Weakly-supervised Log-based Anomaly Detection with Inexact Labels via Multi-instance Learning

Minghua He[†], Tong Jia[‡§*], Chiming Duan[†], Huaqian Cai[¶], Ying Li[‖], and Gang Huang[‡§]
[†]School of Software and Microelectronics, Peking University, Beijing, China
{hemh2120, duanchiming}@stu.pku.edu.cn
[‡]Institute for Artificial Intelligence, Peking University, Beijing, China
[§]National Key Laboratory of Data Space Technology and System, Beijing, China
{jia.tong, hg}@pku.edu.cn
[¶]School of Computer Science, Peking University, Beijing, China
caihq@pku.edu.cn
[‖]National Engineering Research Center for Software Engineering, Peking University, Beijing, China
li.ying@pku.edu.cn

***Abstract*—Log-based anomaly detection is essential for maintaining software availability. However, existing log-based anomaly detection approaches heavily rely on fine-grained exact labels of log entries which are very hard to obtain in real-world systems. This brings a key problem that anomaly detection models require supervision signals while labeled log entries are unavailable. Facing this problem, we propose a new labeling strategy called inexact labeling that instead of labeling an log entry, system experts can label a bag of log entries in a time span. Furthermore, we propose MIDLog, a weakly supervised log-based anomaly detection approach with inexact labels. We leverage the multi-instance learning paradigm to achieve explicit separation of anomalous log entries from the inexact labeled anomalous log set so as to deduce exact anomalous log labels from inexact labeled log sets. Extensive evaluation on three public datasets shows that our approach achieves an F1 score of over 85% with inexact labels.**

***Index Terms*—Weakly-supervised Learning, Log Analysis, Anomaly Detection, Multi-instance Learning.**

I. Introduction

Software systems are becoming increasingly large and complex and are subject to more failures. As system logs record system states and events of running processes, they are an excellent source of information for anomaly detection. Log-based anomaly detection is promising for system reliability and has been widely studied.

In recent years, many log-based anomaly detection approaches have been proposed [1]–[15]. They used historical logs to detect the occurrence of anomalies in the system using supervised or unsupervised/semi-supervised methods. Unsupervised models only use the normal samples for training. They [7]–[10], [12] utilize sequential neural networks such as LSTM, GRU, etc. to learn the sequence of log events, i.e., 'log sequences', under normal conditions. They predict subsequent log events in these sequences and identify any unmatched event as an anomaly. However, as proved in many prior works [1], [5], [11], the effectiveness of unsupervised models is often very limited due to the lack of supervision of anomalous logs. For instance, as discussed in [16], unsupervised models perform worse than 25% F1-score compared with supervised models in both public datasets and real-world industrial datasets. Supervised models [1]–[4] are more effective, however, their effectiveness heavily depends on a large amount of labeled training logs, especially anomalous log entries. Semi-supervised models [5] leverage an unsupervised clustering method to deduce data labels but they still rely on historical anomalous log entries as supervised models. As discussed in [17], when reducing 90% labeled anomalous log entries, the F1-score of semi-supervised models greatly reduces from 91.36% to 31.06%. Therefore, the effectiveness of existing works heavily relies on labeled logs, especially anomalous labeled logs.

However, obtaining labeled logs especially labeled anomalous logs requires huge human efforts which is often unacceptable in real-world systems. Today's software systems often produce a tremendous amount of system logs. For instance, a Microsoft online service system produces over 1PB system logs per day [1]. Analyzing such a huge amount of logs and labeling anomalous log entries from them is almost impossible for system experts. This brings a key contradiction that anomaly detection models require supervision signals while labeled log entries are unavailable.

In our view, existing labeling strategy is one of the key reasons of this contradiction. Existing labeling strategy follows the methodology of traditional supervised machine learning models that gives labels on each data instance, and each data instance is a log entry in the log-based anomaly detection task. This data-instance-level labeling strategy is called exact labeling in this paper. Exact labeling requires system experts to manually analyze tremendous amount of log entries and judge whether each log entry is normal or an anomaly which is nearly impossible in real-world.

As a result, we propose a new labeling strategy called inexact labeling and demonstrate that inexact labeling is much more suitable for the log-based anomaly detection task. The key idea of inexact labeling is that instead of labeling an log

* Corresponding Authors

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00189

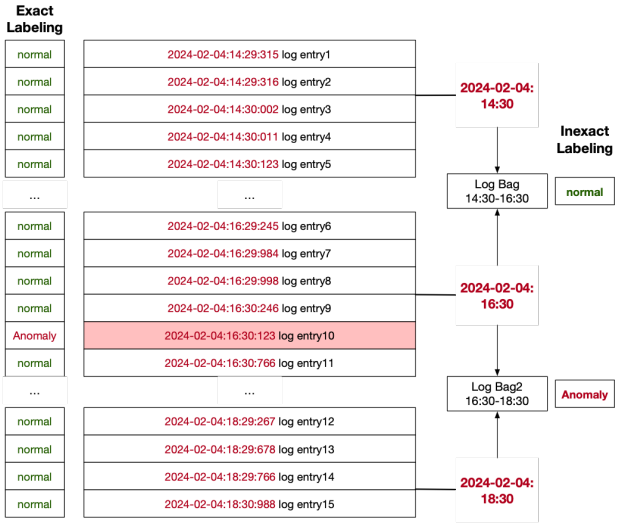


Fig. 1. Exact labeling vs. inexact labeling. A sequence of log entries from 2024-02-04 14:29 to 2024-02-04 18:30 lies in the middle. The labels of exact labeling strategy are shown on the left and the labels of inexact labeling strategy are shown on the right.

entry, system experts label a bag of log entries in a time span. Figure 1 shows the difference of exact labeling and inexact labeling. In the middle part, we show a sequence of log entries in the time span from 2024-02-04 14:29:315 to 2024-02-04 18:30:988. Exact labeling gives labels on each log entry as shown in the left part. Inexact labeling gives labels on a log bag that contains all log entries in a time span. For instance, experts give one ”normal” label on the log bag that contains log entries of two hours from 2024-02-04 14:30 to 2024-02-04 16:30, and one ”anomaly” label on the log bag of log entries from 2024-02-04 16:30 to 2024-02-04 18:30.

Through inexact labeling, system experts only have to determine in which time span the system performs abnormally instead of reading and analyzing tremendous number of log entries. In this way, the human labeling efforts are greatly reduced. In our experience of operating real-world systems, we noticed that system experts can easily determine the anomaly time span based on application performance, request execution status, user reports, etc., and often record the anomaly information in historical anomaly or failure reports. Figure 2 shows an anomaly report from a real-world online service system. The report states that the service mall-cart-server encountered packet dropping in the time span from 2024-01-04 17:31:36 to 2024-01-04 17:40:00, and suggests that this anomaly could be attributed to delays or instability in network communication. From the anomaly report, system experts can easily give an ”anomaly” label on the log bag of time span 2024-01-04 17:31:36 to 2024-01-04 17:40:00.

Though inexact labeling strategy greatly reduces labeling efforts, inexact labels cannot be directly utilized to build anomaly detection models, because inexact labels are fuzzy. The anomalous label of a log bag indicates that it has a high possibility to contain anomalous log entries, but we do not know how many and which of them are anomalous.

Title: Packet dropping reported by customers

Trace ID: ea1a00002f17043610235505025d0007 **Service:** mall-cart-server

Summary

Customers reported that packet dropping and network failure occurs on the container.

Discription

During the time period of ***2024-01-04 17:31:36*** to ***2024-01-04 17:40:00***, the "***mall-cart-server***" service encountered packet dropping within its container. Possible causes for this issue include network communication delays or instability triggered, leading to packet dropping within the nodes and necessitating retransmission.

Fig. 2. An anomaly report from a real-world online service system.

Our solution to this problem is to automatically seperate and deduce exact labels especially anomalous labels of log entries from inexact labels. However, achieving this is not easy. First, as anomalous labels are inexact, there exists a possibility that most log entries in the anomalous log bag are actually normal. This could lead to a consequence that normal log bag and anomalous log bag are very similar. How to explicitly separate and deduce anomalous log entries from the anomalous log bag becomes very challenging. Second, anomalous logs can be very different due to various system faults [12] and it is hard to discover and capture the features of anomalous log entries so as to judge which log entry is anomalous.

Facing these challenges, in this paper, we propose MIDLog, a weakly-supervised log-based anomaly detection approach with inexact labels via multi-instance learning. To solve the first challenge, we leverage the idea of multi-instance learning to constantly compare normal log bags with anomalous log bags to deduce anomalous log entries during training. Multi-Instance Learning (MIL) treats anomaly detection as a regression problem, providing a score for each input data instance, i.e., log entry. During training, for each pair of normal bags and anomalous bags, MIL retrieves the instance with the highest score in each bag and promotes anomalous instances to attain higher scores than normal instances through a ranking loss, thereby achieving explicit separation of anomalous log entries from the inexact labeled anomalous log bag. To sovle the second challenge, we introduce an Global Normality Center Learning (GNCL) stage, which learns the global normality within log sequences through self-distillation and discover features of anomalous log entries by assessing their deviations from the global normality.

