2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS) | 979-8-3503-1958-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/QRS60937.2023.00015

LogContrast: A Weakly Supervised Anomaly Detection Method Leveraging Contrastive Learning

Ting Zhang[1], Xin Huang[2,*], Wen Zhao[3], Guozhao Mo[4], and Shaohuang Bian[4]
[1]School of Software and Microelectronics, Peking University, Beijing, China
[2]School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing, China
[3]National Engineering Research Center for Software Engineering, Peking University, Beijing, China
[4]College of Information and Electrical Engineering, China Agricultural University, Beijing, China
{zhangting2019, zhaowen}@pku.edu.cn, huangxin@bit.edu.cn, {moguozhao, bianshaohuang}@cau.edu.cn
*corresponding author

Abstract—We propose a novel log-based anomaly detection method that leverages weakly supervised contrastive learning, named LogContrast. LogContrast aims to address the issues of limited and noisy labeled logs in real-world scenarios. During the training stage, LogContrast first augments the current log feature through dropout. Subsequently, it treats the current log feature and the augmented feature as a positive pair, while treating the current log feature and other log features in the same batch as negative pairs. The objective is to pull the positive pairs closer together and push the negative pairs farther apart, thereby encouraging similar logs to be closer to each other and dissimilar logs to be farther apart in the feature space. The experimental results demonstrate the excellent performance of LogContrast even with limited labeled logs and greater noise resistance compared to fully supervised methods. In addition, we explore the role of semantic features and demonstrate that semantic features have strong adaptability to constantly evolving logs.

Keywords–system logs; anomaly detection; weakly supervised learning; contrastive learning

1. Introduction

System logs are served as a crucial source of information for monitoring hardware and software issues within a system [1]. They provide runtime information about the system, including its state and any events currently in progress [2]. Therefore, when a system failure occurs, logs contain pertinent information about the anomaly. Operating personnel can query logs to obtain information about the anomaly, trace the source of the problem and resolve it. For instance, suppose a log entry containing the message ”rts: kernel terminated for reason 1004” is generated, operating personnel can infer from the information ”kernel terminated” that the system has experienced a kernel interrupt anomaly.

With the development of IT services, system components have become increasingly large-scale, resulting in an explosion of log data [3]. For example, Google generates millions of new log entries every month, resulting in terabytes of log data generated every day. Manual investigation of log data for anomalies is inefficient and impractical [4]. Moreover, individual developers or system administrators may not possess adequate knowledge of the entire system, particularly as many large enterprise systems often use commercial off-the-shelf components (such as third-party components). Artificial Intelligence for IT Operations (AIOps) aims to enhance IT operations capabilities through advanced AI algorithms to address these challenges and further maintain the stability of enterprise services [5]. With the development of machine learning, machine learning-based log anomaly detection methods have been proposed to automatically assist operating personnel locate in locating and detecting anomalies in logs.

Over the past few decades, many machine learning-based log anomaly detection methods have been proposed. Some methods use log event counts as quantitative input and project the event count vector into a vector space using traditional machine learning techniques. Those vectors that deviate from most or violate some invariant relationships between event counts are classified as anomalies. Representative methods of traditional machine learning include logistic regression [6], one-class SVM [7], LogClustering [8], and invariants mining [9]. However, these traditional machine learning-based methods often take quantitative or statistical data as features and cannot capture the rich semantic information in log texts and the sequential relationships between log event sequences, resulting in unstable performance on different datasets [10]. Therefore, with the development of deep learning, many deep neural network-based log anomaly detection methods have been proposed. These deep learning-based anomaly detection methods can better represent higher-dimensional features in logs and thus help operating personnel locate and detect anomalies more accurately. Examples include DeepLog [11], LogAnomaly [12], LogRobust [10], and LogBERT [13]. However, we observe that existing deep learning-based log anomaly detection methods still have some limitations:

- The superior performance of deep learning heavily relies on supervised training with sufficient labeled data [14]. However, annotating large-scale data can be a daunting task that requires extensive expert knowledge and is both labor-intensive and time-consuming [15]. Consequently, obtaining labeled logs in real-world scenarios is scarce, which poses difficulties in scaling up to large datasets.
- The presence of noisy labels (i.e., incorrectly labeled data)

979-8-3503-1958-3/23/$31.00 ©2023 IEEE
DOI 10.1109/QRS60937.2023.00015

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

can significantly impact deep learning-based log anomaly detection approaches [16]. These noisy labels in the training set can degrade the performance of the model trained with them, thereby resulting in poor performance during practical implementation. In addition, the presence of noisy labels in the testing/validation set can lead to incorrect evaluations.

- Developers may frequently modify source code, including log statements, due to constant updates and modifications to a system [17]. Suhas Kabinna et al. observed that in the projects they studied, approximately 20% to 45% of log statements changed over the entire lifecycle [18]. Some modifications to log output statements may be minor, for example, developers change the log output statement "Logger.info('Receiving block {block_id} src: /{src_ip} dest: /{dest_ip}')" to "Logger.info('Receiving block {block_id} source: /{src_ip} destination: /{dest_ip}')" to express more clearly. However, even minor updates to the log text can cause the log parser to identify it as a new log template and create a new log key, as shown in Fig.1. When this happens, the models that rely solely on log key sequences may encounter out-of-vocabulary problem, and misclassify previously unseen log key as anomaly, resulting in false alarm.

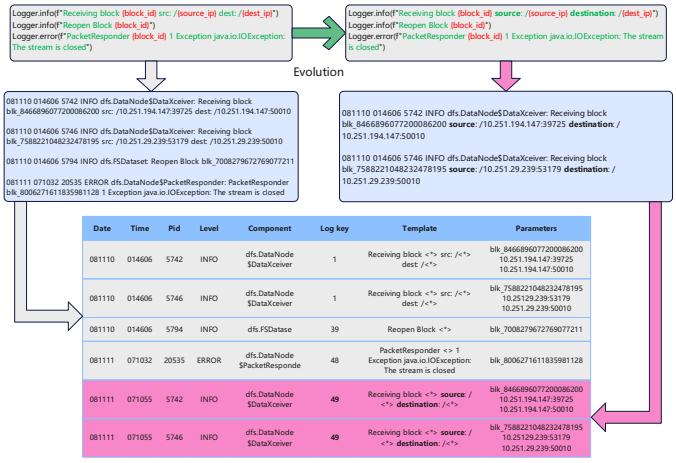


Figure 1. The evolution of logs by updating log output statement and its impact on log parsing.

To address the issues mentioned above, we propose a novel anomaly detection method based on weakly supervised contrastive learning. LogContrast mainly adopts self-supervised contrastive learning and utilizes a small number of labels for supervised learning. By using a small number of labeled samples for weakly supervised contrastive learning, LogContrast mitigates the scarcity and inaccuracies of log labels in real-world scenarios.

The main contributions of this study are as follows:

- We propose LogContrast, an anomaly detection method that leverages weakly supervised contrastive learning to address the problems of scarce and erroneous log labels in reality. This approach only uses a portion of the labels for supervised training, thus reducing the training bias caused by noisy log labels.
- We design a log semantic feature extractor (LSFE) and a log key sequential feature extractor (LKFE) in LogContrast, which more comprehensively extract features from system logs, making the approach adaptable to log modifications or evolutions.
- We investigate the roles of semantic and log key sequential features in anomaly detection tasks and provide visualizations to illustrate the characteristics of these two types of features.

