2025 40th IEEE/ACM International Conference on Automated Software Engineering (ASE)

LogAction: Consistent Cross-system Anomaly Detection through Logs via Active Domain Adaptation

Chiming Duan[x†‡], Minghua He[†‡], Pei Xiao[†‡], Tong Jia[§‖*], Xin Zhang[††], Zhewei Zhong[††], Xiang Luo[††],
Yan Niu[††], Lingzhe Zhang[†], Siyu Yu[†], Yifan Wu[†], Weijie Hong[†], Ying Li[†¶*] and Gang Huang[‖]
[†]School of Software and Microelectronics, Peking University, Beijing, China
{duanchiming, hemh2120, xiaopei, zhang.lingzhe, hongwj}@stu.pku.edu.cn, gaiusyu6@gmail.com, {yifanwu, li.ying}@pku.edu.cn
[§]Institute for Artificial Intelligence, Peking University, Beijing, China
jia.tong@pku.edu.cn
[¶]National Engineering Research Center for Software Engineering, Peking University, Beijing, China
li.ying@pku.edu.cn
[‖]National Key Laboratory of Data Space Technology and System, Beijing, China
{jia.tong, hg}@pku.edu.cn
[††]Bytedance, Beijing, China
{zhangxin.11, zhongzhewei, luoxiang.10, niuyan.13}@bytedance.com

***Abstract*—Log-based anomaly detection is a essential task for ensuring the reliability and performance of software systems. However, the performance of existing anomaly detection methods heavily relies on labeling, while labeling a large volume of logs is highly challenging. To address this issue, many approaches based on transfer learning and active learning have been proposed. Nevertheless, their effectiveness is hindered by issues such as the gap between source and target system data distributions and cold-start problems. In this paper, we propose *LogAction*, a novel log-based anomaly detection model based on active domain adaptation. *LogAction* integrates transfer learning and active learning techniques. On one hand, it uses labeled data from a mature system to train a base model, mitigating the cold-start issue in active learning. On the other hand, *LogAction* utilize free energy-based sampling and uncertainty-based sampling to select logs located at the distribution boundaries for manual labeling, thus addresses the data distribution gap in transfer learning with minimal human labeling efforts. Experimental results on six different combinations of datasets demonstrate that *LogAction* achieves an average 93.01% F1 score with only 2% of manual labels, outperforming some state-of-the-art methods by 26.28%. Website: https://logaction.github.io**

***Index Terms*—Anomaly Detection, Log Analysis, Active Learning, Domain Adaptation.**

I. Introduction

Software systems are becoming increasingly large and complex and are subject to more failures. As system logs record system states and significant 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.

Existing log-based anomaly detection models can be broadly divided into two categories: unsupervised models and supervised models. Unsupervised models [1]–[8] utilize sequential neural networks such as LSTM, GRU, etc. to learn the occurrence possibility of log events in normal event sequences to predict subsequent log events and identify the unmatched log event as an anomaly. Their effectiveness is very limited due to the lack of supervision of anomalous logs [9]. Supervised models [10]–[13] build classification models to identify anomalous logs and are more effective compared to unsupervised models. However, their effectiveness heavily relies on large amounts of labeled logs.

In real-world software systems, identifying anomalous logs poses a significant challenge due to the vast volume of system logs in which such anomalies are deeply buried. Consequently, obtaining accurate data labels, especially for anomalous logs, is a rare and difficult task [14]. To solve this problem, two different ideas are proposed. First, transferring abundant historical labeled logs from other mature systems (source systems) to new systems (target systems) with a few labeled logs for model training, namely transfer-learning-based methods [15], [16]. These methods first use the log sequences from the source systems to train an anomaly-detection model and then fine-tune it using log sequences from the target systems. These works have two main issues. First, their applicability is confined to scenarios where the data distribution gap of log sequences in source and target systems is small. For instance, LogTransfer [15] can only handle scenarios that the source system and target system belong to the same type of switch logs or software family (e.g., Hadoop application and Hadoop file system) which severely

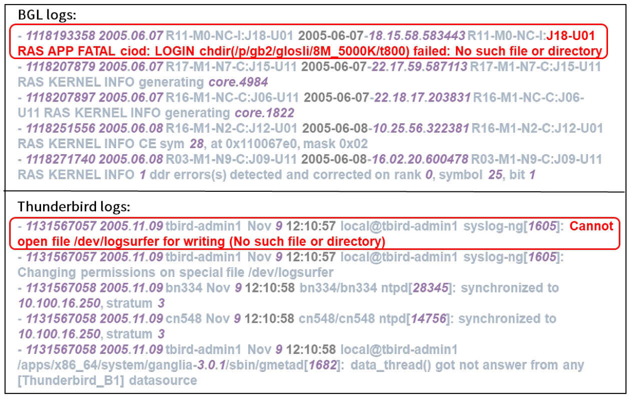
[x] Work was done when Chiming was an intern at Bytedance.
[‡] Equally Contribution.
[*] Corresponding author.

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00064

limit its availability. Second, their utilization of labels from the new system is inadequate. For instance, MetaLog [17] utilizes a meta-learning paradigm to enhance the model's generalization ability, constructing anomaly detection models for new systems. However, due to the lack of effective label utilization, substantial human labeling efforts is required to learn the complete data distribution in the new system.

Second, automatically and actively selecting the most important logs for humans to label, namely active-learning-based methods [18]–[21]. These works leverage active learning, enabling models to actively select samples for human annotations, in order to achieve highly efficient anomaly detection models with minimal human labeling efforts. However, existing active-learning-enabled works suffer from a cold start problem. This implies that the model's ability is heavily dependent on the accumulation of online human labels, particularly those associated with anomalous logs. For instance, ACLog [18] necessitates 205 online labels of logs from the BGL dataset [22]. Accumulating a sufficient number of labels requires a significant amount of time, during which the model's performance remains limited.

In our view, transfer-learning-based methods aim to increase the amount of labeled data by leveraging external labeled data from other systems, while active-learning-based methods aim to increase the quality of labeled data by carefully selecting data instances to be labeled. They consider solving the label lacking problem from different aspects and can benefit each other. From the perspective of transfer-learning-based methods, active learning can help use data labels of key data instances to bridge the gap between different systems. From the perspective of active-learning-based methods, transfer learning can help solve the cold start problem.



```
BGL logs:
- 1118193358 2005.06.07 R11-M0-NC-I:J18-U01 2005-06-07-18.15.58.583443 R11-M0-NC-I:J18-U01
RAS APP FATAL ciod: LOGIN chdir(/p/gb2/glosli/8M_5000K/t800) failed: No such file or directory
- 1118207879 2005.06.07 R17-M1-N7-C:J15-U11 2005-06-07-22.17.59.587113 R17-M1-N7-C:J15-U11
RAS KERNEL INFO generating core.4984
- 1118207897 2005.06.07 R16-M1-NC-C:J06-U11 2005-06-07-22.18.17.203831 R16-M1-NC-C:J06-
U11 RAS KERNEL INFO generating core.1822
- 1118251556 2005.06.08 R16-M1-N2-C:J12-U01 2005-06-08-10.25.56.322381 R16-M1-N2-C:J12-U01
RAS KERNEL INFO CE sym 28, at 0x110067e0, mask 0x02
- 1118271740 2005.06.08 R03-M1-N9-C:J09-U11 2005-06-08-16.02.20.600478 R03-M1-N9-C:J09-U11
RAS KERNEL INFO 1 ddr errors(s) detected and corrected on rank 0, symbol 25, bit 1

Thunderbird logs:
- 1131567057 2005.11.09 tbird-admin1 Nov 9 12:10:57 local@tbird-admin1 syslog-ng[1605]: Cannot
open file /dev/logsurfer for writing (No such file or directory)
- 1131567057 2005.11.09 tbird-admin1 Nov 9 12:10:57 local@tbird-admin1 syslog-ng[1605]:
Changing permissions on special file /dev/logsurfer
- 1131567058 2005.11.09 bn334 Nov 9 12:10:58 bn334/bn334 ntpd[28345]: synchronized to
10.100.16.250, stratum 3
- 1131567058 2005.11.09 cn548 Nov 9 12:10:58 cn548/cn548 ntpd[14756]: synchronized to
10.100.16.250, stratum 3
- 1131567058 2005.11.09 tbird-admin1 Nov 9 12:10:58 local@tbird-admin1
/apps/x86_64/system/ganglia-3.0.1/sbin/gmetad[1682]: data_thread() got not answer from any
[Thunderbird_B1] datasource
```

Fig. 1: Two log sequences from different systems (BGL and ThunderBird). Although they express the same error - file or directory does not exist, their formats show distinct differences.