We evaluated the effectiveness of MIDLog through extensive experiments on three commonly utilized public log datasets: Spirit, Thunderbird, and Hadoop, as well as an industrial dataset System A. Our method attains an F1 score exceeding 85% on three public datasets when an inexact label marks 800 log entries, showcasing state-of-the-art performance. It yields a notable improvement in F1 score, up to 6.20% compared to supervised methods with the same number of exact labels. Additionally, the experiments demonstrate the robustness of our method to label exactness and its applicability in complex industrial scenarios with varying levels of label exactness. In summary, the contributions of this paper are as follows:

- To the best of our knowledge, we are the first to propose the inexact labeling strategy for the log-based anomaly detection task.
- We propose MIDLog, a weakly-supervised log-based anomaly detection approach via multi-instance learning. We leverage the multi-instance learning paradigm to achieve explicit separation of anomalous log entries from the inexact labeled anomalous log bag so as to deduce exact anomalous log labels from inexact labeled log bags.
- We further propose a self-distillation method to learn the global normality within log sequences and discover features of anomalous log entries by assessing their deviations from the global normality.
- Extensive evaluation results on three public datasets and an industrial dataset show the effectiveness of our approach.

II. Preliminaries

A. Log Terminology

Large-scale software-intensive systems commonly use logs to record runtime states, playing a crucial role in system maintenance. In figure 3, we present an example of log data extracted from the publicly available Hadoop dataset, collected from a distributed system. Log messages represent unstructured text generated by logging statements within the system (e.g., logging.info()), containing both log events and parameters. Log events (e.g., "Number of reduces for") form the constant part of log messages, consisting of text strings, and serve as abstractions in log analysis. Log parameters make up the variable part of log messages, recording specific system attributes during runtime (e.g., job ID). Log parsing, typically the initial step in log analysis, separates log events from log parameters in unstructured log messages. An event sequence consists of a sequence of log events that document the execution flow of a particular task, commonly organized by job ID or timestamp. This forms the fundamental unit for log-based anomaly detection.

B. Inexact Supervision Log Anomaly Detection

We aim to leverage the inexact labels to build supervised anomaly detection models. We term this task as **Inexact Supervised Log-based Anomaly Detection (ISLAD)**. In the settings of this task, data labels in the training dataset are at bag level, that is, the labeling unit is a log bag containing certain number of log entries rather than a log entry. The normal label of a log bag indicates that it does not contain any anomalous log entries. The anomalous label of a log bag indicates that it contains anomalous log entries and we do not know how many and which of them are anomalous.

We initially introduce the foundational concepts of the ISLAD task. Given a set of log sequences $X = \{x_1, x_2, ..., x_n\}$ in a software system, our objective is to detect the anomaly status y_i for each log sequence x_i, where $y_i \in \{0, 1\}$. The set of labels for log sequences is denoted as $Y = \{y_1, y_2, ..., y_n\}$. X can be partitioned into k groups: $X = \{X_{B_1}, X_{B_2}, ..., X_{B_k}\}$, where each X_{B_i} denotes a set of multiple consecutive log sequences x_i: $X_{B_i} = \{x_{j+1}, x_{j+2}, ..., x_{j+l}\}$. We refer to X_{B_i} as a bag and x_i as an instance. In the ISLAD task, during the training process, the model is only provided with inexact bag-level labels $\{Y_{B_1}, Y_{B_2}, ..., Y_{B_k}\}$. For all $x_j \in X_{B_i}$, if there exists $y_j = 1$, then $Y_{B_i} = 1$; otherwise, $Y_{B_i} = 0$. This formulation implies that negative bags represent exact instance-level labels, whereas, for positive bags, we can only affirm the presence of anomalous instances without determining which ones. In log bags with anomalous labels, normal instances and anomalous instances in positive bags may not exhibit significant differences. Therefore, the primary challenge of ISLAD is to separate anomalous instances from positive bags.

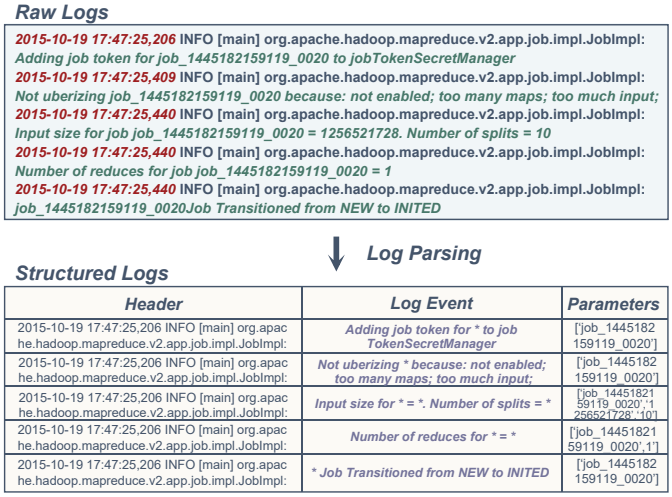


Fig. 3. Examples of key terminologies in log analysis. The red section in raw logs represents timestamps, while the green section represents log contents.

We employ the MIL framework [18] to address the ISLAD task, where the key idea is to encourage anomalous instances to have higher scores through ranking. Specifically, we consider anomaly detection as a regression problem, generating a score $G(v_i)$ for each input instance embedding v_i to represent its anomaly level. For an input bag X_B and its embedding set $\mathcal{V}_B = \{v_1, v_2, ..., v_n\}$, our purpose is to give each anomalous instance a higher score than normal instances, denoted as $G(v^a) > G(v^n)$, where $Y(v^n) = 0$, $Y(v^a) = 1$. However, the lack of instance-level anomaly labels $Y(v_i)$ impedes the ranking of each v_i within the bag. Consequently, we retrieve instances with the maximum scores for sorting to learn distinctive anomaly scores: $\max G(\mathcal{V}_B^a) > \max G(\mathcal{V}_B^n)$, where $Y_B(\mathcal{V}_B^n) = 0$, $Y_B(\mathcal{V}_B^a) = 1$. The fundamental idea is that the instance with the highest score in $\mathcal{V}_B^a$ is likely to be an anomaly, while the instance with the highest score in $\mathcal{V}_B^n$ represents the most similar normal instance to anomalies, and their separation best signifies the division between the two classes. We devise a ranking loss [19] for training:

$$\mathcal{L}_R(\mathcal{V}_B^a, \mathcal{V}_B^n) = \max(0, 1 - \max G(\mathcal{V}_B^a) + \max G(\mathcal{V}_B^n)), \quad (1)$$