2. Preliminary

2.1 Log

Logs are unstructured data which record the events and states of a system during runtime. Log parsing is the process of converting semi-structured log text into structured log templates, which retain the constants and replace the variables with placeholders (i.e., <*>) [19]. Each log corresponds to a unique log key. Log grouping involves splitting a chronological sequence of logs using fixed, sliding, or session windows into many subsequences [20]. Each log subsequence is represented by $\boldsymbol{S}_L = [L_1, L_2, ...]$, with its corresponding log template sequence $\boldsymbol{S}_{LT} = [LT_1, LT_2, ...]$ and log key sequence $\boldsymbol{S}_{LK} = [LK_1, LK_2, ...]$ obtained by log parsing.

2.2 Weakly Supervised Learning

Weakly supervised learning, also called semi-supervised learning, aims to alleviate the issue of having limited amounts of labeled data available for training. Zhou [21] illustrates three typical types of weak supervision: incomplete, inexact and inaccurate supervision.

- Incomplete supervision concerns about the situation where only part of the input data is labeled.
- Inexact supervision happens when the detected labels are not as exact as they need to be. For example, only coarse-grained label information is available due to the labeling cost.
- Inaccurate supervision refers to the problem where the labels are corrupted with noise, which may be caused by labeling errors and machine mistakes.

These three scenarios are commonly encountered in log-based anomaly detection tasks. For instance, in large-scale systems, only a portion of the log data may be labeled, and even among the labeled data, some may be incorrectly labeled. Furthermore, instead of labeling individual log entries, there are cases where logs are labeled at the granularity of log sequences.

2.3 Contrastive Learning

Contrastive learning can be performed with supervised [22] or self-supervised method [23]–[25]. Supervised contrastive learning relies on the labels of data. It treats the data with the same label as positive pairs, and data with different labels as negative pairs. Self-supervised contrastive learning does not require labels, it treats the original data and the data

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

augmented from the original data as positive pairs, and the views of other data as negative pairs. By training model with contrastive loss, positive pairs are closer and negative pairs are farther in the feature space, to achieve alignment of positive pairs and uniformity of negative pairs on the hypersphere space [26]. We hope to combine self-supervised contrastive method with supervised learning using a small number of labeled logs to implement weakly supervised learning, thereby training models that can achieve better detection performance with limited labeled logs.

3. METHODOLOGY

3.1 Overview

Given training sets $\mathcal{D}_{train} = \{(\boldsymbol{S}_L^{(i)}, y^{(i)})\}_{i=1}^{M}$ and testing set $\mathcal{D}_{test} = \{(\boldsymbol{S}_L^{(i)}, y^{(i)})\}_{i=1}^{N}$, we aim to develop a model D_θ, which uses training set to refine its parameters θ. In training set, label $y^{(i)}$ can be 0, 1 or -1 (i.e., $y^{(i)} \in \{0, 1, -1\}$), whereas in testing set, it can only be 0 or 1 (i.e., $y^{(i)} \in \{0, 1\}$), 0 represents log sequence is normal, 1 represents abnormal and -1 indicates unlabeled. During testing stage, the trained model D_θ determines whether a given log sequence $\boldsymbol{S}_L^{(i)}$) in the test set is anomalous (i.e., $D_\theta(\boldsymbol{S}_L^{(i)}) \rightarrow \{0, 1\}$). The objective is to optimize θ such that D_θ can accurately detect anomalies in the testing set.

We design a log parser, a log semantic feature extractor (LSFE) and a log key feature extractor (LKFE) in LogContrast. The overview architecture of LogContrast is shown in Fig.2.

3.2 Feature Extraction and Fusion

LogContrast utilizes log parser to obtain log template sequences, log key sequences, and labels for both training set and testing set. The log parsing process, shown in Fig.1, is commonly performed using methods such as Spell [27] and Drain [28]. However, since this paper does not aim to improve the log parsing method, it will not be discussed in further detail.

LogContrast includes a LSFE and a LKFE to extract features from log template sequences and log key sequences, respectively, in order to capture log features as comprehensively as possible. The motivation for designing these two extractors will be discussed below.

- **Semantic feature.** The most critical aspect of a log is its semantic feature, which provide valuable information about the log text. Phrases such as "error", "shutdown" or "fault" are indicative of abnormal logs. Pre-trained language models such as GPT-3 [29] and BERT [30] can effectively learn the semantic representations from text, and their robustness enables them to capture contextual information from log text, even if the log statements are continually modified. Therefore, LogContrast uses BERT to extract the semantic features from logs. Only the output layer parameters are fine-tuned, and the BERT model's other parameters are frozen to avoid deteriorating the generalization ability.
- **Log key sequence feature.** It is essential for representing the execution of log events. If there is a temporal relationship between two events, such as EventA $\rightarrow$ EventB representing normal behavior and EventB $\rightarrow$ EventA indicating abnormal behavior, it is referred to as a context anomaly, which can be detected using log key sequence features. LSTM [31] is commonly used to extract these features, but its training is serial, making it computationally inefficient. Transformer [32] has recently become popular for its parallelism and high computational efficiency and has shown excellent performance in log anomaly detection tasks, such as NeuralLog [2], Logsy [33] and LogBERT [13]. A Transformer encoder is used as the log key feature extractor, with trainable positional encoding and batch normalization replacing layer normalization [34], making it more suitable for sequential modeling. The LKFE based on the Transformer encoder is shown in Fig.3

The log key sequence $\boldsymbol{S}_{LK} = [LK_1, LK_2, ...]$ is first passed through an embedding layer, which acts like a lookup table, and generates an embedding representation $\boldsymbol{X}^{(0)} = \mathrm{Embedding}(\boldsymbol{S}_{LK})$. The positional encoding is then applied to each log key, but unlike the original Transformer encoder, we use a learnable positional encoding that is trained during the training stage. The positional information is added to the embedding and passed through the Transformer encoder, which is characterized by stacking the multi-head self-attention (MSA) and feed forward layers. Assuming that there are L layers, the embedded log key sequence $\boldsymbol{X}^{(0)}$ is input into the first layer and the output of the l-th layer is calculated as

$$\boldsymbol{Z}^{(l)} = \mathrm{Batchnorm}(\mathrm{Attention}(\boldsymbol{X}^{(l-1)}) + \boldsymbol{X}^{(l-1)}), \quad (1)$$

$$\boldsymbol{X}^{(l)} = \mathrm{Batchnorm}(\mathrm{Feedforward}(\boldsymbol{Z}^{(l)}) + \boldsymbol{Z}^{(l)}), \quad (2)$$

where $\boldsymbol{Z}^{(l)} \in \mathbb{R}^{n \times d_{model}}$ and $\boldsymbol{X}^{(l)} \in \mathbb{R}^{n \times d_{model}}$ represent the latent representation and the output of the l-th layer, respectively. For each layer, $\boldsymbol{W}_Q^{(l)}$, $\boldsymbol{W}_K^{(l)}$ and $\boldsymbol{W}_V^{(l)} \in \mathbb{R}^{d_{model} \times d_{model}}$ are created, representing the parameter matrices of query, key and value, respectively. The multiplication of $\boldsymbol{X}^{(l-1)}$ with these three matrices results in the $\boldsymbol{Q}^{(l)}$, $\boldsymbol{K}^{(l)}$ and $\boldsymbol{V}^{(l)} \in \mathbb{R}^{n \times d_{model}}$, i.e.,