As a result, we pose an idea that transfer learning and active learning should be combined together to solve the label lacking problem. We define this scenario as consistent cross-system anomaly detection (CCAD), that is, leveraging the features extracted from abundant historical labeled logs of mature systems (source systems) to build anomaly detection models for new systems (target systems) and consistently optimize the models with online human labels on the target systems. In this paper, we focus on the CCAD scenario and aim to build a high performance anomaly detection model without or with very few anomalous labels.

However, achieving this is not easy. First, bridging the huge data distribution gaps between source system and target systems is challenging. Logs generated by different systems exhibit varying formats and content. As illustrated in Figure 1, logs originating from two distinct service systems both convey the same error, 'file not exist,' yet they significantly differ in their log formats and content. Moreover, the types and frequencies of anomalies vary across different systems, further amplifying the data distribution gaps. Second, accurately selecting the least but most useful logs for human labeling from huge amount of system logs is challenging. Throughout system operations, thousands of logs can be generated per second. The human effort to label every log is remarkably high. Furthermore, these logs frequently exhibit substantial redundancy, with the information contained within this redundancy often tracing back to previously labeled logs. So, the selection of the most useful logs for human labeling is crucial. These logs should encompass distributional information that hasn't been captured in any historically labeled logs, aiming to minimize the human labeling efforts. Facing these challenges, in this paper, inspired by [23], we propose *LogAction*, a consistent cross-system anomaly detection model via active domain adaptation. Active domain adaptation, a fusion of active learning and transfer learning techniques, has demonstrated its effectiveness in various studies, particularly in reducing the human labeling efforts [23]–[26].

To handle the first challenge, we use global embedding and contrastive learning. We employ a pretrained BART model [27] to extract meaning from diverse log formats, converting them into word vectors in a unified space. These word vectors are then turned into log sequences using time windows. Subsequently, we utilize contrastive learning to encode log sequences from diverse systems into log vectors with similar distributions, effectively minimizing the data distribution gaps across different systems. This allows knowledge from the source system to be applied to the target system, enabling us to create a base anomaly detection model for the target using the source's log vectors.

For the second challenge, we turn to active learning. This helps us select the most informative yet fewest log vectors in the target system for human labeling to improve the base model. We employ two sampling methods, free energy-based sampling and uncertainty-based sampling, to collect log vectors that are distinct from both historically labeled source log vectors and those previously labeled during selection. In our view, these log vectors holds the most informative content. On the one hand, these log vectors contain information the model hasn't encountered before, making them more valuable for the model's learning. On the other hand, these log vectors, distinct from those of the source system, encapsulate information regarding the gap in data distribution between the source and target systems. Learning from this subset of log vectors facilitates further mitigation of the distributional gaps between the source and target systems.

We validated the efficacy of *LogAction* across three distinct public datasets, each representing a completely different system. The experiments demonstrate that our approach surpasses 93% F1-score performance with the utilization of only 2% labeled data in target system. Besides, our approach outperforms state-of-art models that can use for CCAD task by 26.28% on average. The contribution of this paper are as follows:

1) We propose *LogAction*, a novel generalizable anomaly detection approach via active domain adaptation.
2) We utilize global embedding and contrastive learning techniques to mitigate the data distribution gap, while employing active learning to reduce human labeling efforts, effectively addressing the two primary challenges in CCAD task.
3) Evaluation results on three open datasets show the significant effectiveness of our approach.

II. Related Work

A. Log-based anomaly detection

Timely anomaly detection plays a vital role in ensuring system reliability [28]–[33]. Analyzing logs for problem detection and identification has been an active research area [1]–[4], [10], [34]–[42]. These work first parse logs into log events, and then build anomaly detection models. Some state-of-art approaches [34]–[42] extract event sequence at first, and then generate a graph-based model to compare with log sequences in production environment to detect conflicts. Other approaches often build deep learning-based models [1]–[5], [10], [13] to capture the sequence features of log events. Deeplog is a typical work [1] that utilizes LSTM network [43] to model the sequence of log events and the sequence of variables in log text. LogAnomaly [2] utilizes a word2vec model to transform events into vectors with semantic features to improve the anomaly detection result. LogRobust [10] 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. LogTransfer [15] and LogTAD [16] use transfer learning to achieve the cross-system anomaly detection. MetaLog [17] use meta-learning paradigm to enhance the model's generalization ability, constructing anomaly detection models for new systems. However, their effectiveness is constrained by the data distribution gap between domains and the insufficient utilization of labels. As a result, they are not suitable for addressing the CCAD task.

B. Active domain adaptation

Active domain adaptation (ADA) combines the principles of active learning and transfer learning, enabling a model to actively select the valueable samples during transfer to the target domain, thereby reducing the need for human labeling efforts. Rai et al. [24] were the first to demonstrate the synergistic relationship between active learning and transfer learning, prompting various related works. Recently, AADA [44] has employed adversarial training and ADA to address domain shift issues. Bo et al. [26] utilized Transferable Query Selection (TQS) for sample selection, aiming to overcome the limitations of single-domain active learning in ADA. Han et al. [25] improved the performance of active domain adaptation by employing loss-based querying for sample selection. EADA [23] employed an energy-based model for sampling, considering both sample entropy and uncertainty. These approaches have exhibited promising performance in tasks such as image classification and successfully inspire us to integrate active learning and transfer learning to address the CCAD tasks.

III. Approach

In this paper, we introduce *LogAction*, an innovative approach based on active domain adaptation, specifically designed to address the two key challenges in CCAD tasks as defined in Section I. Existing works often inadequately address these challenges, resulting in their subpar performance. In contrast, after log parsing, *LogAction* utilizes global embedding and contrastive learning to map the log sequences of the source and target systems into similar distributions, aiming to mitigate the data distribution gaps. Then, *LogAction* leverages a substantial amount of labeled data from the source system to train the anomaly detection model and fine-tunes it on a high-information subset selected through active learning. In this way, we can obtain a generalized model and make it applicable to anomaly detection tasks in the target system with minimal human labeling efforts. In this section, we will introduce our model. In Section III-A, we provided an overview of our model. In Sections III-B and III-C, we will separately discuss the two primary components of our model: Encoding and Active Domain Adaptation.