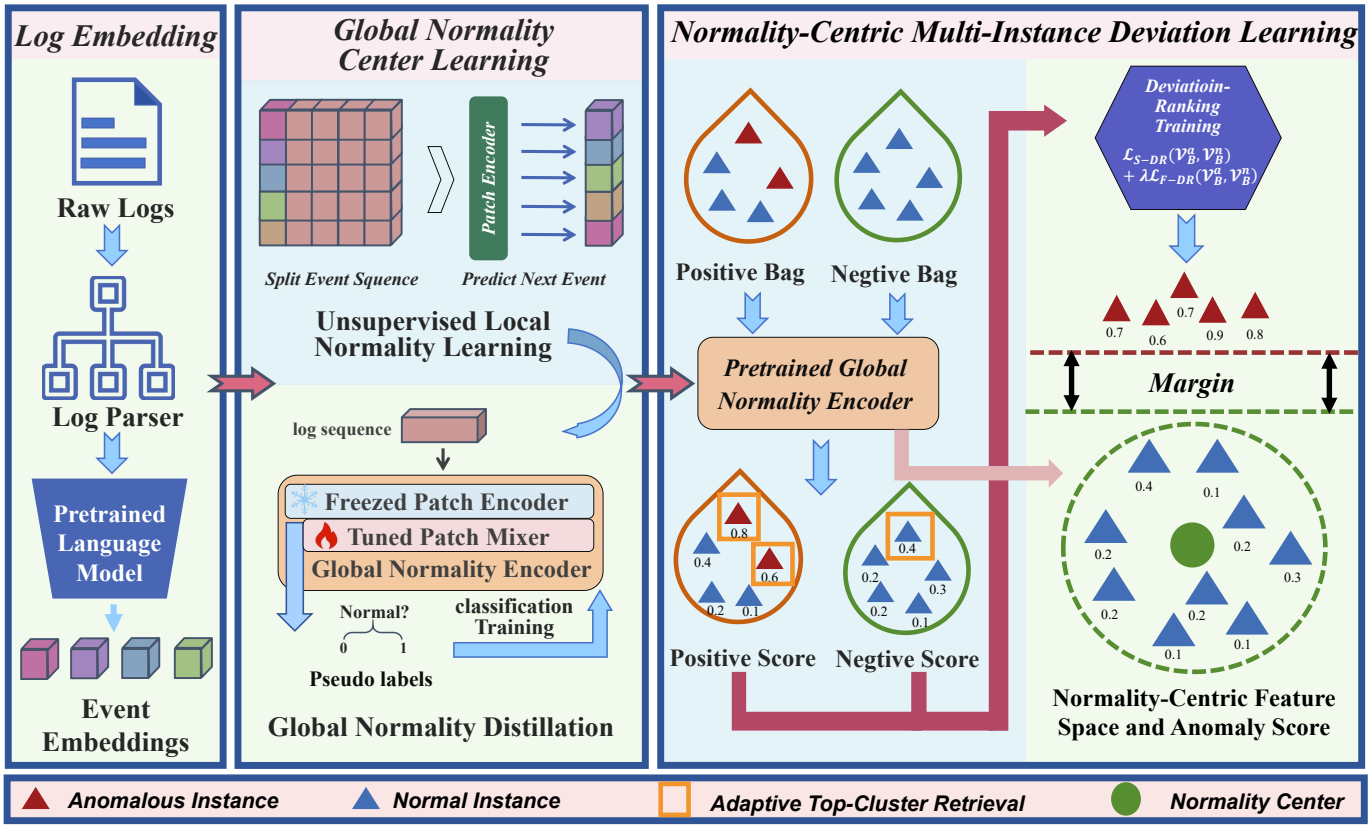


Fig. 4. The proposed weakly-supervised log-based anomaly detection pipeline with inexact labels for MIDLog. MIDLog pipeline comprises three key stages: Log Embedding, Global Normality Center Learning (GNCL), and Normality-Centric Multi-Instance Deviation Learning. First, we parse the input log sequence into a sequence of log events and extract the corresponding event embedding. Second, we use the GNCL stage for pre-training to learn the global normality of logs from the event embedding sequence. Finally, the event embedding sequence is fed into the Normality-Centric Multi-Instance Deviation Learning stage initialized by the GNCL stage to learn distinctive anomaly scores for predicting anomalies. MIDLog enforces the separation of anomalies in the feature space and score space, enabling MIDLog to solve the ISLAD task.

III. Methodology

A. Overview

In this paper, we present MIDLog, a novel approach based on multi-instance learning, specifically designed to address the challenging Inexact Supervision Log Anomaly Detection (ISLAD) task defined in section 2. MIDLog is a method that inputs log entries and outputs anomaly scores, performing anomaly detection based on a set threshold. MIDLog comprises three key stages: Log Embedding, Global Normality Center Learning (GNCL), and Normality-Centric Multi-Instance Deviation Learning. With the funtionalities and advantages of these stages, MIDLog can deduce highly abstract anomalies by assessing their statistical deviations from normality and ensure anomaly separation in both feature space and anomaly score space, thereby showcasing its effectiveness in addressing ISLAD task. Figure 4 illustrates the pipeline of MIDLog.

In MIDLog, we initially employ the classic log parsing technique Drain [20] to handle unstructured raw logs from software systems, acquiring log events and processed log event sequences. Subsequently, we follow previous work [5], [11] to extract semantic embeddings for each log event. Evidence [1] suggests that semantic embeddings offer more informative representations compared to traditional index-based methods. The training of MIDLog is mainly divided into two major stages: a pre-training stage represented by GNCL and a fine-tuning stage represented by Normality-Centric Multi-Instance Deviation Learning. Considering the abstract nature of log anomalies, log event embeddings are fed into the GNCL stage, which captures the global normality of complex patterned log sequences by training a global normality encoder, enabling MIDLog to defining anomalies by assessing their deviations from this normality. Subsequently, log embeddings are fed into the Normality-Centric Multi-Instance Deviation Learning stage, which is based on the global normality center learned in the GNCL stage, learning distinctive anomaly scores for the input log event embedding sequences to predict log anomalies. Leveraging the adaptive capability of the this stage for variable anomaly counts and the enforced separation of anomalies in feature representation space and anomaly score space, MIDLog effectively separates highly abstract anomalous log sequences using inexact anomaly labels.

B. Global Normality Center Learning

In log bags with anomalous labels, anomalous logs may not have significant differences from normal logs, making it particularly difficult to explicitly deduce anomalies under inexact anomaly labels. For this purpose, we introduce the GNCL stage aimed at training a global normality encoder to encode the global normality center of instance-level log sequences. Normality represents the overall characteristics of a series of normal log sequences, which reflect the behavior or state of the system during normal operation. Consequently, MIDLog can deduce anomalies by assessing their statistical deviations from normality based on the Out of Distribution principle. The GNCL stage consists of two key steps: an unsupervised method for learning local normality and a self-distillation process for generalizing it to global normality.

1) Unsupervised Local Normality Learning: The GNCL step initially conducts **Unsupervised Local Normality Learning**, with the goal of learning the local normality of logs by predicting the next log event of local log sequences. Specifically, for a normal log event sequence $x = [e_1, e_2, ..., e_n]$ of length n, we partition x into $n - l$ sequences of length l using a sliding window of length $l < n$, resulting in sequences $\{x_{L_1}, x_{L_2}, ..., x_{L_{n-l}}\}$, where $x_{L_i} = [e_i, e_{i+1}, ..., e_{i+l-1}]$. Unsupervised Local Normality Learning encodes local normality by predicting the next log event e_{i+l} of x_{L_i}, thus fitting the function $f(x_{L_i}) = \overline{e}_{i+l}$, where $\overline{e}_{i+l}$ denotes the one-hot vector of e_{i+l}. This strategy has proven effective in learning the normal patterns of log sequences without relying on exact anomaly labels [7], [8], rendering it well-suited for the ISLAD task. Considering the potential noise and temporal nature of log event sequences, inspired by [1], Unsupervised Local Normality Learning designs a **Patch Encoder** that employs a bidirectional long short-term memory network (Bi-LSTM) with an attention mechanism to fit the objective function f. This architecture has a significantly faster inference speed than Transformer and has been proven to be capable of handling log sequences [21], [22]. For each timestamp t, the system retains a hidden layer vector h^t and attention weights $\alpha^t = tanh(W_t^\alpha \cdot h_t)$, with the objective of fitting:

$$Softmax(W \cdot \sum_{t=0}^{t=T} \alpha^t \cdot h_t) = \overline{e}_{i+l}, \tag{2}$$

This architecture enables the patch encoder to ignore potential noisy logs while preserving long-term information from the input log event sequences.

2) Global Normality Distillation: Given that the patch encoder only encodes the local normality of log sequences with a length of l, the encoding of instance-level log sequences with a length of $n > l$ is incomplete. Therefore, it is necessary to generalize it to the global normality of log sequences with a length of n, which eliminates the gap between feature scales and helps to determine a more reliable normality center. To this aim, the GNCL stage has developed the **Global Normality Distillation** step, which designs a self-distillation process to extend the patch encoder trained by Unsupervised Local Normality Learning into a **Global Normality Encoder** without relying on any exact anomaly labels. The resulting global normality encoder is capable of encoding the global normality.

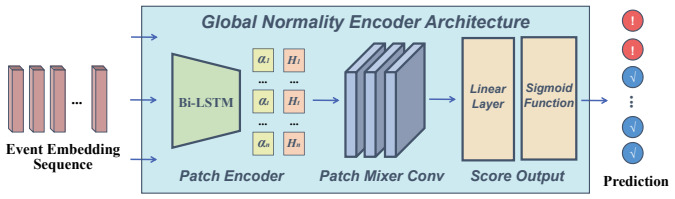


Fig. 5. The network architecture of MIDLog. Given a sequence embedding of log events, the network predicts anomalies in logs. This framework comprises three cascaded modules, comprising a Bi-LSTM, a convolutional neural network, and a linear layer.

As shown in Figure 5, the Global Normality Distillation step added a convolutional neural network **Patch Mixer** with three convolutional layers based on the patch encoder trained by the Unsupervised Local Normality Learning step to combine multiple local representations of length l encoded by the patch encoder into a global representation of length n. The convolutional neural networ is able to extract global information from sequences while retaining effective local information. The combined network serves as a new encoder called global normality encoder. The Global Normality Distillation step designs a self-distillation process to train the global normality encoder without relying on any exact labels. Specifically, Global Normality Distillation first uses the patch encoder trained by Unsupervised Local Normality Learning as a teacher model to infer on the training set $X_{train} = \{x_1, x_2, ..., x_n\}$ and generate a pseudo-label set $\hat{Y}_{train} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$, where $\hat{y}_i \in \{0, 1\}$. To generate pseudo-labels, we perform anomaly detection using Patch Encoder in a manner that predicts the next log event in the log sequence. When the predicted event does not match the actual event, the log sequence is identified as anomalous, resulting in a pseudo-label of 1; otherwise, it is labeled as 0. Subsequently, the new training set $\{X_{train}, \hat{Y}_{train}\}$ is used to train the student model global normality encoder, with the training process optimized by cross-entropy loss. It is noteworthy that during training, we initialize the global normality encoder with the patch encoder trained in Unsupervised Local Normality Learning, and we freeze the parameters of the patch encoder, optimizing solely the patch mixer. Evidence [23] suggests that this freezing strategy can preserve the local normality learned by the patch encoder, aiding in the identification of fine-scale anomalous logs.

C. Normality-Centric Multi-Instance Deviation Learning

The Normality-Centric Multi-Instance Deviation Learning stage relies on the global normality center learned in the GNCL stage. It learns distinctive anomaly scores for the input log event embedding sequences to predict log anomalies. The vanilla MIL method only optimizes the anomaly scores, and indirect capturing and learning of anomaly log features are

suboptimal, which may not effectively separate anomalies. To address this, the Normality-Centric Multi-Instance Deviation Learning stage extends MIL to multi-instance deviation learning (MIDL step), utilizing both feature distance and anomaly scores as training signals to achieve explicit separation of anomalies. Furthermore, this stage proposes an adaptive sampling strategy (ATCR step) for the MIDL step, which adaptively determines the number of anomalies in positive/anomalous bags through clustering, thereby effectively separating potential anomalies. We first elucidate the network structure , which consists of a global normality encoder and a linear layer. The global normality encoder is initialized by the GNCL stage to utilize learned normality for detecting abstract anomalies.