$$\boldsymbol{Q}^{(l)} = \boldsymbol{X}^{(l-1)} \boldsymbol{W}_Q^{(l)}, \quad (3)$$

$$\boldsymbol{K}^{(l)} = \boldsymbol{X}^{(l-1)} \boldsymbol{W}_K^{(l)}, \quad (4)$$

$$\boldsymbol{V}^{(l)} = \boldsymbol{X}^{(l-1)} \boldsymbol{W}_V^{(l)}, \quad (5)$$

and calculating self-attention as

$$\mathrm{Attention}(\boldsymbol{X}^{(l-1)}) = \mathrm{softmax}(\frac{\boldsymbol{Q}^{(l)} {\boldsymbol{K}^{(l)}}^T}{\sqrt{d_{model}}}) \boldsymbol{V}^{(l)}. \quad (6)$$

Assuming the MSA with H heads, $\boldsymbol{Q}_h^{(l)}$, $\boldsymbol{K}_h^{(l)}$ and $\boldsymbol{V}_h^{(l)}$ denote the query, key and value of the h-th head, respectively. The output of $\boldsymbol{Z}^{(l)} \in \mathbb{R}^{n \times d_{model}}$ is obtained by concatenating $\{\boldsymbol{Z}_h^{(l)} \in \mathbb{R}^{n \times \frac{d_{model}}{H}}\}_{h=1}^{H}$ from multiple heads.

The feed-forward layer comprises two linear layers and an activation function Gaussian error linear unit (GELU) which can be expressed as

$$\mathrm{Feedforward}(\boldsymbol{Z}^{(l)}) = \mathrm{Linear}(\mathrm{GELU}(\mathrm{Linear}(\boldsymbol{Z}^{(l)}))) \quad (7)$$

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

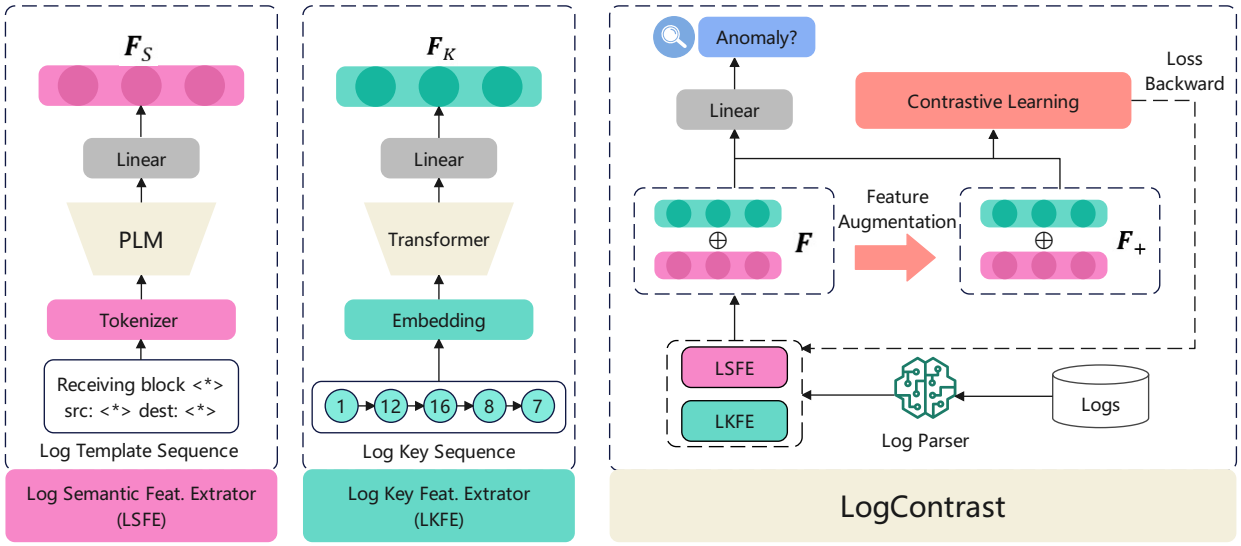


Figure 2. The overview architecture of LogContrast.

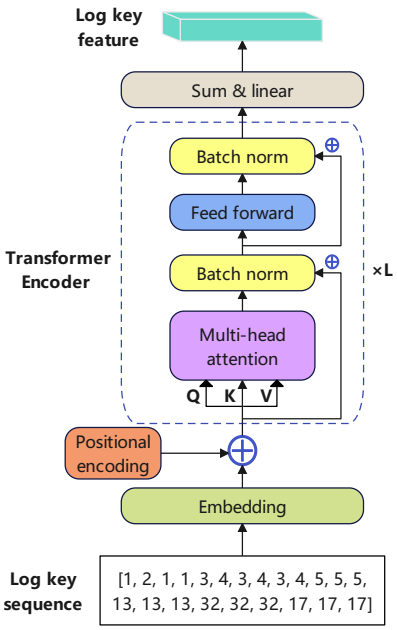


Figure 3. The LKFE based on the Transformer encoder.

The final output of the Transformer encoder is denoted as $\boldsymbol{X}^{(L)} \in \mathbb{R}^{n \times d_{model}}$. By taking the sum of the first dimension of $\boldsymbol{X}^{(L)}$, a linear layer is then applied to map $\boldsymbol{X}^{(L)}$ from dimension d_{model} to d_{feat} and obtain the feature representation $\boldsymbol{F}_K \in \mathbb{R}^{d_{feat}}$ which is the final output of the LKFE, i.e.,

$$\boldsymbol{F}_K = \text{Linear}(\text{Sum}(\boldsymbol{X}^{(L)}, \text{dim} = 1)). \tag{8}$$

Once the log text and log key sequence undergo feature extraction, LogContrast obtains the feature representations $\boldsymbol{F}_S$ and $\boldsymbol{F}_K$ for log semantics and log key sequence, respectively. These features are then concatenated to form the fusion feature $\boldsymbol{F} = \boldsymbol{F}_S \oplus \boldsymbol{F}_K$. Next, LogContrast augments the fusion feature and conducts contrastive learning.

3.3 Feature Augmentation and Contrastive Learning

LogContrast uses dropout [35] which is a technique that randomly close some neuron connections in a neural network for feature augmentation during training. This introduces variability in the model's output if different neurons are closed. Log updates and evolution can result in perturbations in the extracted features, as log text may undergo token deletion, addition and replacement, and new log keys may be generated. The use of dropout during training can simulate these changes, and make the model more adaptable to continuously evolving logs. Specifically, during training stage, LogContrast applies dropout to a fused feature $\boldsymbol{F}$ to obtain an augmented feature $\boldsymbol{F}_+$, which is a positive instance for $\boldsymbol{F}$. Other fused features in the current batch are negative instances for $\boldsymbol{F}$ as they belong to different views. The contrastive learning process is illustrated in Fig.4.