A. Overview

Figure 2 illustrates the overview of the *LogAction*. *LogAction* includes three main phases: Log Parser, Encoding and Active Domain Adaptation. In the Log Parser phase, we employ the advanced log parsing method Drain [45] to obtain templates of log events. Subsequently, we utilize the pre-trained BART model [27] to extract the semantic information from these log event templates, transforming them into word vectors. Pre-trained models have demonstrated superior semantic extraction capabilities compared to word embedding models like GloVe [46] and Word2Vec [47]. Afterward, we employ a sliding time window to split the log event templates represented by word vectors, forming the log sequences. As illustrated in Figure 2, each log sequence is a two-dimensional matrix, where each row represents a word vector of a log event. In the Encoding phase, we encode the log sequences from both the source system and the target system, mapping them to similar distribution. The resulting encoded log sequences are referred to as log vectors. We utilize contrastive learning to train the encoder, treating normal log sequences from both the source system and the target system as one class and anomalous log sequences as another. During the *LogAction* process, the encoder is trained jointly with the downstream anomaly detection model, and the labels come from active learning processes. A detailed description of our encoder implementation will be provided in Section III-B. In the Active Domain Adaptation phase, we initially use labeled log vectors from the source system to train an anomaly detection model called the ”Classifier(source)”. Because it is trained on log vectors from a source system with similar distributions to target system, the Classifier(source) possesses a certain degree of generalization. However, to further tailor it for the target system, we require some labeled data from the target system to fine-tune it. To minimize the need for human labeling during fine-tuning, we utilize active domain adaptation techniques to accomplish this phase. We primarily employ two modes of sampling using active learning: free energy-based sampling and uncertainty-based sampling. In the free energy-based sampling phase, we choose log vectors with the highest free energy deviation from the source system. In the uncertainty-based sampling phase, we select log vectors situated at the boundary between normal and anomalous log vector classifications. Subsequently, we manually label the log vectors chosen through active learning and fine-tune Classifier(source) into Classifier(target) using them to complete the domain adaptation process. Detailed insights into our Active Domain Adaptation approach will be provided in Section III-C.

B. Encoding

After log parsing, diverse formats of raw logs are parsed into log sequences within the same feature space. However, the fault-tolerance mechanisms and structural differences contribute significantly to variations in error categories and frequencies across different systems. As a result, significant data distribution gaps still exist in the log sequences between the source system and the target system. So, we utilize contrastive learning to map the log sequences from the source system and the target system into log vectors, aligning their distributions to further mitigate the data distribution gaps.

Figure 3 illustrates the encoding process, wherein we consider the normal log sequences from the source system and the target system as one class (Class 0), while treating the anomalous log sequences from the source and target systems as another class (Class 1). By employing contrastive learning, we aim to minimize the intra-class distance and maximize the inter-class distance. Specifically, the input to the encoder is a log sequence $S = \{s_1, s_2, ..., s_t\}$, which has undergone word embedding processing. Here, s represents a embedded log event, and t denotes the time window size. The class of each log sequence is annotated by $y \in \{0, 1\}$, when $y = 0$, it indicates that the log sequence S belongs to Class 0, and when $y = 1$, it signifies that the log sequence S belongs to Class 1. We employ the encoder to encode the log sequence S into a log vector $V = \{v_1, v_2, ..., v_r\}$, effectively modeling the distribution $V = p_\theta(S)$, where θ denotes the parameters of the encoder and r denotes the encoding dimensionality. During training, we construct a discriminator to identify the class to which the encoded result V belongs, modeling the distribution $y' = q_\phi(V)$. The objective of the discriminator is to correctly identify the class to which the encoded log vector belongs. Our intuition is that if the encoded results V can concentrate in the feature space based on their respective classes, they should be more easily distinguishable by the discriminator. As illustrated in Figure 4, the encoder’s architecture comprises two LSTM layers and a fully connected layer. The log vector V serves as the input, and the output being $y' \in [0, 1]$, where y' and $1-y'$ represent the probabilities of V belonging to Class 0 and Class 1, respectively. We utilize cross-entropy loss to assess the encoding performance, given by

$$\mathcal{L} = \frac{1}{N} \sum_{i}^{N} - \left[y_i \cdot \log(y'_i) + (1 - y_i) \cdot \log(1 - y'_i) \right], \quad (1)$$

In this context, N represents the number of training log sequences. We employ L to update our parameters θ and ϕ. Specifically, our training objective is to minimize the following objective function:

$$\begin{aligned} \theta, \phi &= \underset{\theta,\phi}{argmin}\{\mathcal{L}\}, \\ &= \underset{\theta,\phi}{argmin}\{\frac{1}{N} \sum_{i}^{N} - \left[y_i \cdot \log(y'_i) + (1 - y_i) \cdot \log(1 - y'_i) \right]\}, \\ &= \underset{\theta,\phi}{argmin}\{\frac{1}{N} \sum_{i}^{N} - [y_i \cdot \log(q_\phi(p_\theta(S_i))) \\ &\qquad\qquad + (1 - y_i) \cdot \log(1 - q_\phi(p_\theta(S_i)))]\}, \end{aligned} \quad (2)$$

After training, we utilize the encoder to encode the log sequence S, and employ the encoded log vector V to accomplish the active domain adaptation process.

C. Active Domain Adaptation

After encoding, we utilize the log vectors from the source system to train a base anomaly detection model, referred to as Classifier(Source). Subsequently, we consistently optimize the Classifier(Source) using online human labels from the target systems. In order to reduce the human labeling efforts, we employ an active learning approach, selecting subsets that provide the most useful information. In our view, the most valuable log vectors would be those that have not been learned by the model. The information within them should differ from the historically labeled log vectors from the source system and those already labeled during the active learning process. Accordingly, we employ two selection strategies to meet this requirement: free energy-based sampling and uncertainty-based sampling, as shown in Algorithm 1.

In the phase based on free energy-based sampling, we select the log vectors that differ from the historically labeled log vectors in source system. Specifically, we model the log vectors from the source system as a Gaussian distribution. In this Gaussian distribution, log vectors with high free energy imply that they deviate from the center and reside at the

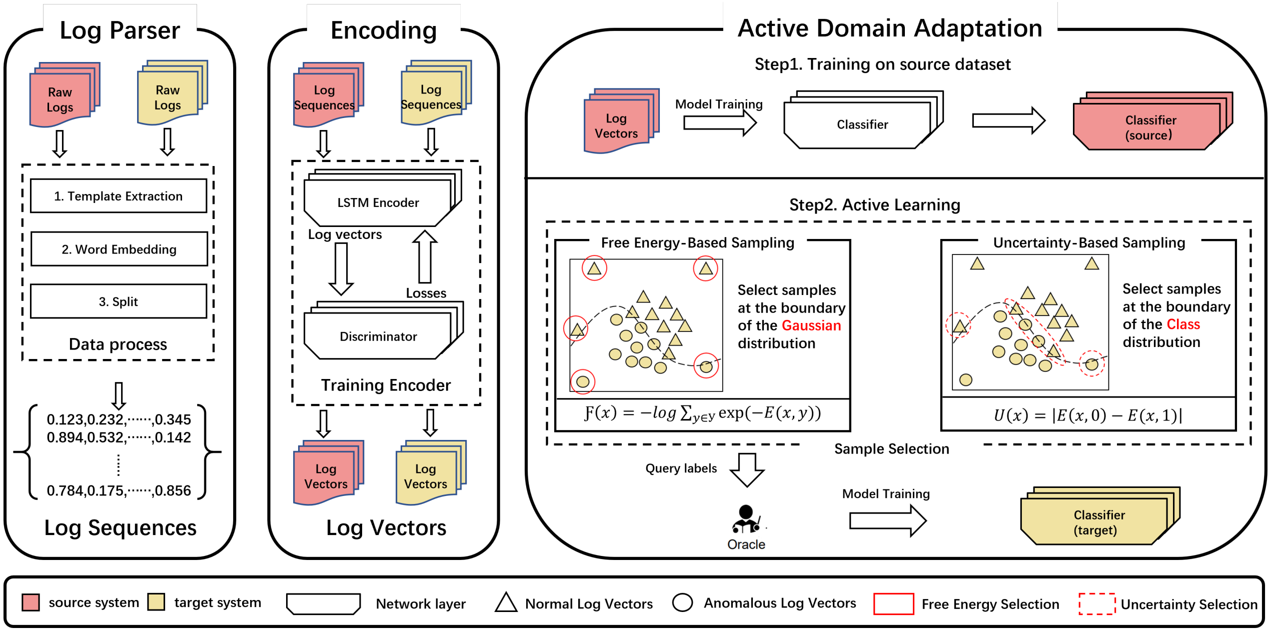


Fig. 2: The overview of *LogAction*. *LogAction* includes three main phases: Log Parser, Encoding and Active Domain Adaptation. Firstly, the raw system logs are labeled and parsed into log event sequences. Secondly, *LogAction* encodes the log sequences from both the source system and the target system, mapping them to similar distribution. Finally, *LogAction* is initially trained using labeled log vectors from the source system. Subsequently, it is fine-tuned with a very limited amount of target system logs via active learning to adapt to cross-system anomaly detection.