1) Adaptive Top-Cluster Retrieval: The adaptive top-cluster retrieval step aims to determine the number of anomalies in positive/anomalous bags to separate potential anomalies. The vanilla MIL method assumes that anomalies in the positive bags are sparse, thus only retrieving a fixed number of instances from the positive bags to participate in optimization, however, as for the ISLAD task, the number of anomalies in positive bags is uncertain. Therefore, we employ a novel sampling strategy in this step, with the key idea of adaptively retrieving potential anomalies using clustering.

We first introduce the sampling strategy of vanilla MIL. As per the terminology introduced in section 2, given bag X_B and its embedding sequence $\mathcal{V}_B$, the global normality encoder G encodes its anomaly score sequence: $G(\mathcal{V}_B)$. According to the introduction in Section 2, MIL retrieves the instance with the highest score and uses ranking loss EQ (1) for optimization, where $Y_B(\mathcal{V}_B^n) = 0$, $Y_B(\mathcal{V}_B^a) = 1$. In order to facilitate description, EQ (1) can be rewritten as follows.

$$\begin{aligned}\mathcal{L}_R(\mathcal{V}_B^a, \mathcal{V}_B^n) = \max(0, 1 + \max G(\mathcal{V}_B^n) \\ - \frac{1}{\|S(\mathcal{V}_B^a)\|} \sum G(S(\mathcal{V}_B^a))),\end{aligned} \tag{3}$$

$$S(\mathcal{V}_B^a) = \{v_i | G(v_i) \in Top_1 G(\mathcal{V}_B^a)\}, \tag{4}$$

We refer to S as the sampling function, which selects potential anomaly instances to participate in optimization based on the anomaly scores of the positive bag. Existing work [24], [25] extends the retrieved instances from $Top{-}1$ to $Top{-}k$ scores within $\mathcal{V}_B^a$ to separate more potential anomalies.

$$S(\mathcal{V}_B^a) = \{v_i | G(v_i) \in Top_k G(\mathcal{V}_B^a)\}, \tag{5}$$

Where k is a predefined fixed hyperparameter. The theoretical foundation in [24] indicates that when k is approximately equal to the number of anomalies μ in $\mathcal{V}_B^a$, the ranking loss can effectively separate potential anomalies. However, within ISLAD, the number of anomalies μ in different positive bags is uncertain, rendering the strategy of employing a fixed k across diverse bags ineffective. ATCR introduces clustering to adapt to the varying numbers of anomalies in the positive bags. Initially, ATCR applies $k-means$ clustering to $G(\mathcal{V}_B)$, resulting in z cluster centers $c_1, c_2, ..., c_z$. Afterward, ATCR computes the average score of each cluster center as m_i, and selects the size of the cluster with the highest average score as k.

$$\begin{aligned}m_i =& \frac{1}{|c_i|} \sum_{G(v) \in c_i} G(v), \\ k =& t \cdot \arg\max_{c_i} |m_i|,\end{aligned} \tag{6}$$

Where t is a hyperparameter used to adjust the retrieval ratio. Compared to the $Top-k$ approach, ATCR can adjust to the differing number of anomalies across various bags, aiding in the clear separation of potential anomalous instances.

2) Multi-Instance Deviation Learning: The Normality-Centric Multi-Instance Deviation Learning stage extends MIL to **Multi-Instance Deviation Learning (MIDL)**. It utilizes feature distance and anomaly scores as concurrent training signals, facilitating clear separation of anomalies in both feature space and score output space, crucial for addressing the ISLAD task. Drawing inspiration from neural deviation learning [26], MIDL introduces a novel loss function called the deviation-ranking loss. The key idea of neural deviation learning is to learn the feature deviations of normal and abnormal instances to achieve a clear separation of anomalies. This loss function encodes a more compact representation of normality and simultaneously enforces a margin between normal and anomalous instances in both feature space and score output space. We first introduce the deviation-ranking loss, which consists of a deviation loss and a ranking loss:

$$\begin{aligned}\mathcal{L}_{DR}(\mathcal{V}_B^a, \mathcal{V}_B^n) =& dev(\mathcal{V}_B^n) + \max(0, a - \\ & dev(S(\mathcal{V}_B^a)) + dev(\mathcal{V}_B^n)),\end{aligned} \tag{7}$$

where dev quantifies the deviation of instances from the normality center, a represents a hyperparameter defining the deviation margin, and S denotes a sampling function driven by ATCR (EQ (5), (6)). Such design encourages tight clustering of normal instances around the normality center [27], while positioning anomalous instances farther from this center, effectively facilitating anomaly separation in ISLAD.

Subsequently, MIDL defines dev separately at the feature and score levels. For the feature level, MIDL defines dev as the Euclidean distance between instance features and the feature normality center. Specifically, MIDL defines the feature normality center as the average features of normal instances: $C = \frac{1}{|\mathcal{V}_B^n|} \sum_{v^n \in \mathcal{V}_B^n} \hat{G}(v^n)$. Here, $\hat{G}$ encodes the input embedding's features. Then, MIDL calculates the feature deviation:

$$dev_F(\mathcal{V}_B) = \frac{1}{|\mathcal{V}_B|} \sum_{v \in \mathcal{V}_B} \|\hat{G}(v) - C\|^2, \tag{8}$$

This strategy is inspired by the One-Class Classification paradigm [28], which has been proven to encode compact normal representations. Finally, the deviation-ranking loss at the feature level is defined as:

$$\begin{aligned}\mathcal{L}_{F-DR}(\mathcal{V}_B^a, \mathcal{V}_B^n) =& dev_F(\mathcal{V}_B^n) + \max(0, a - \\ & dev_F(S(\mathcal{V}_B^a)) + dev_F(\mathcal{V}_B^n)),\end{aligned} \tag{9}$$

At the score level, MIDL establishes a prior distribution as the normality center for scores, which yields interpretable scores. Score level deviation is defined as:

$$dev_S(\mathcal{V}_B) = \frac{1}{|\mathcal{V}_B|} \sum_{v \in \mathcal{V}_B} \| \frac{\tilde{G}(v) - \mu}{\sigma} \|^2, \tag{10}$$

Where $\tilde{G}$ encodes the scores of input embeddings. Notably, unlike G, the scores encoded by $\tilde{G}$ do not undergo the $Sigmoid$ function, that is, $G = Sigmoid(\tilde{G})$. This occurs because the $Sigmoid$ function compresses the score space, which is unfavorable for constraining a compact normality center. Research [29] indicates that anomaly scores in real-world scenarios closely adhere to the standard Gaussian distribution, motivating us to adopt this configuration for robust real-world performance. We propose employing $\mathcal{L}_R$ (EQ (3)) as the ranking loss at the score level, which introduces a bias margin in the score space after $Sigmoid$ compression, mitigating potential impacts of hyperparameter a while remaining consistent with MIDL and MIL methods. The deviation-ranking loss at the score level is represented as:

$$\begin{aligned} \mathcal{L}_{S-DR}(\mathcal{V}_B^a, \mathcal{V}_B^n) = dev_S(\mathcal{V}_B^n) + \max(0, 1 + \max G(\mathcal{V}_B^n) \\ - \frac{1}{\|S(\mathcal{V}_B^a)\|} \sum G(S(\mathcal{V}_B^a))), \end{aligned} \tag{11}$$

Finally, the loss function of MIDL is as follows, where λ is a hyperparameter utilized for balancing the loss function.

$$\mathcal{L}(\mathcal{V}_B^a, \mathcal{V}_B^n) = \mathcal{L}_{S-DR}(\mathcal{V}_B^a, \mathcal{V}_B^n) + \lambda \mathcal{L}_{F-DR}(\mathcal{V}_B^a, \mathcal{V}_B^n), \tag{12}$$

D. Anomaly Detection

Following the above stages, we can establish a MIDLog network for detecting log anomalies. For the incoming log event sequence x, MIDLog first obtains its embedding sequence v, and then outputs its anomaly score $G(v)$. If $G(v)$ exceeds the predefined threshold ϕ, x is considered anomalous. We adjust the threshold ϕ on the validation set and select the best-performing threshold ϕ for evaluating the test set.

IV. Experimental Evaluation

In this section, we evaluate our approach by answering the following research questions:

RQ1: How effective is MIDLog in the ISLAD task?

RQ2: Is MIDLog reliable under supervision with different exactness labels?

RQ3: How do different modules contribute to MIDLog?

RQ4: Can MIDLog effectively deduce and distinct anomalies from bags with anomalous labels?

A. Experimental Setup

1) Datasets: We conducted comprehensive experiments on three widely used log-based anomaly detection public datasets: Spirit, Thunderbird [30], and Hadoop [31], as well as an industrial dataset System A. The Spirit dataset was gathered from a supercomputer deployed at Sandia National Labs. It includes 558 days of continuous system logs, totaling over 30 GB of log messages. The Thunderbird dataset comprises over 200 million log messages collected from a supercomputer boasting 9024 processors and 27072 GB of memory. The Hadoop dataset was gathered from a 46-core Hadoop distributed file system spanning five machines. The System A was collected from an industrial software testing platform, built on a Kubernetes-based microservices architecture, containing over ten million logs. Considering the huge scale of the Spirit and Thunderbird datasets, we followed the settings of the previous study [32], [33] and selected the earliest 1 GB and 10 million log messages from the Spirit and Thunderbird datasets, respectively, for experimentation.