Contrastive learning aims to minimize the distance between similar samples (positive pairs) and maximize the distance between dissimilar samples (negative pairs) in the feature space [23], [24]. This helps to ensure that the feature representations of normal and anomalous log sequences are well separated in the feature space. LogContrast applies the InfoNCE loss [23] for contrastive learning on a batch of features. The formula for computing contrastive loss is

$$\mathcal{L}_{cl} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{e^{\text{sim}(\boldsymbol{F}^{(i)}, \boldsymbol{F}_+^{(i)})/\tau}}{\sum_{j=1}^{B} e^{\text{sim}(\boldsymbol{F}^{(i)}, \boldsymbol{F}^{(j)})/\tau}}. \tag{9}$$

In Eq.(9), $\text{sim}(\boldsymbol{F}^{(i)}, \boldsymbol{F}_+^{(i)})$ denotes the cosine similarity between feature vectors $\boldsymbol{F}^{(i)}$ and $\boldsymbol{F}_+^{(i)}$, which measures the similarity between positive pairs. B represents the batch size. The denominator of the formula sums over all feature pairs in the batch, including both positive and negative pairs, and

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

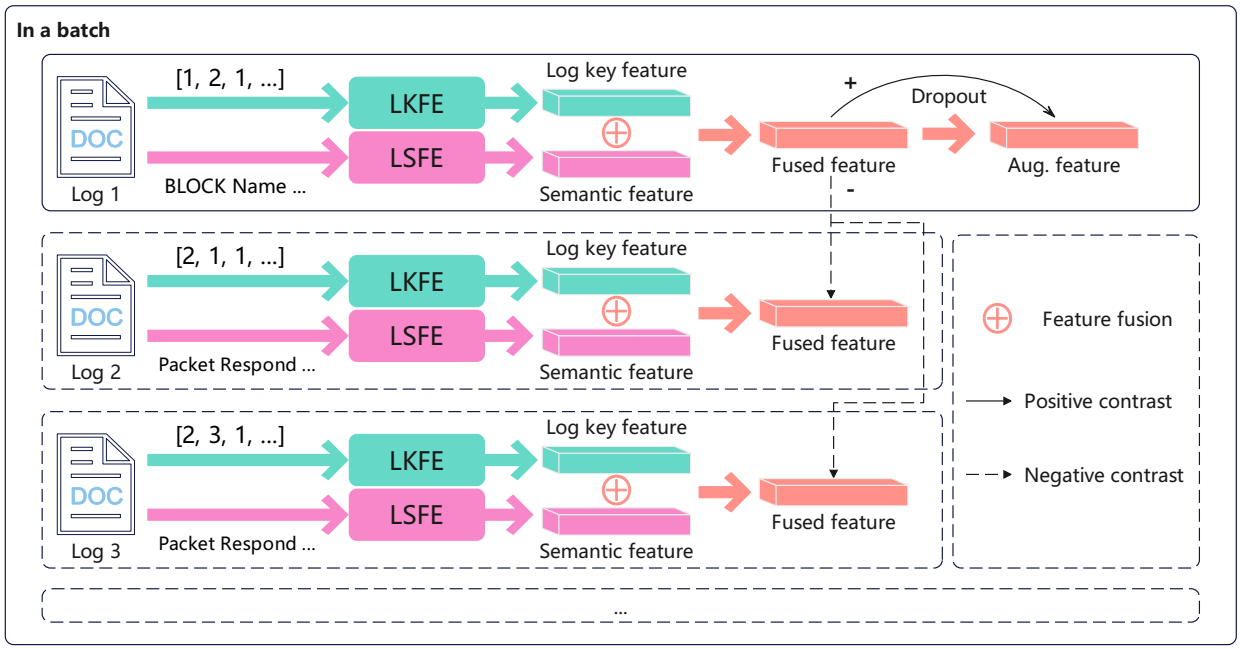


Figure 4. The process of contrastive learning.

represents the total cosine similarity between them. τ is a temperature hyperparameter.

Self-supervised contrastive learning is reasonable, but there is a scenario where two log sequences within the same batch are identical, resulting in treating them as negative pair instead of positive one. To address this issue, a few labeled data can be used to mitigate this issue by employing cross-entropy loss for supervised learning. The formula for calculating cross-entropy loss is

$$\mathcal{L}_{ce} = -\frac{1}{B}\sum_{i=1}^{B}[y^{(i)}\log p^{(i)} + (1-y^{(i)})\log(1-p^{(i)})], \quad (10)$$

where $y^{(i)}$ is the label of the i-th log sequence in the batch, $p^{(i)}$ is the probability that the sequence is predicted to be anomalous by LogContrast, which is obtained by applying a linear layer and softmax to the feature $F^{(i)}$.

The loss function of LogContrast is ultimately formulated as weakly supervised, which can be computed as the following formula:

$$\mathcal{L} = \mathcal{L}_{ce} + \lambda_{cl}\mathcal{L}_{cl}. \quad (11)$$

3.4 Anomaly Detection

During the testing stage, LogContrast relies on an anomaly threshold p_{th} obtained during training. When the predicted probability p of a log sequence being classified as an anomaly is greater or equal to p_{th}, the log sequence is considered as anomaly. To determine the anomaly threshold p_{th}, we utilize the predicted anomaly probabilities $\{p^{(i)}\}_{i=1}^{M}$ for all log sequences during model training. These probabilities are served as candidate values for the threshold. We select the probability that maximizes the F1 score (as described in Section 4-B) from the candidate values and save it as the threshold p_{th}, which is used for anomaly detection during testing.

4. Experiments

4.1 Datasets

Loghub [36] has open-sourced several real-world log datasets, including distributed system, supercomputer system and operating system log datasets. We conducted experiments to evaluate the effectiveness of LogContrast using two widely-used log datasets in Loghub, namely HDFS and BGL.

The HDFS dataset [37] comprises 11,175,629 log entries generated by MapReduce jobs running on 203 Amazon EC2 nodes. Roughly 2.9% of the logs are anomalous, and we can obtain 575,061 log sequences by grouping them with session windows based on identifier *block_id*.

The BGL dataset [38] is collected from the BlueGene/L supercomputer system at LLNL in Livermore, California. It consists of 4,747,963 log entries, with 348,460 being anomalous. Anomalous logs have fine-grained anomaly labels and are widely used for log anomaly detection. We used fixed windows to group logs since BGL dataset lacks identifier similar to *block_id* [20].

We use the entire datasets as the testing sets, and randomly select 10,000 logs to build the training sets. They are imbalanced real-world log datasets, with anomalous logs being much fewer than normal logs. Table I shows the detailed statistical information of the datasets used in the experiments.

4.2 Evaluation Metrics

Log-based anomaly detection methods are typically evaluated using four metrics, which are precision, recall, F1 score, and accuracy, with the following definitions:

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

TABLE I
STATISTICAL INFORMATION OF THE DATASETS.