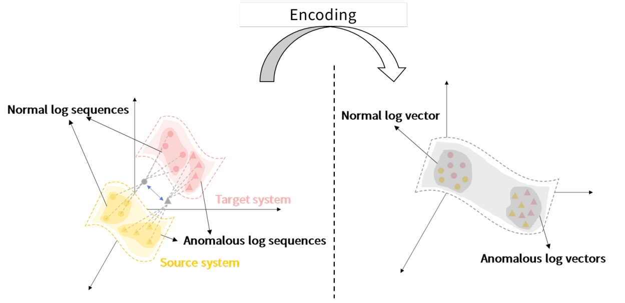


Fig. 3: The Encoding Phase. In the initial stage, log sequences from the source system and the target system originate from distinct distributions (located in different hyperplanes). Employing contrastive learning, the objective is to map the distributions of normal log sequences from the source system and normal as well as anomalous log sequences from the target system to similar distributions (situated within the same hyperplane).

boundary of the distribution. In other words, the rationale behind free energy-based sampling is to use Gaussian distributions to model log vector distributions. By labeling and subsequently training on log vectors that fall within the distribution gap, *LogAction* can effectively reduce the distribution gap between the source and target systems, thereby facilitating model transfer. Therefore, we selecting the target system's log vectors that has the highest free energy in Gaussian distribution for human labeling. These log vectors differ from historical log vectors in source system. Moreover, this subset of log vectors exhibits a distribution distinct from that of the source system, occupying the data distribution gap between the source and target systems. Learning from this specific set of data facilitates further reduction in the distribution gap between the systems. Inspired by EADA [23], we use the energy-based model (EBM) [48] to build the Gaussian distribution for source system log vectors. Generally, we employ an EBM

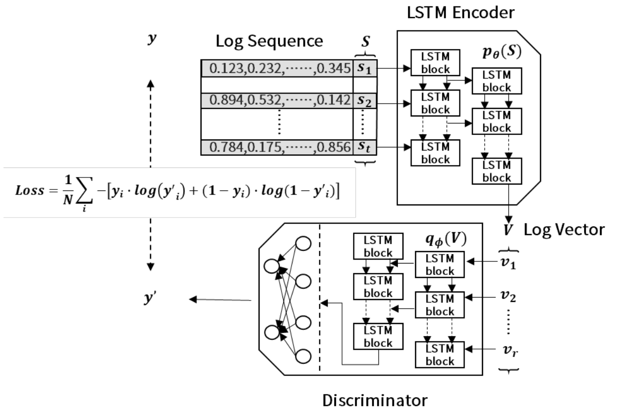

LSTM Encoder
y
Log Sequence
S
0.123,0.232,······,0.345 s_1
0.894,0.532,······,0.142 s_2
0.784,0.175,······,0.856 s_t
LSTM block
$p_\theta(S)$
$Loss = \frac{1}{N}\sum_i -[y_i \cdot log(y'_i) + (1-y_i) \cdot log(1-y'_i)]$
V Log Vector
$q_\phi(V)$
v_1
v_2
v_r
y'
Discriminator

Fig. 4: The overview of encoder.

Algorithm 1 Sample Selection

```
Input: The pool of unlabeled log vectors from the target system $\mathcal{V}_t$, Classifier(source) $E$
function CalculateFreeEnergy($V$,$E$)
    $\mathcal{F}(V) \leftarrow -log \sum_{l \in \{0,1\}} exp(-E(V,l))$   ▷ Calculate Free Energy
    return $\mathcal{F}(V)$   ▷ Output: Free Energy for vector $V$
end function
function CalculateUncertainty($V$,$E$)
    $P(V,0) \leftarrow \frac{E(V,0)}{\sum_{l' \in \{0,1\}} E(V,l)}$
    $P(V,1) \leftarrow \frac{E(V,1)}{\sum_{l' \in \{0,1\}} E(V,l)}$
    $U(V) \leftarrow |P(V,0) - P(V,1)|$   ▷ Calculate uncertainty
    return $U(V)$   ▷ Output: Uncertainty for vector $V$
end function
function SampleSelection($\mathcal{V}_t$,$E$,$\Delta_1$,$\Delta_2$)
    $\mathcal{F}_{set} \leftarrow \emptyset$
    $U_{set} \leftarrow \emptyset$
    for $V_i$ in $\mathcal{V}$ do   ▷ Iterate through log vectors
        $\mathcal{F}_{set}[i] \leftarrow$ CalculateFreeEnergy($V_i$,$E$)
        $U_{set}[i] \leftarrow$ CalculateUncertainty($V_i$,$E$)
    end for
    $\mathcal{V}_t^1 \leftarrow$ Sampling the maximum free energy though $\mathcal{F}_{set}$ within $\mathcal{V}$ at a sampling rate of $\Delta_1$
    $\mathcal{V}_t^2 \leftarrow$ Sampling the maximum uncertainty though $U_{set}$ within $\mathcal{V}_t^1$ at a sampling rate of $\Delta_2$
    return $\mathcal{V}_t^2$
end function
Output: Sampling subset $\mathcal{V}_t^2$ utilized for manual labeling
```

as Classifier(source) and train it using log vectors V from the source system alongside corresponding labels $l \in \{0,1\}$. Here, $l = 0$ signifies a normal log vector, and $l = 1$ represents an anomalous log vector. Classifier(source) takes log vectors V as input and produces the energy $E(V,0)$ and $E(V,1)$ for the normal and anomalous distributions, respectively. The training objective of Classifier(source) aims to minimize the energy assigned for correct classification. In other words, for an input log vector V and label l, we aim to satisfy the condition:

$$l = argmin_{l' \in \{0,1\}} E(V,l'), \tag{3}$$

After training, we can employ Energy-Based Models (EBMs) to model the joint distribution, $p(V,l)$, of the input vector V and the corresponding label l:

$$\begin{aligned} p(V,l) &= exp(-E(V,l))/Z, \\ Z &= \sum_{V \in \mathcal{V}} \sum_{l \in \{0,1\}} exp(-E(V,l)), \end{aligned} \tag{4}$$

Correspondingly, the marginal distribution of V can be obtained as follows:

$$p(V) = \sum_{l \in \{0,1\}} p(V,l) = \sum_{l \in \{0,1\}} exp(-E(V,l))/Z, \tag{5}$$

Simultaneously, $p(V)$ can be represented by the free energy $\mathcal{F}(V)$ of V:

$$p(V) = \frac{exp(-\mathcal{F}(V)}{\sum_{V \in \mathcal{V}} exp(-\mathcal{F}(x))}, \tag{6}$$

So, according to Equations 5 and 6, we can calculate the free energy of the Gaussian distribution for the input log vector V:

$$\mathcal{F}(V) = -log \sum_{l \in \{0,1\}} exp(-E(V,l)), \tag{7}$$

According to Equation 7, we calculate the free energy $\mathcal{F}(V)$ within the target system's unlabeled log vectors, and by selecting a subset of log vectors with high free energies, we accomplish the free energy-based sampling part.