2) Compared Methods: To comprehensively evaluate the effectiveness of MIDLog, we compare it with state-of-the-art deep learning log anomaly detection methods. DeepLog [7] and LogAnomaly [8] represent state-of-the-art unsupervised methods, which learn normal sequence patterns from log sequences and identify deviations from predicted results as anomalies. SemPCA [34] and Log2Graph [35] are the latest unsupervised methods, utilizing principal component analysis and graph neural networks to learn representations of normal logs, and identifying deviating representations as anomalies. LogRobust [1] and CNN [36] stand as leading supervised techniques that convert log sequences into vectors and employ classification-based methodologies for anomaly detection. Additionally, we examine PLELog [5], an advanced semi-supervised approach grounded in PU-learning. PLELog estimates probability labels for unlabeled data to generate pseudo-labels for supervised training.

3) Evaluation Metrics: Precision, Recall, and F1 Score are utilized as evaluation metrics to comprehensively assess the effectiveness of MIDLog in log-based anomaly detection. The definitions are as follows: $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, $F1 = \frac{2*Prec*Rec}{Prec+Rec}$. Additionally, to mitigate potential errors stemming from selecting score thresholds, when comparing MIDLog and its variants, AUC-ROC (Area under Curve of Receiver Operating Characteristic) and AUC-PR are employed as metrics. These metrics are not influenced by score threshold selection, enabling a more thorough evaluation. We also defined two new metrics to evaluate the degree of anomaly separation at the score level. The N/P Gap describes the anomalous separation in the test set. The PN/P Gap describes the anomalous separation of positive bags in the test set.

$$\begin{aligned} N/P\ Gap &= mean_{v^a \in V_B}(G(v^a)) - mean_{v^n \in V_B}(G(v^n)), \\ PN/P\ Gap &= mean_{v^a \in V_B^a}(G(v^a)) - mean_{v^n \in V_B^a}(G(v^n)), \end{aligned} \tag{13}$$

4) Implementation Details: To ensure the reproducibility of the study, we present specific implementation details. Following the setting of prior research [8], [33], we employ a sliding window of size 20 to perform data grouping, where each log sequence instance contains 20 log entries. Subsequently, a fixed number of consistent instances in the timeline are grouped into bags to construct the training set. This setup constrains each bag to contain consistent multiple log event sequences, thereby strictly ensuring the experiment under the ISLAD settings. All experiments utilize bag-level labels for

TABLE I
EXPERIMENTAL RESULTS ON EVALUATED DATASETS IN THE CONTEXT OF INEXACT SUPERVISION LOG-BASED ANOMALY DETECTION

Model	Spirit			Thunderbird			Hadoop			System A		
	Precision	***Recall***	***F1-Score***	***Precision***	***Recall***	***F1-Score***	***Precision***	***Recall***	***F1-Score***	***Precision***	***Recall***	***F1-Score***
PLELog	73.91	94.45	82.93	54.66	97.02	69.93	72.82	94.10	82.11	78.45	58.93	67.30
SemPCA	89.27	95.37	92.22	28.76	20.57	23.99	28.75	58.48	38.55	67.30	49.07	56.76
Log2Graph	78.29	93.40	85.18	52.23	72.16	60.60	61.60	73.18	66.90	59.64	63.02	61.29
Deeplog	88.95	93.14	91.00	64.37	68.60	66.42	44.34	77.65	56.45	73.12	57.57	64.42
LogAnomaly	89.31	93.33	91.28	66.20	64.86	65.52	70.03	70.76	70.39	68.73	68.33	68.53
CNN	47.88	99.62	64.68	10.67	98.43	19.26	28.16	99.49	43.90	32.62	96.05	48.70
LogRobust	70.59	96.71	81.61	39.23	89.88	54.62	35.35	75.69	48.19	29.98	98.28	45.94
CNN*	99.99	87.69	93.34 ± 3.30	89.32	73.48	80.48 ± 3.67	81.40	89.26	84.98 ± 4.58	69.25	80.49	74.35 ± 1.51
LogRobust*	95.42	98.48	96.91 ± 1.28	79.40	82.77	80.84 ± 2.09	83.16	83.54	83.33 ± 2.33	70.77	83.43	76.53 ± 1.20
Ours	96.55	97.84	**97.19**	82.27	91.49	**86.63**	91.37	87.76	**89.53**	85.95	74.41	**79.76**

training and instance-level for testing. Similar to most existing works [5], [8], [22], we utilize the Drain for log parsing and partitioning the dataset into training, testing, and validation sets in a 6:3:1 ratio.

Within the MIDLog architecture, the global normality encoder incorporates two layers of Bi-LSTM with a hidden layer size of 128, alongside a three-layer CNN comprising kernels of size 3, 4, and 5, with out channels set to 48. The AdmaW optimizer is employed for training optimization, with a learning rate of 1e-3 and a batch size of 32. The ATCR component is configured with 3 clustering centers, and the parameter t is set to 0.1. The hyperparameter λ for balancing the loss function of joint training is set to 0.1. Experiments are conducted on a Linux server equipped with Intel Xeon(R) Silver 4214R 2.4GHz processors and RTX3090 GPUs with 24GB memory. We employ publicly available implementations [33] and settings of baseline models for comparison. Our code is open-sourced in https://github.com/hemh02/MIDLog.

B. RQ1: How effective is MIDLog in the ISLAD task?

This RQ evaluates whether MIDLog is effective on public datasets. In order to simulate ISLAD tasks in real industrial scenarios, we set the number of log entries contained in each bag to 800. This number significantly exceeds the typical number of log entries in most log sessions [33], enabling the verification of the advantages of inexact labeling strategy in reducing annotation costs. For unsupervised and semi-supervised methods, we consider negative bags as normal data and positive bags as unlabeled data.

Additionally, we compare inexact labels with an equal number of exact labels. Specifically, we counted the number of inexact anomaly labels used and randomly selected an equal number of exact anomaly labels for training supervised models, which are labeled as LogRobust* and CNN* in Table I. Due to the randomness in label selection, we conducted five experiments and reported the mean and standard deviation of the results. Table I presents comparative results with baselines.

Generally speaking, MIDLog achieves state-of-the-art performance on the System A, Spirit, Thunderbird, and Hadoop datasets. Among the baseline models, the semi-supervised method PLELog achieves the best overall performance, followed by the unsupervised method. This occurs because semi-supervised methods utilize additional unlabeled data compared to unsupervised methods. Supervised methods yielded catastrophic results as they heavily relied on exact anomaly labels, making them unable to differentiate highly abstract anomalies under inexact labels. Consequently, supervised methods classify potentially normal instances in positive bags as anomalies, achieving high recall but at the expense of extremely low precision. Meanwhile, MIDLog surpasses all baseline models in precision under inexact label supervision, attributed to its encoding of a compact global normality representation.

Specifically, MIDLog achieves higher results than the state-of-the-art semi-supervised method PLELog by 12.46%, 14.26%, 16.70%, and 7.42%, respectively. This significant improvement suggests that MIDLog effectively utilizes inexact label information to differentiate highly abstract anomalies in the ISLAD task, as discussed in Section 3. Furthermore, compared to state-of-the-art unsupervised and supervised methods, MIDLog exhibits greater advantages, with F1 score differences of up to 20.21% and 41.34%, respectively. Moreover, MIDLog surpasses supervised methods with the same number of exact labels by up to 6.20% in terms of F1 score, which is an ideal situation pursued in industrial scenarios. This intriguing finding suggests that an equal quantity of inexact labels provides richer anomaly information compared to exact labels. This discovery highlights the potential value of the proposed inexact labeling strategy in real industrial scenarios.

C. RQ2: Is MIDLog reliable under supervision with different exactness labels?

This research question assesses the robustness of MIDLog across varying levels of label exactness. For this purpose, we redistribute the number of log entries within each bag from 100 to 2400 to change the exactness of training labels, where a higher log entry count within a bag implies lower label exactness. Figure 6 illustrates the performance of MIDLog under varying label exactness.

