-G. Schneider, I. Weber, and J. Grundy, "Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2015, pp. 24–34.

[22] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, "Automated it system failure prediction: A deep learning approach," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 1291–1300.

[23] D. Wu, H. Xu, Z. Jiang, W. Yu, X. Wei, and J. Lu, "Edgelstm: Towards deep and sequential edge computing for iot applications," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1895–1908, 2021.

[24] D. Wu, Q. Guan, Z. Fan, H. Deng, and T. Wu, "Automl with parallel genetic algorithm for fast hyperparameters optimization in efficient iot time series prediction," *IEEE Transactions on Industrial Informatics*, 2022.

[25] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[26] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002.

[27] A. Raganato and J. Tiedemann, "An analysis of encoder representations in transformer-based machine translation," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. The Association for Computational Linguistics, 2018.

[28] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP/IJCNLP)*, 2019, pp. 3728–3738.

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations (ICLR), Poster*, 2019.

[30] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext. zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

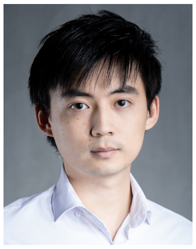
[31] S. Lu, M. Wang, D. Wang, X. Wei, S. Xiao, Z. Wang, N. Han, and L. Wang, "Black-box attacks against log anomaly detection with adversarial examples," *Information Sciences*, vol. 619, pp. 249–262, 2023.

[32] A. Goodge, B. Hooi, S.-K. Ng, and W. S. Ng, "Robustness of autoencoders for anomaly detection under adversarial impact." in *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 1244–1250.

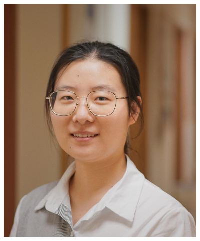
[33] F. Liu, H. Liu, and W. Jiang, "Practical adversarial attacks on spatiotemporal traffic forecasting models," in *Proceedings of the 36th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022, pp. 1–18.

Xiaoxue Ma received the B.S. (Hons) degree in e-commerce engineering with law from Beijing University of Posts and Telecommunications, Beijing, China, in 2018, and the M.S. degree in Information Technology from the Hong Kong University of Science and Technology, Hong Kong, China, in 2019. She is currently working toward the Ph.D. degree in computer science at the City University of Hong Kong while serving as a visiting researcher at University College London. Her research interests mainly focus on designing self-defined algorithms and leveraging deep learning techniques to extract and analyse meaningful textual information in software engineering domains, such as bug report summarization, log-based anomaly analysis.

Jacky Keung received the B.S. (Hons) degree in computer science from the University of Sydney, and the Ph.D. degree in software engineering from the University of New South Wales, Australia. He is an Associate Professor at the Department of Computer Science, City University of Hong Kong. He has authored papers in prestigious journals, including IEEE Transactions on Software Engineering, ACM Transactions on Software Engineering and Methodology, and many other leading journals and conferences. His research interests include software engineering, Artificial intelligence, Machine Learning and Data Science. He is a Senior Member of the IEEE, and Chairman IEEE Hong Kong Section Computer Society Chapter.



Pinjia He is an assistant professor of The Chinese University of Hong Kong, Shenzhen. He has been a postdoctoral researcher in the Computer Science Department at ETH Zurich after receiving his Ph.D. degree from the Chinese University of Hong Kong in 2018. His main research interests include software engineering, software testing, software security, AIOps, and Trustworthy AI. His research appeared in top computer science venues, such as ICSE, ESEC/FSE, ASE, ISSTA, and OSDI. He received the first IEEE Open Software Services Award and an ISSRE Most Influential Paper Award. The LogPAI project led by him has been starred 3,000+ times on GitHub and downloaded 90,000+ times.



Yan Xiao is an Associate Professor at School of Cyber Science and Technology in Sun Yat-sen University. She received her Ph.D. degree from the City University of Hong Kong and held a research fellow position at National University of Singapore. Her current research focus is evaluating the trustworthiness of deep learning systems including trustworthiness evaluation for autonomous driving, deep neural network repair for improving the accuracy of models, input discrimination for deep learning systems. Her research is within the domain of Software Engineering and Artificial Intelligence in application domains related to software bug localization, software code readability, software integration test, data mining, and big data analysis. Her research is undertaken in a broad spectrum of application domains, such as self-driving cars, deep neural network analysis, and hydrological data analysis. More information is available on her homepage: https://yanxiao6.github.io/.

Xiao Yu received his Ph.D. degree from the School of Computer Science at Wuhan University in 2020. He also earned a joint Ph.D. degree from the Department of Computer Science at City University of Hong Kong in 2021. He has authored or co-authored numerous research papers in some international journals such as ACM Transactions on Software Engineering and Methodology, IEEE Transactions on Reliability, IEEE Transactions on Emerging Topics in Computing, Journal of Systems and Software, and Information and Software Technology. His research interests primarily focus on utilizing data mining and deep learning techniques to extract meaningful insights from vast data repositories available in software engineering domains.

Yishu Li is a Ph.D. candidate in the Department of Computer Science, City University of Hong Kong. She received an M.S. degree in Creative Technology from the University of Bristol. She has worked as the requirement engineer in a global bank and a start-up. Her current research is on software Requirements Engineering, focusing on establishing artifacts to guide the elaboration of requirements for AI-based and FinTech systems. She is exploring how natural language processing techniques as well as machine learning approaches can assist with industrial RE practices in the software development process. More specifically, she concentrates on requirement-related case studies with real-world context.