The rationale behind uncertainty-based sampling is to label log vectors for which the model exhibits the greatest difficulty in making accurate predictions. By learning through these challenging log vectors, *LogAction* can improve the performance of the anomaly detection model on the target system. In the uncertainty-based sampling phase, we select log vectors that are distinct from those already labeled in the active learning process. More precisely, we choose the target system's log vectors positioned at the boundary between normal and anomalous classifications for manual labeling. These instances signify scenarios the model hasn't encountered previously, as the model would otherwise confidently classify them into their respective categories. Specifically, for a log vector V in the pool of unlabeled log vectors, the Classifier(source) can provide the probabilities $P(V,0)$ and $P(V,1)$ for V belonging to normal log sequences and anomalous log sequences, respectively:

$$\begin{aligned} P(V,0) &= \frac{E(V,0)}{\sum_{l' \in \{0,1\}} E(V,l)}, \\ P(V,1) &= \frac{E(V,1)}{\sum_{l' \in \{0,1\}} E(V,l)}, \end{aligned} \tag{8}$$

If $P(V,0)$ and $P(V,1)$ are closer, it indicates that the model lacks confidence in classifying the log vector V. Therefore, we calculate the absolute difference between $P(V,0)$ and $P(V,1)$ to determine the uncertainty $U(V)$ of the model for the log vector V:

$$U(V) = |P(V,0) - P(V,1)|, \tag{9}$$

We make selections based on log vectors' uncertainty and free energy, and employ fine-tuning of the selected vectors for the Classifier(source). This adaptation aims to enhance its performance in the target system's log anomaly detection task. Ultimately, we obtain a Classifier(target) with improved generalization capabilities.

IV. Experiment

Ours experiments focus on the following research questions (RQS):

- **RQ1:** How does *LogAction* performs in terms of effectiveness?

- **RQ2:** How does the varying number of labels impact the effectiveness of the *LogAction*?
- **RQ3:** Does each main component contribute to *LogAction*?

A. Dataset and Experiment Settings

We conducted our experiments on three public log datasets from Loghub [49]: BGL, Thunderbird, and Zookeeper. **BGL** refers to an open dataset comprising logs gathered from a BlueGene/L supercomputer system [22]. **Thunderbird** contains logs obtained from a Thunderbird supercomputer system [22], with 9,024 processors and 27,072GB memory. **ZooKeeper** contains logs amassed by aggregating data from the ZooKeeper service [50] within a lab at CUHK. These log datasets originate from three entirely distinct systems. We designate one dataset as the source system and another as the target for experimentation, resulting in a total of six combinations. We split each set of data into training and testing sets with a 7:3 ratio. To prevent data leakage, we strictly split the training and testing sets based on time, ensuring a certain time gap between them to avoid any data leakage. When the dataset serves as the target system, the training set becomes the sample pool for selection. The detailed contents of each dataset are outlined in Table I.

Our code can be accessed via https://anonymous.4open.science/r/LogAction-B821. The experiments were carried out on a Linux server equipped with an Intel(R) Xeon(R) Gold 6126T 2.60GHz CPU, 256GB of memory, and four RTX A4000 GPUs with 128GB of GPU memory. During training, we use the Adam optimizer with a learning rate of 1×10^{-3}. The number of epochs is set to 60, and the batch size is 512. The dimensional settings of the model layers, such as the hidden size of the LSTM in the encoder, are adopted from previous seminal works [1], [17]. These parameters remain fixed throughout all experiments without modification. In the active domain adaptation process, the energy alignment weight is set to 0.01, the energy-based sampling rate to 0.1, and the active ratio to 0.01.

TABLE I: Summary of the BGL , ThunderBird, and Zookeeper dataset

Dataset	**Training Set / Sample Pool**		**Testing Set**	
	#Normal	**#Anomalous**	**#Normal**	**#Anomalous**
BGL	26367	8633	11301	3699
ThunderBird	33205	1795	14231	769
Zookeeper	25412	614	10892	262

B. Competitors and Implementation Details

For RQ1, we opted for several state-of-the-art transfer-learning-based methods, active-learning-based methods, and some deep learning methods for comparison against *LogAction*. For RQ2, we investigated the sensitivity of *LogAction* to varying levels of human labeling by setting different quantities of manual labels. For RQ3, we delve into the specific roles of each main componentwithin *LogAction*. We conducted a dissection of *LogAction* by separately dismantling its transfer learning, active learning and tow sampling strategy components, resulting in four variants. Through comparison, we explored the individual impacts of each component. Below is an introduction to each of the competitors.

TABLE II: Hyperparameters

Parameter Category	**Parameter Name**	**Value**
Training Configuration	epoch	60
	batch size	512
	learning rate	10^{-3}
Encoder	LSTM layer	2
	LSTM hidden size	512
Anomaly Detection Model	Input Size	512
	Hidden size	64
	Layer size	64
Active Domain Adaptation	Energy align weight	0.01
	First sample ratio	0.1
	Active ratio	0.01

1) Transfer-learning-based methods:

- **LogTransfer** [15] is a supervised transfer learning method. It involves training a model on the source system using a shared network and fine-tuning the first half of the network with the target system logs to complete the transfer learning process.
- **LogTAD** [16] is an unsupervised transfer learning method. It learns the hypersphere center of normal logs from the source system's logs and utilizes normal logs from the target system to transfer hypersphere center, thereby achieving the purpose of model migration.
- **MetaLog** [17] is a generalizable cross-system anomaly detection approach, it utilizes a meta-learning paradigm to enhance the model's generalization ability, constructing anomaly detection models for new systems with limited labels.

2) Active-learning-based methods:

- **ACLog** [18] utilizes normal logs to train an unsupervised model and employs active learning to select logs within a 'fuzzy' window for human labeling. These labeled logs are used to denoise and enhance the original training set, thereby boosting the performance of the unsupervised model.

3) Deep learning methods:

- **LogCluster** [51] organizes and clusters historical logs to extract information aimed at identifying anomalous logs.
- **DeepLog** [1] employs LSTM networks to predict the concluding log event in a sequence of normal logs. It detects anomalies by assessing the variance between the predicted log event and the actual log event that occurs.

4) Variants of LogAction:

- ***LogAction***$_{wt}$ (*LogAction* without transfer learning) dismantles the transfer-learning component, omitting the utilization of logs from the source system to train the base model. Instead, it directly employs active learning to select samples to train the anomaly detection model on the target system. *LogAction*$_{wt}$ is utilized to explore the role of the transfer-learning component.
- ***LogAction***$_{wa}$ (*LogAction* without active learning) replaces the active sample selection strategy with a random sample selection of equivalent size. *LogAction*$_{wa}$ is employed to explore the function of the active learning component.
- ***LogAction***$_{wu}$ (*LogAction* without uncertainty-based sampling) is employed to explore the function of the uncertainty-based sampling strategy.
- ***LogAction***$_{we}$ (*LogAction* without free energy-based sampling) is employed to explore the function of the energy-based sampling strategy.

We consistently used the same labeled quantity for the target system whenever required (2% relative to the sample pool). For models with sample selection strategies (such as ACLog and *LogAction*$_{wt}$), we conducted two rounds of selection, with 1% of the samples chosen in each round. For methods lacking sample selection strategies (such as LogTAD and *LogAction*$_{wa}$), we utilized random sample selection to assign labels, and the ultimate outcomes were derived by averaging across 5 random iterations. For RQ2, we examined the influence of manual labeling quantities ranging from 0% to 5% (at intervals of 0.5%) on six combinations of datasets using the *LogAction*. The main adjustment involved varying the quantity of human labeling by augmenting the selection rounds, wherein each round involved the selection of 0.5% of the samples.

C. Measurements

To evaluate the performance of *LogAction* in target test set, we utilize three assessment metrics: *Precision*, *Recall*, and *F1-score*. These metrics rely on four key parameters: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). The *Precision* metric measures the accuracy of anomaly detection within our system. It's calculated as the ratio of TP to the sum of TP and FP, expressed as $\frac{TP}{TP+FP}$. Conversely, the *Recall* metric assesses the system's capability to detect all anomalous log sequences. It's computed as the ratio of TP to the sum of TP and FN, represented by $\frac{TP}{TP+FN}$. The *F1-score* represents the harmonic mean of precision and recall, providing a balanced evaluation incorporating both precision and recall. Its computation follows this formula: $\frac{2\cdot(Precision\cdot Recall)}{Precision+Recall}$.