Overall, MIDLog exhibits robustness to label exactness, with fluctuations in AUC-PR ranging from 2.52% to 13.17% and AUC-ROC ranging from 0.74% to 3.92% across the four

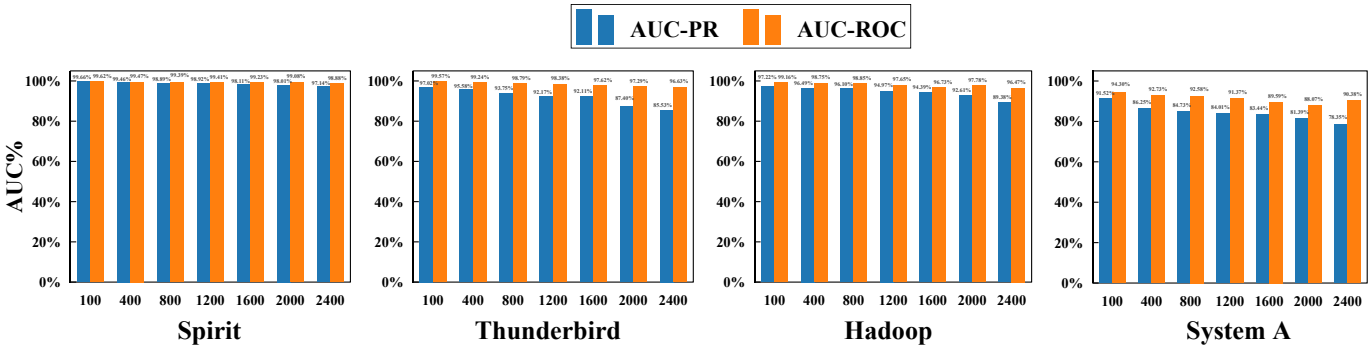


Fig. 6. Experimental results on public datasets under supervision with different exactness labels. The horizontal axis of the graph represents the number of log entries within bags in the training dataset. Higher quantities correlate with less exact anomaly labels.

datasets as label exactness varies. Even in the extreme scenario where each inexact label annotates 2400 log entries, MIDLog achieves AUC-PR values of 97.14%, 85.53%, 89.38%, and 78.35%, and AUC-ROC values of 98.88%, 96.63%, 96.47%, and 90.38% across the four datasets, still exhibiting outstanding performance. Additionally, we note that as label exactness increases incrementally, MIDLog achieves enhanced performance. When the number of log entries annotated by each inexact label decreases to 100, MIDLog even achieves an exceptional AUC-PR of 99.66% and AUC-ROC of 99.62% on the Spirit dataset. This illustrates the substantial robustness of MIDLog in addressing highly inexact anomaly labels and its capability to effectively handle ISLAD task, thereby ensuring its suitability in intricate industrial scenarios with varying levels of label exactness.

TABLE II
ABLATION OF PROPOSED THREE KEY COMPONENTS

Dataset	Model	Metric	
		AUC-PR	*AUC-ROC*
Spirit	w/o GNCL	97.68	98.98
	w/o ATCR	98.33	99.13
	w/o MIDL	93.88	97.21
	Ours	**98.89**	**99.39**
Thunderbird	w/o GNCL	91.86	97.80
	w/o ATCR	92.62	98.29
	w/o MIDL	84.42	96.18
	Ours	**93.75**	**98.79**
Hadoop	w/o GNCL	94.87	97.91
	w/o ATCR	95.33	98.44
	w/o MIDL	91.62	97.95
	Ours	**96.10**	**98.85**

D. RQ3: How do different modules contribute to MIDLog?

In this section, our goal is to investigate the contributions of three key components to MIDLog, including pre-trained global normality encoder (GNCL), adaptive MIL sampling strategy (ATCR), and multi-instance deviation learning (MIDL). Accordingly, we constructed three variants of MIDLog: MIDLog w/o GNCL, which only used the UNLN step for pre-training; MIDLog w/o ATCR, which used the vanilla MIL sampling strategy (EQ (4)) during training; and MIDLog w/o MIDL, which used a naive MIL loss for training (EQ (1)). The other settings of these variants were kept consistent with MIDLog. The ablation study is performed following the RQ1 setting, specifically with 800 log entries in each bag, with results displayed in table II.

In general, across the three experimental datasets, all components exhibited varying levels of performance degradation, thereby affirming the efficacy of the proposed components. Notably, the removal of the MIDL component has the most substantial impact. This arises from the fact that naive MIL relies solely on anomaly scores as training signals, leading to suboptimal indirect feature learning and an inability to address the ISLAD task. In contrast, MIDL extensively utilizes the normality center encoded by GNCL, facilitating the enforced separation of anomalies at both feature and score levels, thereby enabling effective anomaly separation in the ISLAD task. The effect of removing the GNCL component is marginally less significant than that of the MIDL component, suggesting that generalizing local normality to global normality is beneficial for addressing the ISLAD task by bridging the gap between feature scales and facilitating the identification of a more reliable normality center. The effect of removing the ATCR component is the least significant, as it aids in retrieving potentially challenging anomalous instances. When anomalous log patterns in the dataset are relatively straightforward, and the distribution of anomalous instances becomes more uniform, the number of challenging anomalies to separate decreases, and the naive retrieval strategy can still effectively distinguish anomalies.

The removal of the MIDL component has a significant impact, resulting in maximum reductions of 7.33% and 2.61% in AUC-PR and AUC-ROC, respectively. To delve deeper into the role of the MIDL component, we visually examine its effects separately at both the feature and score levels. Regarding the feature level, we utilize t-SNE [37] for visualizing the features encoded by the global normality encoder. Figure 7 demonstrates that the MIDL component indeed encodes a compact representation of normality, introducing a margin between normal and anomalous features, as elaborated in Section 3, which is essential for addressing the ISLAD task.

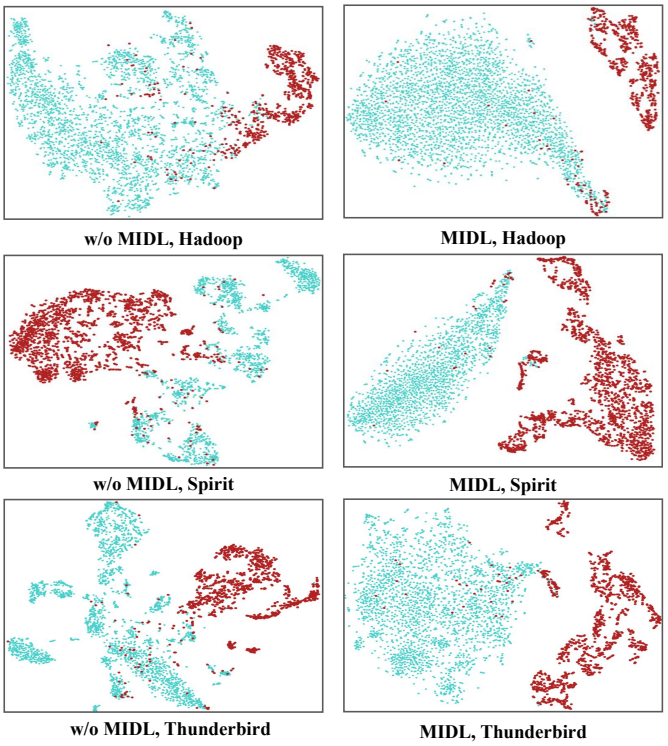


Fig. 7. Ablation of the MIDL component at the feature levels. Green points represent normal logs, while red points represent abnormal logs.

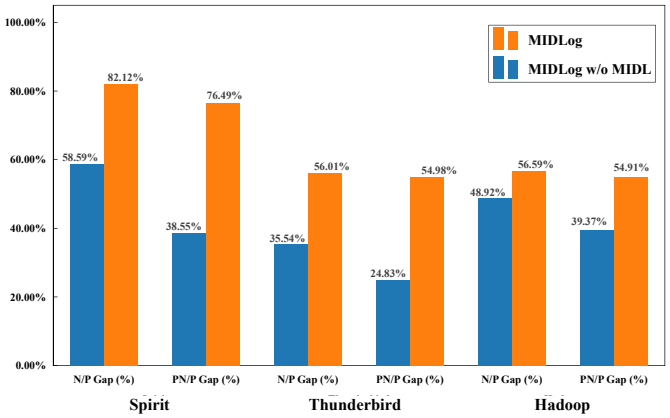


Fig. 8. Ablation of the MIDL component at the score levels on the Spirit, Thunderbird, and Hadoop datasets. The N/P Gap indicates the score gap between normal instances and abnormal instances in the test set, while the PN/P Gap represents the score gap between normal instances and abnormal instances within positive bags in the test set.

As for the score level, we illustrate the score gaps between normal and anomalous instances in the test set, as shown in figure 8. It's evident that the MIDL component establishes a margin between normal and anomalous scores, with maximum gaps of 23.53% and 37.91% for the two metrics presented in the figure. These pieces of evidence compellingly demonstrate that the MIDL component encodes a compact representation of normality and introduces margins between normal and anomalous instances at both the feature and score levels, thereby effectively tackling the ISLAD task.

E. RQ4: Can MIDLog effectively deduce and distinct anomalies from bags with anomalous labels?

In this research question, we demonstrate MIDLog's capability to detect various types of anomalies. Specifically, we collected three types of anomalies from the test sets of Spirit, Thunderbird, and Hadoop: single short-term anomalies, long-term anomalies, and multiple short-term anomalies. Subsequently, we plotted the anomaly score curves generated by MIDLog, as shown in figure 9. It is evident that MIDLog can effectively assign small scores to normal logs and large scores to anomalous logs, while maintaining a significant distinction between the scores of normal and anomalous logs. In scenarios where only normal logs are present, MIDLog consistently provides scores close to 0. Furthermore, for both long-term anomalies and multiple short-term anomalies, which are more complex than single short-term anomalies, MIDLog demonstrates accurate detection capabilities. This evidence highlights MIDLog's capability to deduce and distinct different types of anomalies from bags with anomalous labels.