Dataset	Grouping	Training set			Testing set		
		# of log sequences	# of anomalous log sequences	% of anomalous log sequences	# of log sequences	# of anomalous log sequences	% of anomalous log sequences
HDFS	Session window	10,000	304	3.04%	575,061	16,838	2.93%
BGL	Fixed window	10,000	779	7.79%	942,699	76,024	8.06%

- Precision: the percentage of correctly detected anomalous log sequences among all predicted anomalous log sequences, i.e.,

$$Precision = \frac{TP}{TP + FP}. \tag{12}$$

- Recall: the percentage of correctly detected anomalous log sequences among all actual anomalous log sequences, i.e.,

$$Recall = \frac{TP}{TP + FN}. \tag{13}$$

- F1 score: the harmonic mean of precision and recall, i.e.,

$$F1 = \frac{2 \times TP}{2 \times TP + FP + FN}. \tag{14}$$

- Accuracy: the percentage of correctly classified log sequences among all log sequences, i.e.,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{15}$$

True Positive (TP): log sequence is anomalous and the model detects it as also anomalous.
False Negative (FN): log sequence is normal, but the model incorrectly detects it as anomalous.
False Positive (FP): log sequence is anomalous, but the model incorrectly detects it as normal.
True Negative (TN): log sequence is normal and the model detects it as also normal.

4.3 Hyperparameter Settings

During the experiments, we utilized AdamW [39] as the optimizer with learning rate of 1e-5 and weight decay of 0.01. The weight hyperparameter λ_{cl} is set to 0.1 and the temperature hyperparameter τ is set to 0.5 for contrastive loss. Additionally, we applied a dropout probability of 0.1 for data augmentation. Both semantic and log key sequential features are of dimension 512, and the fused feature dimension, obtained by merging the two features, is 1024. The size of log key vocabularies is set to 120 for the HDFS dataset and 2000 for the BGL dataset, which are determined based on the number of different log keys obtained after log parsing. LogContrast model is trained and saved with the highest F1 score in training.

4.4 Result Analysis

We compare LogContrast with the state-of-the-art log-based anomaly detection methods through comparative experiments, and explore the performance of LogContrast in the scenarios where log labels are scarce and inaccurate through ablation experiments. We formulated research questions (RQs) and answered them through experiments.

RQ1: How effective is LogContrast compared with the state-of-the-art approaches?

LogContrast conducts weakly supervised learning with a ratio of labels for supervision $r_{sup} = 0.2$ during training on the HDFS and BGL testing sets. The trained models are then tested on the testing sets.

We evaluate the performance of LogContrast and compare it with five influential semi-supervised and supervised baseline methods, namely Deeplog [11], LogAnomaly [12], LogBERT [13], LogEncoder [40] and LogRobust [10].

The experimental results of LogContrast and the baseline methods are presented in Table II. It can be observed that LogContrast outperforms the state-of-the-art semi-supervised methods. Compared to the supervised method LogRobust, LogContrast achieves superior performance on the HDFS dataset and performs slightly below LogRobust in terms of recall and F1 score on the BGL dataset.

RQ2: Does the weakly supervised contrastive learning method utilized in the training process of LogContrast effectively alleviate the issue of limited log labels?

To explore the effectiveness of weakly supervised contrastive learning, we train LogContrast models by varying the ratio of supervised learning r_{sup} from 0.0 to 1.0 with the interval of 0.1, and conduct evaluation on the testing sets. The experimental results are shown in Fig.5

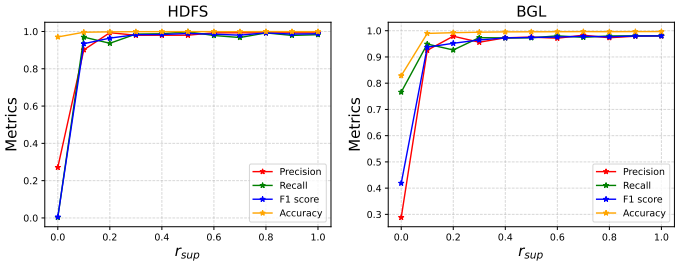


Figure 5. The effect of r_{sup} variation on the metrics.

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

TABLE II
PERFORMANCE OF LOGPROMPT AND FOUR OTHER LOG ANOMALY DETECTION FRAMEWORKS

Method	HDFS			BGL		
	Precision↑	Recall↑	F1 score↑	Precision↑	Recall↑	F1 score↑
DeepLog (semi-supervised)	0.840	0.695	0.761	0.237	0.853	0.384
LogAnomaly (semi-supervised)	0.833	0.849	0.835	0.289	0.851	0.462
LogBERT (semi-supervised)	0.803	0.718	0.758	0.867	0.925	0.895
LogEncoder (semi-supervised)	0.945	0.943	0.944	0.935	0.876	0.896
LogRobust (supervised)	0.919	0.925	0.923	0.921	**0.999**	**0.975**
LogContrast (semi-supervised)	**0.993**	**0.954**	**0.973**	**0.985**	0.926	0.955

As shown in Fig.5, LogContrast performs poor on the HDFS and BGL testing sets when it is trained without any supervision ($r_{sup} = 0$). However, we can observe that increasing r_{sup} from 0 to 0.2 leads to significant improvements for all metrics, with performance becoming close to that of fully supervised method with the r_{sup} of 1.0.

The experimental results indicate that LogContrast can effectively alleviate the problem of limited log labels in the real world by leveraging fewer labeled logs for weakly supervised learning while achieving comparable performance to fully supervised learning.

RQ3: Does the weakly supervised contrastive learning method utilized in the training process of LogContrast effectively alleviate the issue of training bias caused by incorrect log labels?

To investigate the impact of incorrect log labels in the training sets on LogContrast, we introduce label noise in the training sets to simulate the noise that exists in real-world log data, and vary the label noise ratio that denotes the ratio of label reversal in the training sets, represented by r_{noise}. We train LogContrast with two different supervised ratios, $r_{sup} = 0.2$ and $r_{sup} = 1.0$, corresponding to weakly supervised and fully supervised methods, respectively. Finally, we evaluate the trained models on the testing sets, and the experimental results are presented in Fig.6.

Fig.6 shows that when there is label noise in the training data, both weakly supervised and fully supervised methods have a significant decrease on all metrics. The higher r_{noise}, the more evaluation metrics decrease, indicating that label noise increases the uncertainty in detecting anomalous logs, resulting in poorer performance.

On the HDFS dataset, the weakly supervised method achieves lower precision than the fully supervised method only when half of the labeled logs are noise ($r_{noise} = 0.5$). For $r_{noise} = 0.1$ and $r_{noise} = 0.2$, the weakly supervised method shows lower recall, F1 score, and accuracy compared to the fully supervised method. However, for $r_{noise} = 0.3$ and $r_{noise} = 0.4$, the weakly supervised method performs better than the fully supervised method in terms of recall, F1 score, and accuracy. When $r_{noise} = 0.5$, both methods exhibit close performance, with recall and F1 score approaching 0.

On the BGL dataset, the fully supervised method outperforms weakly supervised method for all metrics when $r_{noise} = 0.1$ and $r_{noise} = 0.2$. When $r_{noise} = 0.3$, the weakly supervised method achieves slightly higher precision but lower recall compared to the fully supervised method, while the F1 score and accuracy are close. When $r_{noise} = 0.4$, all metrics of the weakly supervised method surpass those of the fully supervised method. When $r_{noise} = 0.5$, all metrics of both methods are nearly 0.