D. Evaluation Results

1) ***RQ1:*** *How does LogAction performs in terms of effectiveness?:*

We conducted experiments on six combinations of source → target systems involving *LogAction* and Competitors, with the experimental results depicted in Table III. It is important to note that DeepLog, LogCluster, and ACLog were not trained using labels from the source systems, hence their performance is solely dependent on the target systems. As illustrated in the Table III, our approach yielded F1 scores surpassing all competitors across all six datasets, demonstrating the superior performance of *LogAction*.

Compared to the Deep learning methods, *LogAction* requires only 2% of labeled data to achieve an average F1 score surpassing DeepLog and LogCluster by 39.63%. While these unsupervised deep learning methods do not rely on any human labeled data, their performance tends to be inadequate due to the lack of learning from historical anomalous logs. This deficiency arises from the continual emergence of new log types alongside system changes. Because these unsupervised methods detect anomalies based on historical normal logs, encountering new log types often leads to false positives even when such logs are normal. Reflected in the result, they often exhibit lower precision (averaging only 51.74%) due to these false positives. Conversely, *LogAction* learns the occurrence of anomalous logs with minimal human labeling efforts to detect anomalies. When encountering logs of new types, the model does not classify them as anomalies because they do not resemble historical anomalous logs. As a result, *LogAction* exhibits a higher precision (averaging 96.91%).

Compared to the transfer-learning-based methods, *LogAction* outperforms the state-of-the-art transfer learning method MetaLog by 15.42% in F1 score. MetaLog requires 1% of anomaly labels for training. Although this reduces the number of labels needed, it imposes constraints on the type of labels. Since anomalies occur infrequently, obtaining 1% anomaly logs necessitates filtering them from a vast amount of normal logs, which increases labeling costs. In contrast, *LogAction* imposes no restrictions on label types and achieves an F1 score of 93.01% with only 2% of the total labeled data, further reducing labeling costs. *LogAction* outperforms LogTransfer and LogTAD by 17.20% on average F1 score across six combined datasets. As a supervised transfer learning approach, LogTransfer suffers from the gap in data distribution between the source and target systems' logs, resulting in poor performance. When the data distribution in the target system's logs is relatively simple (such as BGL or ThunderBird), LogTransfer demonstrates a certain level of generalization capability (averaging 87.66% F1 score). However, in instances where the distribution of data in the target system logs is intricate, such as in Zookeeper, substantial disparities arise between the data distributions of the source and target systems. Consequently, the models trained on labeled logs from the source system exhibit limited generalization capability, leading to lower F1-scores, averaging at 56.51%. LogTAD, an unsupervised transfer learning method, is also affected by the lack of historical anomaly log learning (similar to DeepLog and LogCluster), impacting its performance. It still requires some manual labeling to continually update the model. In contrast, *LogAction* mitigates the disparity in data distribution by employing active learning to select samples with the highest information content, thus minimizing the need for human

TABLE III: Comparison results with baseline methods for Cross-system Anomaly Detection

Method	ThunderBird → BGL			Zookeeper → BGL			BGL → Zookeeper		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
LogCluster	75.35%	69.63%	82.10%	75.35%	69.63%	82.10%	22.65%	17.87%	30.92%
DeepLog	82.02%	72.66%	94.16%	82.02%	72.66%	94.16%	29.04%	26.48%	32.14%
MetaLog	90.80%	90.24%	91.37%	89.33%	86.22%	92.68%	60.94%	65.21%	57.20%
LogTransfer	91.54%	89.03%	94.19%	94.39%	92.66%	96.19%	57.66%	70.33%	48.85%
LogTAD	89.20%	88.32%	90.10%	90.20%	90.96%	89.45%	50.96%	48.14%	54.12%
ACLog	90.37%	95.34%	85.90%	90.37%	95.34%	85.90%	40.50%	57.38%	31.29%
LogAction	**96.03%**	**96.21%**	**95.84%**	**97.46%**	**97.06%**	**97.87%**	**80.66%**	**99.86%**	**67.65%**

Method	BGL → ThunderBird			Zookeeper → ThunderBird			ThunderBird → Zookeeper		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
LogCluster	71.23%	63.54%	81.03%	71.23%	63.54%	81.03%	22.65%	17.87%	30.92%
DeepLog	34.43%	54.19%	25.23%	34.43%	54.19%	25.23%	29.04%	26.48%	32.14%
MetaLog	82.87%	80.98%	84.85%	83.96%	75.71%	94.22%	57.60%	62.05%	53.75%
LogTransfer	80.60%	75.08%	87.00%	84.11%	80.74%	87.78%	55.35%	61.40%	50.38%
LogTAD	83.87%	93.45%	76.07%	79.91%	69.99%	93.11%	51.89%	58.88%	46.39%
ACLog	46.10%	70.01%	34.36%	46.10%	70.01%	34.36%	40.50%	57.38%	31.29%
LogAction	**92.09%**	**95.16%**	**89.20%**	**95.52%**	**93.96%**	**97.14%**	**96.27%**	**99.19%**	**93.51%**

labeling efforts.

Compared to the active-learning-based method, *LogAction* achieves an average F1 score improvement of 30.60% over ACLog. ACLog, based on unsupervised methods, employs active learning to continuously select samples for online model updating, thereby enhancing the performance of the unsupervised model while reducing the required amount of human labeling. However, ACLog utilizes manually labeled anomalous samples for dataset denoising without fully exploiting these anomalous samples. On datasets with higher levels of noise, such as BGL, ACLog exhibits a substantial improvement compared to unsupervised anomaly detection methods (with an average F1 increase of 11.96%). However, on datasets with lower noise levels, like Thunderbird and Zookeeper, the utilization efficiency of samples chosen through active learning is limited. Consequently, ACLog performs poorly on these datasets (average 43.30% in F1 score). In contrast, *LogAction*, based on supervised methods, can comprehensively learn patterns from anomalous logs, resulting in more efficient anomaly detection. Furthermore, *LogAction* mitigates the cold-start issue of ACLog by leveraging labels from other mature source systems for training the base model.

2) ***RQ2:*** *How does the varying number of labels impact the effectiveness of the LogAction?:*

We investigated the impact of different levels of manual annotations on *LogAction*, and the results are presented in Figure 5. Overall, as the volume of labels increased, the performance of *LogAction* showed a gradual improvement. Notably, the most significant enhancement was observed between the 0% and 1% annotation levels, indicating poor performance of *LogAction* when no labels were available. At this stage, Classifier(source) operated without any fine-tuning, solely relying on the logs from the source system to detect anomalies in the target system. Despite encoding, while the log vectors from the source and target systems exhibited similar distributions, Classifier(source) demonstrated some degree of generalization, achieving an average F1 score of 64.10% on the target system. However, notable distribution gaps persisted between the log vectors of the source and target systems, impeding Classifier(source) from adequately fulfilling the anomaly detection requirements of the target system. Through energy-based sampling and uncertainty-based sampling, *LogAction* selected a subset of the most informative samples for human labeling on the target system. These samples contained non-redundant information and encapsulated the distribution gaps between the log vectors of the source and target systems. Following fine-tuning on this subset of log vectors, Classifier(source) transformed into Classifier(target), more suitable for anomaly detection in the target system. Remarkably, a mere 1% of manual annotations led to an average 22.06% improvement in the F1 score of the model on the target system. Even with a very limited labeling proportion (0.5%), *LogAction* can achieve an average F1-score of 89.95%. This demonstrates *LogAction* ’s effective ability to balance labeling cost and performance. In other words, the amount of labeled logs input to *LogAction* can be adjusted according to practical needs. For example, when logs are difficult to label, *LogAction* can be initiated with 0.5% or even less labeled logs. Conversely, if labeling costs are low, 2% labeled data can be used to achieve the full performance of *LogAction*.