F. Threats to Validity

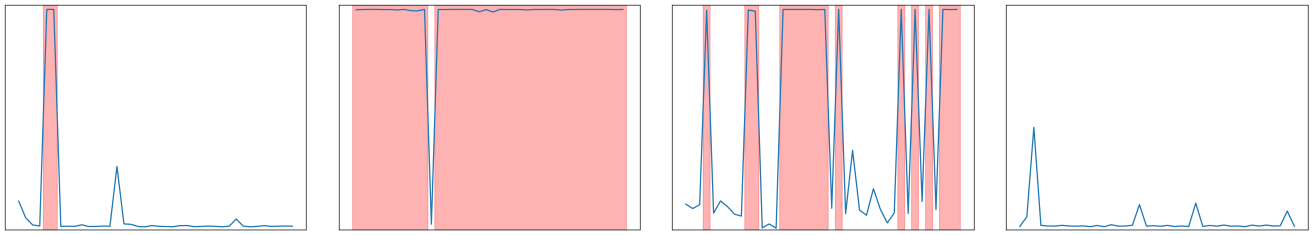
First, we did not use other commonly used datasets, such as HDFS and BGL for evaluation, because these datasets are relatively too simple [38]. These datasets have been shown to achieve an F1-score of over 95% even without providing any anomaly labels [7], [8]. As a result, the values of labels in these datasets are low thus are not suitable for evaluating models that require labeling signals.

Second, in the experiments, we grouped a fixed number of consistent log event sequences into each bag. Another grouping method is to group log event sequences within a fixed time span. Intuitively, the second method seems more suitable for simulating the real-world ISLAD task as system experts are more convenient to label a time span. However, our experiments mainly aim to evaluate the capability of MIDLog to leverage inexact labels of bags containing multiple log entries. The first method can ensure that each bag contains multiple log entries which is more feasible for evaluation. For real-world application, the usage of these two grouping methods are similar. Suppose the length of the longest human-labeled time span is L which containing N log entries. The first method groups dataset with fixed number n of log entries into each bag just ensuring $n \geq N$. The second method groups dataset with fixed length of time span l ensuring $l \geq L$.

V. Related Work

A. Log Anomaly Detection

Analyzing logs for problem detection and identification has been an active research area [1], [7], [8], [12], [17], [39]–[46]. These works first parse logs into log events, and then build anomaly detection models. Some state-of-art approaches [12], [41]–[44] extract event sequences at first, and then generate a graph-based model to compare with log sequences in a production environment to detect conflicts. Other approaches often build deep learning-based models [1], [5], [8], [17], [39], [40] to capture the sequence features of log events. Deeplog



(a) Single Short-term Anomalies (b) Long-term Anomalies (c) Multiple Short-term Anomalies (d) Normal Logs

Fig. 9. Real anomaly quantitative analysis on Spirit, Thunderbird, and Hadoop datasets. Red blocks represent the ground truth of anomalous logs, while the blue line indicates the predicted scores by MIDLog.

[7] is a classic work that utilizes the LSTM network to model the sequence of log events and the sequence of variables in log text. LogAnomaly [8] utilizes a word2vec model to transform events into vectors with semantic features to improve the anomaly detection result. LogRobust [1] utilizes TF-IDF and word vectorization to transform logs into semantic vectors. In this way, updated new logs can be transformed into semantic vectors and participate in model training and deduction. PLELog [5] proposes a semi-supervised model that leverages an unsupervised clustering method to deduce data labels and construct a supervised anomaly detection model with the deduced data labels. Some other works [47], [48] claim to have studied log anomaly detection under a weakly supervised setting. In fact, the setting in these works is equivalent to the setting in [5], which we refer to as a semi-supervised setting. Different from prior works, we focus on the log-based anomaly detection supervised by inexact labels. We propose an inexact labeling strategy that instead of labeling an log entry, system experts label a bag of log entries in a time span. To the best of our knowledge, we are the first to notice and define the inexact supervision log-based anomaly detection scenario.

B. Weakly Supervised Learning

Weakly supervised learning focuses on training machine learning models using various weak and readily available supervisory signals [49]–[51]. It primarily addresses three distinct types of supervisory signals [52], [53]: incomplete supervision, inaccurate supervision, and inexact supervision. Incomplete supervision aims to utilize unlabeled data to generate supervisory signals, a paradigm widely recognized as semi-supervised learning [5], [54], [55]. Recent methods [21], [56], [57] based on the concept of Positive-unlabeled learning, employing labeled data to estimate the labels of unlabeled data, while introducing various empirical risks to adjust for estimation biases. Inaccurate supervision accounts for situations where labels are influenced by noise and corruption [50], [58], [59]. Existing studies [60]–[63] have effectively extracted valuable label information from various noise sources by developing denoising networks, thereby reducing the impact of label noise. In this paper, we focus on inexact supervision learning.

Multi-Instance Learning (MIL) is a form of weakly supervised learning, in which data during training is structured into sets known as bags and supervised using set-level labels [18], [52], [64]–[66]. This formulation allows the utilization of inexactly labeled data, making it naturally suitable for addressing the ISLAD task. Recent MIL research has mainly focused on anomaly detection in videos [67]–[69]. Sultani et al. [19] pioneered video anomaly detection using the MIL framework, treating videos as bags and video segments as instances, and proposing a ranking loss for anomaly detection. Subsequent research by RTFM [24] followed this framework, expanding the instance retrieval strategy from top-1 to top-k and offering corresponding theoretical justifications. MIST [70] proposed a sparse continuous sampling strategy to generate more reliable instance-level pseudo-labels. Other works [71], [72] have focused on optimizing the ranking loss by imposing additional constraints on the score differences among normal instances. However, the effectiveness of these methods is only assured under the moderate assumption of video data, where anomalies are predefined, such as violent events in videos. In contrast, anomalies in log data are highly abstract and uninformed. Consequently, these methods struggle to achieve a clear separation of log anomalies and are inadequate for addressing the ISLAD task.

VI. Conclusion and Future Work

In this paper, we propose MIDLog, a novel weakly-supervised log-based anomaly detection approach based on multi-instance learning paradigm. MIDLog can deduce highly abstract anomalies by assessing their statistical deviations from normality and ensure anomaly separation in both feature representation space and anomaly score space under inexact supervision. Evaluation results on three public log datasets show that MIDLog attains an F1 score exceeding 85% when an inexact label marks 800 log entries. Besides, under inexact supervision, it even outperforms supervised methods with the same number of exact labels by 6.20%. In the future, we plan to implement inexact labeling strategies in real-world industrial systems and collecting real logs with valuable, high-quality inexact labels. We aim to create realistic and reliable public benchmarks for future research within the community.

Acknowledgment

This work was supported by National Key Laboratory of Data Space Technology and System.

REFERENCES

[1] 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.

[2] W. Xia, Y. Li, T. Jia, and Z. Wu, "Bugidentifier: An approach to identifying bugs via log mining for accelerating bug reporting stage," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2019, pp. 167–175.

[3] T. Reidemeister, M. A. Munawar, and P. A. Ward, "Identifying symptoms of recurrent faults in log files of distributed information systems," in *2010 IEEE Network Operations and Management Symposium-NOMS 2010*. IEEE, 2010, pp. 187–194.

[4] J. Tong, L. Ying, T. Hongyan, and W. Zhonghai, "An approach to pinpointing bug-induced failure in logs of open cloud platforms," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. IEEE, 2016, pp. 294–302.

[5] 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 *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1448–1460.

[6] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "Devops," *Ieee Software*, vol. 33, no. 3, pp. 94–100, 2016.

[7] 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.

[8] 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.

[9] K. Yin, M. Yan, L. Xu, Z. Xu, Z. Li, D. Yang, and X. Zhang, "Improving log-based anomaly detection with component-aware analysis," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 667–671.

[10] J. Kim, V. Savchenko, K. Shin, K. Sorokin, H. Jeon, G. Pankratenko, S. Markov, and C.-J. Kim, "Automatic abnormal log detection by analyzing log history for providing debugging insight," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, 2020, pp. 71–80.

[11] C. Zhang, T. Jia, G. Shen, P. Zhu, and Y. Li, "Metalog: Generalizable cross-system anomaly detection from logs with meta-learning," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–12.

[12] T. Jia, Y. Li, Y. Yang, and G. Huang, "Hilogx: noise-aware log-based anomaly detection with human feedback," *The VLDB Journal*, vol. 33, no. 3, pp. 883–900, 2024.

[13] M. He, T. Jia, C. Duan, H. Cai, Y. Li, and G. Huang, "Llmelog: An approach for anomaly detection based on llm-enriched log events," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2024, pp. 132–143.

[14] P. Xiao, T. Jia, C. Duan, H. Cai, Y. Li, and G. Huang, "Logcae: An approach for log-based anomaly detection with active learning and contrastive learning," in *2024 IEEE 35th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2024, pp. 144–155.

[15] C. Duan, T. Jia, H. Cai, Y. Li, and G. Huang, "Afalog: A general augmentation framework for log-based anomaly detection with active learning," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 46–56.

[16] H. Guo, J. Yang, J. Liu, J. Bai, B. Wang, Z. Li, T. Zheng, B. Zhang, J. Peng, and Q. Tian, "Logformer: A pre-train and tuning pipeline for log anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, 2024, pp. 135–143.