In general, the weakly supervised method shows relatively lower performance compared to the fully supervised method when the noise level is low. However, in scenarios with higher noise levels, the weakly supervised method exhibits stronger resilience to noise. When the ratio of label noise reaches 0.5, both the weakly supervised and fully supervised methods are significantly impacted, making anomaly detection challenging. Therefore, we need to make a trade-off in selecting the appropriate method based on the noise level present in the data.

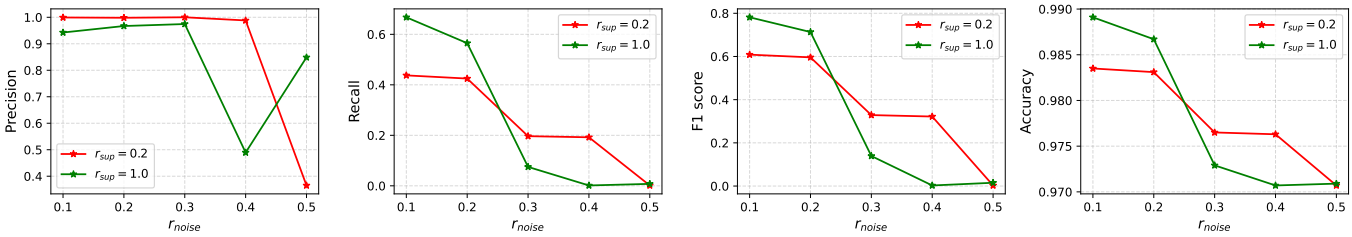
RQ4: What are the respective roles of log semantics and log key sequences in feature extraction for LogContrast?

In order to explore the significance and effectiveness of log semantic feature extractor (LSFE) and log key sequence feature extractor (LKFE), we conduct an ablation experiment. The ablation study aimed to investigate the roles and contributions of these two feature extractors. Specifically, we train three models, namely LogContrast-sem, LogContrast-logkey, and LogContrast-both, which only use the LSFE, only use the LKFE, and use both feature extractors, respectively. We use weakly supervised contrastive learning method with a supervised ratio of $r_{sup} = 0.2$ to train these models and evaluate them on the testing sets. Table III presents the experimental results, and we will analyze them from feature and dataset perspectives.

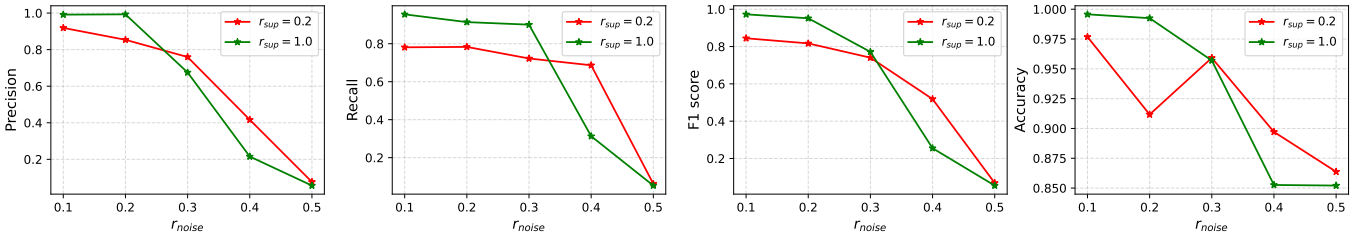
Semantic feature: Table III indicates that when relying solely on semantic features, LogContrast-sem exhibits lower precision, recall, and F1 score on the HDFS dataset compared to LogContrast-logkey and LogContrast-both, falling below 0.9. However, on the BGL dataset, LogContrast-sem achieves evaluation metrics above 0.9, close to LogContrast-logkey and LogContrast-both, with higher recall than LogContrast-logkey and LogContrast-both.

Log key sequence feature: The results from Table III

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

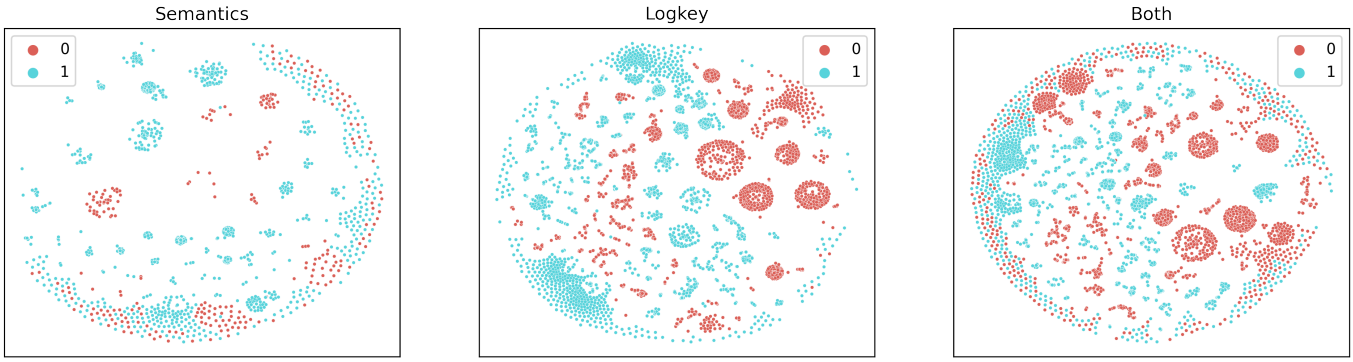


(a) HDFS

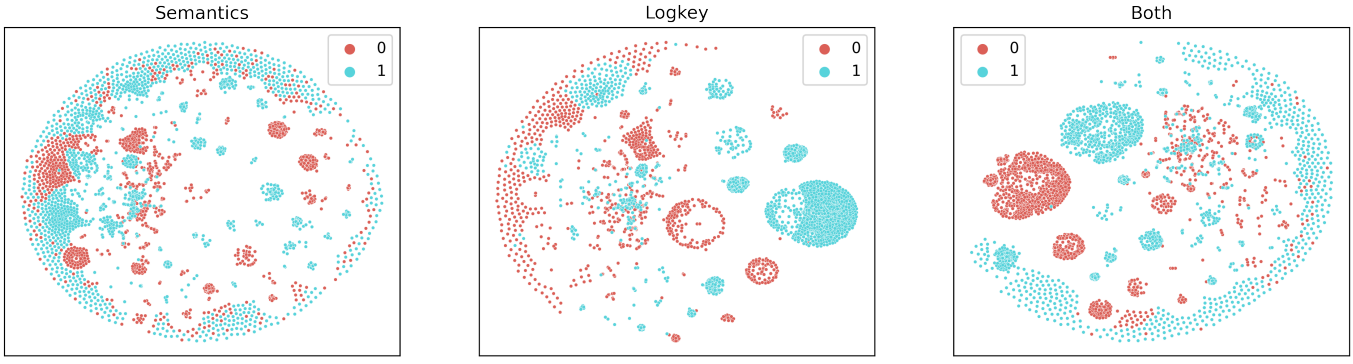


(b) BGL

Figure 6. The evaluation results on the testing sets with LogContrast trained under weakly supervised ($r_{sup} = 0.2$) and fully supervised ($r_{sup} = 1.0$) methods, and different levels of label noise ratio r_{noise} present in the training sets.



(a) HDFS



(b) BGL

Figure 7. t-SNE visualizations of the features ("0" represents normal instances, while "1" represents anomalous instances).