Furthermore, as depicted in the Figure 5, once the level of manual annotations reached a certain threshold, the model's performance plateaued. Additional manual annotations at this

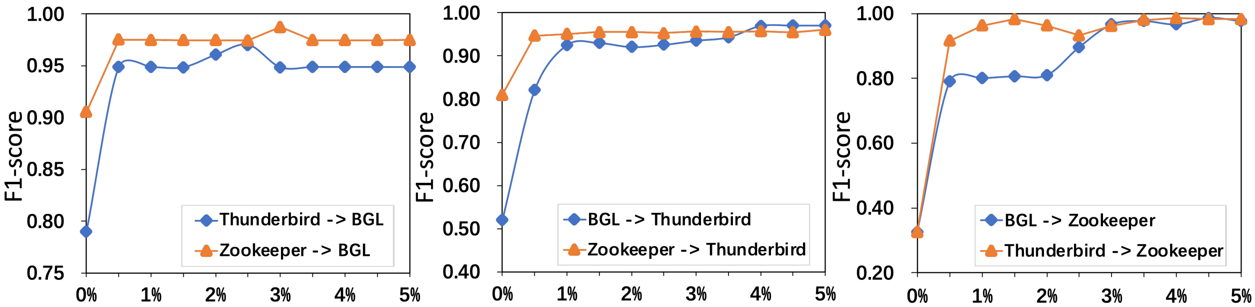


Fig. 5: Human labeling efforts

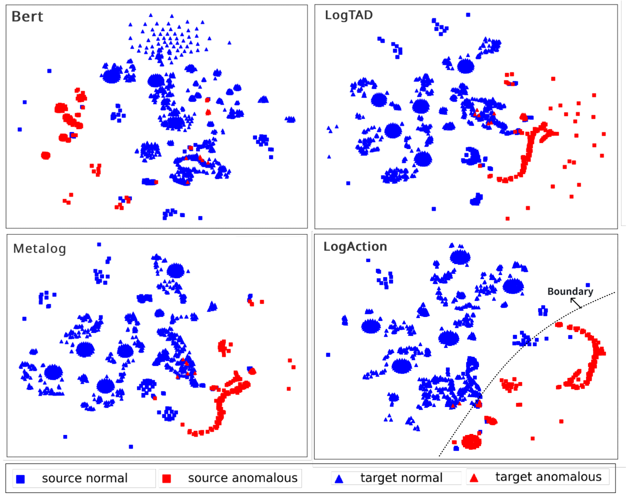


Fig. 6: The difference of data distribution before and after encoding in BLG → ThunderBird.

stage resulted in minimal improvement as the most valuable samples had already been broadly selected, leaving mostly redundant samples. Further model training on these redundant samples yielded little additional benefit since the model had already captured and learned from the information contained within them.

3) ***RQ3:*** *Does each main component contribute to LogAction?:*

In this section, we perform an ablation study to assess the effectiveness of main components in *LogAction*. The findings of the ablation study are shown in Table IV. In comparison to $LogAction_{wt}$ and $LogAction_{wa}$, *LogAction* surpasses them in F1 scores by 21.98% ($LogAction_{wt}$) and 9.00% ($LogAction_{wa}$) respectively. Upon removal of the transfer learning, $LogAction_{wt}$ directly select samples in target system without utilizing the logs of the source system to train the basic model, thus encountering a cold start issue. During the initial rounds of free energy-based sampling and uncertainty-based sampling, lacking fundamental distributional knowledge of the target system prevents $LogAction_{wt}$ from selecting the most valuable samples for fine-tuning. In contrast, after encoding, the log vectors from both the source and target systems exhibit similar distributions. *LogAction* utilizes the log vectors from the source system to train a basic anomaly detection model, thereby possessing a certain degree of generalization ability to the target system. Consequently, *LogAction* accurately models the distribution of the target system, enabling precise active selection and acquisition of the most valuable samples. Compared to $LogAction_{wt}$, the method without the active learning segment, $LogAction_{wa}$, demonstrates higher efficiency. $LogAction_{wa}$ adopts a random selection approach in lieu of the original active learning component, which to some extent can identify valuable samples for fine-tuning. However, its efficacy remains lower than *LogAction* since random selection may include redundant samples. Labeling these redundant samples does not enhance model efficiency as the model has already grasped the knowledge they contain. In comparison, *LogAction* leverages two sampling methods to avoid generating redundant samples and further alleviate data distribution gaps, thus exhibiting superior performance. We further investigated the contributions of two sampling strategies in active learning. As shown in Table IV, after ablating each sampling method, the model's performance decreased by an average of 3.05% and 3.35%, respectively. This further highlights the importance of both sampling strategies. On one hand, *LogAction* employs free energy-based sampling to capture samples located at the distribution boundaries, thereby further reducing the distribution gap between the source and target systems and facilitating model transfer. On the other hand, *LogAction* utilizes uncertainty-based sampling to identify samples near the decision boundary between normal and anomalous classes that the model finds difficult to detect, thereby enhancing anomaly detection performance.

In addition, we further investigate the role of encoding in the method. For the Encoding component, we utilized the t-SNE dimensionality reduction method to visualize the data distributions of log vectors after encoding. We investigated the t-SNE dimensionality reduction results of encoded representations from several different encoders, including BERT, LogTAD, MetaLog, and *LogAction*. Specifically, the visualization of the BGL → Thunderbird dataset is depicted in Figure 6, where

TABLE IV: Ablation of Proposed two Key Components of *LogAction*

Method	**ThunderBird → BGL**			**Zookeeper → BGL**			**BGL → Zookeeper**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
LogAction$_{wt}$	90.53%	87.55%	93.73%	90.53%	87.55%	93.73%	32.60%	50.89%	23.98%
LogAction$_{wa}$	89.33%	88.82%	89.86%	96.39%	96.40%	96.38%	62.60%	98.87%	45.80%
LogAction$_{wu}$	92.07%	90.59%	93.60%	95.32%	95.29%	95.35%	77.18%	99.02%	63.23%
LogAction$_{we}$	91.75%	93.54%	90.03%	96.91%	98.03%	95.81%	73.66%	99.59%	58.44%
LogAction	**96.03%**	**96.21%**	**95.84%**	**97.46%**	**97.06%**	**97.87%**	**80.66%**	**99.86%**	**67.65%**

Method	**BGL → ThunderBird**			**Zookeeper → ThunderBird**			**ThunderBird → Zookeeper**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
LogAction$_{wt}$	89.32%	**99.52%**	81.01%	89.32%	**99.52%**	81.01%	32.60%	50.89%	23.98%
LogAction$_{wa}$	79.76%	89.82%	71.73%	90.30%	85.06%	96.23%	85.66%	80.74%	91.22%
LogAction$_{wu}$	85.91%	93.11%	79.74%	92.31%	91.28%	93.36%	96.95%	96.90%	97.01%
LogAction$_{we}$	86.07%	92.47%	80.49%	92.98%	89.69%	96.51%	96.55%	96.63%	96.48%
LogAction	**92.09%**	95.16%	**89.20%**	**95.52%**	93.96%	**97.14%**	**96.27%**	**99.19%**	**93.51%**

blue and red represent normal log vectors and anomalous log vectors, respectively. Triangles and circles denote log vectors from the source and target systems. Among these, *LogAction* 's encoder performs most prominently, as the encoded log vectors from different systems are the most tightly clustered. After encoding the data distributions between the source and target systems become analogous, leading to a reduction in the gaps of their distributions. *LogAction* reduces the distribution gap between the source system and target system logs through contrastive learning, thereby better facilitating model transfer. In comparison, the encoding performance of MetaLog and LogTAD is suboptimal, with many normal and abnormal log vectors intermixed. Moreover, the comparison with the BERT encoder further highlights that the semantic distributions of logs from different systems are inconsistent, making it insufficient to achieve distribution alignment solely by extracting semantics.