[17] C. Duan, T. Jia, Y. Li, and G. Huang, "Aclog: An approach to detecting anomalies from system logs with active learning," in *2023 IEEE International Conference on Web Services (ICWS)*. IEEE, 2023, pp. 436–443.

[18] M.-A. Carbonneau, V. Cheplygina, E. Granger, and G. Gagnon, "Multiple instance learning: A survey of problem characteristics and applications," *Pattern Recognition*, vol. 77, pp. 329–353, 2018.

[19] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6479–6488.

[20] 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.

[21] Z. Zhu, L. Wang, P. Zhao, C. Du, W. Zhang, H. Dong, B. Qiao, Q. Lin, S. Rajmohan, and D. Zhang, "Robust positive-unlabeled learning via noise negative sample self-correction," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3663–3673.

[22] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "Swisslog: Robust anomaly detection and localization for interleaved unstructured logs," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[23] Z. Allen-Zhu and Y. Li, "Towards understanding ensemble, knowledge distillation and self-distillation in deep learning," *arXiv preprint arXiv:2012.09816*, 2020.

[24] Y. Tian, G. Pang, Y. Chen, R. Singh, J. W. Verjans, and G. Carneiro, "Weakly-supervised video anomaly detection with robust temporal feature magnitude learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 4975–4986.

[25] W. Li and N. Vasconcelos, "Multiple instance learning for soft bags via top instances," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2015, pp. 4277–4285.

[26] G. Pang, C. Shen, and A. Van Den Hengel, "Deep anomaly detection with deviation networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 353–362.

[27] 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.

[28] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[29] H.-P. Kriegel, P. Kroger, E. Schubert, and A. Zimek, "Interpreting and unifying outlier scores," in *Proceedings of the 2011 SIAM International Conference on Data Mining*. SIAM, 2011, pp. 13–24.

[30] 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.

[31] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets for ai-driven log analytics," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 355–366.

[32] 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.

[33] V. 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*, 2022, pp. 1356–1367.

[34] L. Yang, J. Chen, S. Gao, Z. Gong, H. Zhang, Y. Kang, and H. Li, "Try with simpler-an evaluation of improved principal component analysis in log-based anomaly detection," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 5, pp. 1–27, 2024.

[35] Z. Li, J. Shi, and M. Van Leeuwen, "Graph neural networks based log anomaly detection and explanation," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, 2024, pp. 306–307.

[36] S. Lu, X. Wei, Y. Li, and L. Wang, "Detecting anomaly in big data system logs using convolutional neural network," in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*. IEEE, 2018, pp. 151–158.

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

[38] M. Landauer, F. Skopik, and M. Wurzenberger, "A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1354–1375, 2024.

[39] K. Yin, M. Yan, L. Xu, Z. Xu, Z. Li, D. Yang, and X. Zhang, "Improving log-based anomaly detection with component-aware analysis," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 667–671.

[40] J. Kim, V. Savchenko, K. Shin, K. Sorokin, H. Jeon, G. Pankratenko, S. Markov, and C.-J. Kim, "Automatic abnormal log detection by analyzing log history for providing debugging insight," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 71–80.

[41] T. Jia, Y. Wu, C. Hou, and Y. Li, "Logflash: Real-time streaming anomaly detection and diagnosis from system logs for large-scale software systems," in *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 2021, pp. 80–90.

[42] A. Nandi, A. Mandal, S. Atreja, G. B. Dasgupta, and S. Bhattacharya, "Anomaly detection using program control flow graph mining from execution logs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 215–224.

[43] T. Jia, L. Yang, P. Chen, Y. Li, F. Meng, and J. Xu, "Logsed: Anomaly diagnosis through mining time-weighted control flow graph in logs," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 447–455.

[44] T. Jia, P. Chen, L. Yang, Y. Li, F. Meng, and J. Xu, "An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services," in *2017 IEEE International Conference on Web Services (ICWS)*, 2017, pp. 25–32.

[45] X. Yang, X. Huang, C. Duan, T. Jia, S. Dong, Y. Li, and G. Huang, "Enhancing web service anomaly detection via fine-grained multi-modal association and frequency domain analysis," 2025. [Online]. Available: https://arxiv.org/abs/2501.16875

[46] W. Hong, Y. Yang, J. Wu, D. Shangguan, Y. Lai, Q. Bai, and Y. Li, "Nicsdg: A non-intrusive approach to constructing concise service dependency graphs for microservice systems," in *2024 IEEE 35th International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2024, pp. 79–84.

[47] H. Guo, Y. Guo, J. Yang, J. Liu, Z. Li, T. Zheng, L. Zheng, W. Hou, and B. Zhang, "Loglg: Weakly supervised log anomaly detection via log-event graph construction," in *International Conference on Database Systems for Advanced Applications*. Springer, 2023, pp. 490–501.

[48] T. Zhang, X. Huang, W. Zhao, G. Mo, and S. Bian, "Logcontrast: A weakly supervised anomaly detection method leveraging contrastive learning," in *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*. IEEE, 2023, pp. 48–59.

[49] Z. Ren, S. Wang, and Y. Zhang, "Weakly supervised machine learning," *CAAI Transactions on Intelligence Technology*, vol. 8, no. 3, pp. 549–580, 2023.

[50] X. Cheng, Y. Cao, X. Li, B. An, and L. Feng, "Weakly supervised regression with interval targets," in *International Conference on Machine Learning*. PMLR, 2023, pp. 5428–5448.

[51] G. Pang, C. Shen, H. Jin, and A. van den Hengel, "Deep weakly-supervised anomaly detection," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1795–1807.

[52] M. Jiang, C. Hou, A. Zheng, X. Hu, S. Han, H. Huang, X. He, P. S. Yu, and Y. Zhao, "Weakly supervised anomaly detection: A survey," *arXiv preprint arXiv:2302.04549*, 2023.

[53] H. Chen, J. Wang, L. Feng, X. Li, Y. Wang, X. Xie, M. Sugiyama, R. Singh, and B. Raj, "A general framework for learning from weak supervision," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[54] T. Sakai, M. C. Plessis, G. Niu, and M. Sugiyama, "Semi-supervised classification based on classification from positive and unlabeled data," in *International conference on machine learning*. PMLR, 2017, pp. 2998–3006.

[55] Y. Cao, Z. Wan, D. Ren, Z. Yan, and W. Zuo, "Incorporating semi-supervised and positive-unlabeled learning for boosting full reference image quality assessment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5851–5861.

[56] C. Li, Y. Dai, L. Feng, X. Li, B. Wang, and J. Ouyang, "Positive and unlabeled learning with controlled probability boundary fence," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

[57] R. Kiryo, G. Niu, M. C. du Plessis, and M. Sugiyama, "Positive-unlabeled learning with non-negative risk estimator," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, 2017, pp. 1675–1685.

[58] M. Hu, H. Han, S. Shan, and X. Chen, "Weakly supervised image classification through noise regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 517–11 525.

[59] E. Safranchik, S. Luo, and S. Bach, "Weakly supervised sequence tagging from noisy rules," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5570–5578.

[60] Y. Oh, B. Kim, and B. Ham, "Background-aware pooling and noise-aware loss for weakly-supervised semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 6913–6922.

[61] Y. Li, Y. Duan, Z. Kuang, Y. Chen, W. Zhang, and X. Li, "Uncertainty estimation via response scaling for pseudo-mask noise mitigation in weakly-supervised semantic segmentation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 36, no. 2, 2022, pp. 1447–1455.

[62] Y. Li, L. Song, and C. Zhang, "Sparse conditional hidden markov model for weakly supervised named entity recognition," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 978–988.

[63] M. He, N. Gu, Y. Shi, Q. Zhang, and Y. Chen, "Fair: A causal framework for accurately inferring judgments reversals," in *International Conference on Artificial Neural Networks*. Springer, 2023, pp. 171–182.

[64] L. Perini, V. Vercruyssen, and J. Davis, "Learning from positive and unlabeled multi-instance bags in anomaly detection," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 1897–1906.

[65] S. Park, H. Kim, M. Kim, D. Kim, and K. Sohn, "Normality guided multiple instance learning for weakly supervised video anomaly detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 2665–2674.

[66] H. Zhou, J. Yu, and W. Yang, "Dual memory units with uncertainty regulation for weakly supervised video anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 3, 2023, pp. 3769–3777.

[67] S. Li, F. Liu, and L. Jiao, "Self-training multi-sequence learning with transformer for weakly supervised video anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1395–1403.

[68] Y. Liu, J. Liu, M. Zhao, S. Li, and L. Song, "Collaborative normality learning framework for weakly supervised video anomaly detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 5, pp. 2508–2512, 2022.

[69] H. Park, J. Noh, and B. Ham, "Learning memory-guided normality for anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 14 372–14 381.

[70] J.-C. Feng, F.-T. Hong, and W.-S. Zheng, "Mist: Multiple instance self-training framework for video anomaly detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 14 009–14 018.

[71] J. Zhang, L. Qing, and J. Miao, "Temporal convolutional network with complementary inner bag loss for weakly supervised anomaly detection," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 4030–4034.

[72] B. Wan, Y. Fang, X. Xia, and J. Mei, "Weakly supervised video anomaly detection via center-guided discriminative learning," in *2020 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2020, pp. 1–6.