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

TABLE III
THE EVALUATION RESULTS OF LOGCONTRAST-SEM, LOGCONTRAST-LOGKEY, AND LOGCONTRAST-BOTH ON THE TESTING SETS.

Dataset	Method	Metrics			
		Precision↑	Recall↑	F1 score↑	Accuracy↑
HDFS	LogContrast-sem	0.8811	0.8437	0.8620	0.9921
	LogContrast-logkey	0.9807	**0.9906**	**0.9856**	**0.9992**
	LogContrast-both	**0.9932**	0.9540	0.9732	0.9985
BGL	LogContrast-sem	0.9354	**0.9642**	0.9496	0.9917
	LogContrast-logkey	0.9726	0.9334	0.9526	0.9925
	LogContrast-both	**0.9847**	0.9262	**0.9546**	**0.9929**

demonstrate that LogContrast-logkey shows good performance on both log datasets. On the HDFS dataset, LogContrast-logkey surpasses LogContrast-sem and LogContrast-both in recall, F1 score and accuracy, with only a slightly lower precision compared to LogContrast-both. On the BGL dataset, LogContrast-logkey achieves evaluation metrics that are comparable to LogContrast-both, although slightly lower. These findings highlight the importance of log key sequence features in log-based anomaly detection task.

Furthermore, a subset of 10,000 normal log sequences and 10,000 anomalous log sequences is chosen from each type of testing set. Features are extracted from these sequences using LogContrast-sem, LogContrast-logkey and LogContrast-both, and visualized in a two-dimensional space using a dimensionality reduction method t-SNE [41]. The visualizations are shown in Fig.7.

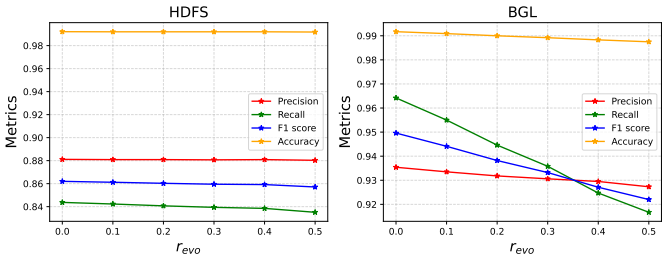


Figure 8. The effect on the evaluation results of LogContrast-sem by varying log evolution ratios (r_{evo}) in the testing sets.

Fig.7(a) presents the feature visualization on the HDFS dataset. In the semantic feature space, some anomalous logs cluster in the top-left region, while others are uniformly distributed below and to the right of the surface of the hypersphere, mixed with normal logs. In the log key feature space, most anomalous logs are concentrated in the bottom-left and upper regions of the hypersphere, whereas normal logs predominantly occupy the upper-right region, forming several smaller clusters. When both types of features are combined, normal and anomalous logs exhibit characteristics from both feature spaces. Overall, the log key sequence feature space shows the most distinct separation between normal and anomalous logs.

Fig.7(b) shows the feature visualization on the BGL dataset. In the semantic feature space, anomalous logs are distributed on the surface of the hypersphere, with higher density on the left side, while normal logs also cluster predominantly on the left. In the log key feature space, a significant number of anomalous logs gather on the right side of the hypersphere, while normal logs, along with a small portion of anomalous logs, are distributed on the upper-left surface and center of the hypersphere. When both types of features are combined, a portion of anomalous logs is uniformly distributed below and to the right of the hypersphere, while another portion forms a smaller cluster in the top-left region. Normal logs primarily cluster on the left side. Overall, the combined feature space shows the most distinct separation between normal and anomalous logs.

In summary, semantic features tend to distribute normal and anomalous logs more uniformly on the surface of the hypersphere, while log key features enable the separation of normal and anomalous logs into distinct clusters. Log key features demonstrate better discrimination between normal and anomalous logs compared to semantic features, making them crucial for log-based anomaly detection. However, as indicated in the Table III, the combination of both features achieves improved performance on the BGL dataset, emphasizing the importance of considering semantic features as well.

We identify possible reasons for the significant difference in training and evaluating performance of LogContrast-sem on these two datasets as follows:

- The HDFS dataset is grouped by session windows, some of which contain hundreds of logs, leading to excessively long log texts. The LSFE will automatically truncate overly long texts, and if anomalous logs are outside the truncated range, they cannot be detected. The BGL dataset is grouped by fixed windows with size of 5, resulting in shorter log sequences and ensuring semantic integrity.
- In natural language processing, words such as ”a” and ”the” are called stop words. They have a high frequency of occurrence but carry little information. We find that in the HDFS dataset, words such as ”block” and ”receive” frequently appear in both normal and anomalous logs, and they are similar to stop words in natural language.

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

Depending on these words to distinguish between normal and anomalous logs is difficult. This is a limitation of our current LSFE, which can be improved in the future by removing these stop words.

In order to balance the generalization ability and the adequacy of feature extraction, constructing the LogContrast-both model is a viable approach. In resource-constrained scenarios, it is possible to build a model using either semantic features or log key sequence features alone. Although the performance of LogContrast-sem on the HDFS dataset may be slightly subpar, we can demonstrate the significant role of semantic information in adapting to constantly evolving and modified logs in real-world scenarios. We will delve deeper into this aspect in RQ5.

RQ5: Can LogContrast adapt to the continuously updated and modified logs in real-world scenarios by extracting semantic features?

To simulate the evolution of log output statements due to updates, we performed four types of update operators on a certain proportion of logs in the testing set, including:

- Token insertion. A candidate word is randomly selected with equal probability from a portion of commonly used words (about 15,000 words) in the pre-trained language model's vocabulary as the word to be inserted, and then it is randomly inserted into the log with equal probability.
- Token deletion. A token in the log is randomly deleted with equal probability.
- Token replacement. A candidate word is randomly selected with equal probability from a portion of commonly used words (about 15,000 words) in the pre-trained language model's vocabulary as the word to be replaced, and then it is randomly replaced with a token in the log with equal probability.
- Log shuffling. A random segment of tokens in the log is selected and shuffled randomly.

The LogContrast-sem model is trained on a training set of non-evolved logs and then evaluated on an evolved testing set. The evolution ratio r_{evo} is defined as the proportion of logs in the testing set that underwent evolution. When a log underwent evolution, a random log update operator is selected with equal probability from among the four operators. The experiment varies r_{evo} from 0.0 to 0.5 with the interval of 0.1, and the results are presented in Fig.8.

Fig.8 reveals that all metrics decline as the log evolution ratio (r_{evo}) increases on the testing sets, although the degree of decline varies between the datasets. On the HDFS dataset, the metrics exhibit a modest decrease. However, on the BGL dataset, the metrics experience a more pronounced decline, particularly the recall. As r_{evo} increases from 0.0 to 0.5, there is an approximate decrease of 0.006, 0.05, 0.03 and 0.003 in precision, recall, F1 score and accuracy, respectively. This suggests that the evolution of log data somewhat compromises the capability for anomaly detection, albeit to a minimal extent.

Furthermore, our approach not only addresses the challenge of log evolution but also appears capable of mitigating complications arising from imprecise log parsing. When errors occur during log parsing, they tend to manifest in two scenarios [2]: (1) parameters could be inaccurately identified as constants, and (2) constants might be erroneously labeled as parameters. The former situation is comparable to token insertion, while the latter situation resembles token deletion.

Overall, the LogContrast-sem model trained with log semantic features can maintain high evaluation metrics even when 50% of the logs undergo minor evolution. If only log key sequence features are used for log anomaly detection, new log keys may be introduced during log parsing, which may cause out-of-vocabulary problem and result in the model malfunctioning or considering new log keys as anomalies. Therefore, the experiment demonstrates that the LogContrast model can adapt to log data evolution by extracting log semantic features.

RQ6: How does adjusting dropout rate of feature augmentation during training influence the performance of LogContrast?

To examine the influence of different dropout rates for feature augmentation during training, we conduct training with dropout rates of 0.1, 0.2 and 0.3, respectively. We then validate these models on the testing set. The outcomes are presented in Table IV.

TABLE IV
THE EVALUATION RESULTS ON THE TESTING SETS, WHEN MODELS ARE TRAINED WITH DIFFERENT DROPOUT RATES FOR FEATURE AUGMENTATION.

Dataset	Dropout rate	Metrics			
		Precision↑	Recall↑	F1 score↑	Accuracy↑
HDFS	0.1	**0.9932**	**0.9540**	**0.9732**	**0.9985**
	0.2	0.9930	0.9508	0.9714	0.9984
	0.3	0.9932	0.9471	0.9696	0.9983
BGL	0.1	**0.9847**	0.9262	**0.9546**	**0.9929**
	0.2	0.9831	0.9262	0.9538	0.9928
	0.3	0.9800	**0.9269**	0.9527	0.9926

From Table IV, it can be observed that on the HDFS dataset, precision exhibits minor fluctuations, whereas recall, F1 score and accuracy exhibit a decline as the dropout rate increases from 0.1 to 0.3. In contrast, the BGL dataset showcases minimal shifts in recall, yet precision, F1 score and accuracy all diminish with an increase in dropout rate from 0.1 to 0.3. The experiment suggests that elevating the dropout rate from 0.1 to 0.3 results in a slight performance decrease for the model. This trend is attributed to a heightened dissimilarity between augmented and original features, subsequently intensifying the challenge of contrastive learning.

5. Related Work

Early log-based anomaly detection methods primarily focused on traditional machine learning techniques such as Logistic Regression (LR) [6], Decision Trees [42], Support Vector Machine (SVM) [7], [43], Principal Component Analysis (PCA) [37], Invariant Mining [9], and Log Clustering [8]. While these methods showcase better time efficiency compared to

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

neural network-based deep learning methods, their precisions and recalls are relatively lower than those achieved by deep learning methods.

With the advancement of deep learning, semi-supervised and unsupervised deep learning-based methods have gained popularity in log-based anomaly detection. DeepLog [11] and LogAnomaly [12] leverage LSTM [31], while LogBERT [13] and LogEncoder [40] utilize BERT [30] as their backbone models. They focus on capturing the sequential patterns of normal logs in training stage. During the detection stage, the model predicts the next log event based on the current sequence. A sequence is classified as normal if the actual event is in the top-k predicted events, and anomalous otherwise. It is generally observed that supervised learning methods exhibit superior performance compared to semi-supervised and unsupervised methods, such as LogRobust [10] and NeuralLog [2]. However, they are not suitable for real-world scenarios with large-scale unlabeled log data.

6. Conclusion

In this paper, we propose LogContrast, a log-based anomaly detection method that utilizes weakly supervised contrastive learning. LogContrast incorporates a log semantic feature extractor, leveraging a pre-trained language model, and a log key feature extractor based on a Transformer encoder to extract essential features from logs. Self-supervised contrastive learning is the main approach used during model training, with a small number of labeled logs providing supervised signals. We evaluated the LogContrast model on the HDFS and BGL datasets, simulating scenarios in which log labels are scarce or incorrect. The experimental results demonstrate the excellent performance of LogContrast even with limited labeled logs and greater noise resistance compared to fully supervised methods. In addition, we explore the role of semantic features and demonstrate through experiments that semantic features have strong adaptability to constantly evolving logs.

References

[1] S. Satpathi, S. Deb, R. Srikant, and H. Yan, "Learning latent events from network message logs," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1728–1741, 2019.

[2] 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.

[3] H. Mi, H. Wang, Y. Zhou, M. R.-T. Lyu, and H. Cai, "Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1245–1255, 2013.

[4] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, vol. 55, no. 2, pp. 55–61, 2012.

[5] Y. Dang, Q. Lin, and P. Huang, "Aiops: real-world challenges and research innovations," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 2019, pp. 4–5.

[6] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen, "Fingerprinting the datacenter: automated classification of performance crises," in *Proceedings of the 5th European conference on Computer systems*, 2010, pp. 111–124.

[7] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[8] 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*, 2016, pp. 102–111.

[9] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection." in *USENIX annual technical conference*, 2010, pp. 1–14.

[10] 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*, 2019, pp. 807–817.

[11] 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*, 2017, pp. 1285–1298.

[12] 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 *IJCAI*, vol. 19, no. 7, 2019, pp. 4739–4745.

[13] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[14] E. Zheltonozhskii, C. Baskin, A. Mendelson, A. M. Bronstein, and O. Litany, "Contrast to divide: Self-supervised pre-training for learning with noisy labels," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 1657–1667.

[15] S. Wang, C. Li, R. Wang, Z. Liu, M. Wang, H. Tan, Y. Wu, X. Liu, H. Sun, R. Yang *et al.*, "Annotation-efficient deep learning for automatic medical image segmentation," *Nature communications*, vol. 12, no. 1, p. 5915, 2021.

[16] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM computing surveys (CSUR)*, vol. 54, no. 2, pp. 1–38, 2021.

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.

[17] W. Xu, *System problem detection by mining console logs*. University of California, Berkeley, 2010.

[18] S. Kabinna, C.-P. Bezemer, W. Shang, M. D. Syer, and A. E. Hassan, "Examining the stability of logging statements," *Empirical Software Engineering*, vol. 23, pp. 290–333, 2018.

[19] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 121–130.

[20] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*. IEEE, 2016, pp. 207–218.

[21] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[22] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[23] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[24] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[25] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6894–6910.

[26] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9929–9939.

[27] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 859–864.

[28] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE international conference on web services (ICWS)*. IEEE, 2017, pp. 33–40.

[29] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[30] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[32] 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.

[33] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-attentive classification-based anomaly detection in unstructured logs," in *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2020, pp. 1196–1201.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: a large collection of system log datasets towards automated log analytics," *arXiv preprint arXiv:2008.06448*, 2020.

[37] 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*, 2009, pp. 117–132.

[38] 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'07)*. IEEE, 2007, pp. 575–584.

[39] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[40] J. Qi, Z. Luan, S. Huang, C. Fung, H. Yang, H. Li, D. Zhu, and D. Qian, "Logencoder: Log-based contrastive representation learning for anomaly detection," *IEEE Transactions on Network and Service Management*, 2023.

[41] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. 11, 2008.

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

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

Authorized licensed use limited to: Civil Aviation University of China. Downloaded on October 13,2025 at 01:35:03 UTC from IEEE Xplore. Restrictions apply.