E. Threats to Validity

Regarding the hyperparameters, except for the sensitivity experiments, all other hyperparameters were held constant throughout our experiments. A comprehensive sensitivity analysis of these hyperparameters is therefore lacking. Nonetheless, these parameters are not the primary focus of our study and can be set according to the optimal configurations reported in prior classical works [1], [17]. For instance, concerning the LSTM hidden size in the encoder component, the encoder's generalization capability is predominantly driven by the contrastive learning framework, with the LSTM serving as a standard feature extractor.

Regarding the fixed-size log windows, in the BGL, ThunderBird, and Zookeeper datasets, we employ fixed-size log windows to segment logs into log sequences, which may affect model performance. Since other baselines also use fixed-size log windows for these three datasets, to ensure fairness, we adopt the same approach and maintain window sizes consistent with those used in prior studies.

Regarding the labeled logs used in *LogAction*, for fairness considerations, we used 2% labeled logs in the comparative experiments between the baselines and *LogAction*. In practice, obtaining even 2% labeled logs may incur high costs. However, 2% labeled logs is not strictly necessary. Striving to balance labeling cost and effectiveness, *LogAction* allows the proportion of input labels to be adjusted according to practical needs. As demonstrated in Section IV-D2, it maintains strong performance even with as little as 0.5% of labeled data.

V. Conclusion

Accurately identifying log labels from a large volume of logs is a highly challenging task, posing difficulties in training anomaly detection models. Existing methods that tackle the scarcity of labeled data primarily involve transfer learning and active learning. Nevertheless, both approaches have their limitations, yet they can be complementary and mutually resolved. In this paper, we propose *LogAction*, a novel anomaly detection approach based on active domain adaptation to tackle the issue of limited labeled data. *LogAction* combines transfer learning and active learning techniques. On one hand, it employs active learning to selectively label key logs, bridging the gap between distinct systems in transfer learning. On the other hand, *LogAction* employs transfer learning, training the base model using labels from the source system, to mitigate the cold-start issue in active learning. Compared to several existing state-of-the-art transfer learning and active learning methods, *LogAction* outperforms them by 26.28% while requiring only 2% of the amount of human annotation. In the future, we will focus on researching new methods to achieve more precise log-based anomaly detection models with minimal human labeling.

Acknowledgment

This research is supported by PKU-ByteDance Research Project.

References

[1] 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*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1285–1298.

[2] 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.

[3] 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.

[4] 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.

[5] S. He, T. Deng, B. Chen, R. S. Sherratt, and J. Wang, "Unsupervised log anomaly detection method based on multi-feature." *Computers, Materials & Continua*, vol. 76, no. 1, 2023.

[6] 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)*, 2024, pp. 132–143.

[7] ——, " Weakly-Supervised Log-Based Anomaly Detection with Inexact Labels via Multi-Instance Learning ," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 2918–2930. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICSE55347.2025.00189

[8] P. Xiao, T. Jia, C. Duan, M. He, W. Hong, X. Yang, Y. Wu, Y. Li, and G. Huang, *CLSLog: Collaborating Large and Small Models for Log-based Anomaly Detection*. New York, NY, USA: Association for Computing Machinery, 2025, p. 686–690. [Online]. Available: https://doi.org/10.1145/3696630.3728524

[9] 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.

[10] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, and D. Zhang, "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*, ser. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 807–817.

[11] 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)*, 2019, pp. 167–175.

[12] 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*, 2010, pp. 187–194.

[13] X. Xie, S. Jian, C. Huang, F. Yu, and Y. Deng, "Logrep: Log-based anomaly detection by representing both semantic and numeric information in raw messages," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 194–206.

[14] T. Jia, Y. Li, Y. Yang, G. Huang, and Z. Wu, "Augmenting log-based anomaly detection models to reduce false anomalies with human feedback," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3081–3089.

[15] R. Chen, S. Zhang, D. Li, Y. Zhang, F. Guo, W. Meng, D. Pei, Y. Zhang, X. Chen, and Y. Liu, "Logtransfer: Cross-system log anomaly detection for software systems with transfer learning," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 37–47.

[16] X. Han and S. Yuan, "Unsupervised cross-system log anomaly detection via domain adaptation," in *Proceedings of the 30th ACM international conference on information & knowledge management*, 2021, pp. 3068–3072.

[17] 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.

[18] 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.

[19] 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.

[20] C. Duan, T. Jia, Y. Yang, G. Liu, J. Liu, H. Zhang, Q. Zhou, Y. Li, and G. Huang, "Eagerlog: Active learning enhanced retrieval augmented generation for log-based anomaly detection," in *ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025, pp. 1–5.

[21] 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)*, 2024, pp. 144–155.

[22] 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.

[23] B. Xie, L. Yuan, S. Li, C. H. Liu, X. Cheng, and G. Wang, "Active learning for domain adaptation: An energy-based approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8708–8716.

[24] P. Rai, A. Saha, H. Daumé III, and S. Venkatasubramanian, "Domain adaptation meets active learning," in *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, 2010, pp. 27–32.

[25] K. Han, Y. Kim, D. Han, and S. Hong, "Loss-based sequential learning for active domain adaptation," *arXiv preprint arXiv:2204.11665*, 2022.

[26] B. Fu, Z. Cao, J. Wang, and M. Long, "Transferable query selection for active domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7272–7281.

[27] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[28] L. Zhang, Y. Zhai, T. Jia, X. Huang, C. Duan, and Y. Li, "Agentfm: Role-aware failure management for distributed databases with llm-driven multi-agents," in *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, ser. FSE Companion '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 525–529. [Online]. Available: https://doi.org/10.1145/3696630.3728492

[29] 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," in *Companion Proceedings of the ACM on Web Conference 2025*, ser. WWW '25. New York, NY, USA: Association for Computing Machinery, 2025, p. 548–556. [Online]. Available: https://doi.org/10.1145/3701716.3715221

[30] L. Zhang, Y. Zhai, T. Jia, C. Duan, S. Yu, J. Gao, B. Ding, Z. Wu, and Y. Li, "Thinkfl: Self-refining failure localization for microservice systems via reinforcement fine-tuning," 2025. [Online]. Available: https://arxiv.org/abs/2504.18776

[31] C. Duan, Y. Yang, T. Jia, G. Liu, J. Liu, H. Zhang, Q. Zhou, Y. Li, and G. Huang, " Famos: Fault Diagnosis for Microservice Systems Through Effective Multi-Modal Data Fusion ," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2025, pp. 2613–2624. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ICSE55347.2025.00073

[32] C. Duan, F. Yang, P. Zhao, L. Zheng, Y. Dagli, Y. Liu, Q. Lin, and D. Zhang, "Soil: Score conditioned diffusion model for imbalanced cloud failure prediction," in *Companion Proceedings of the ACM Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 65–72. [Online]. Available: https://doi.org/10.1145/3589335.3648303

[33] L. Zhang, T. Jia, K. Wang, W. Hong, C. Duan, M. He, and Y. Li, “Adaptive root cause localization for microservice systems with multi-agent recursion-of-thought,” 2025. [Online]. Available: https://arxiv.org/abs/2508.20370

[34] 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.

[35] X. Yu, P. Joshi, J. Xu, G. Jin, H. Zhang, and G. Jiang, “Cloudseer: Workflow monitoring of cloud infrastructures via interleaved logs,” *SIGARCH Comput. Archit. News*, vol. 44, no. 2, p. 489–502, mar 2016.

[36] 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.

[37] 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.

[38] 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.

[39] J. Xu, P. Chen, L. Yang, F. Meng, and P. Wang, “Logdc: Problem diagnosis for declartively-deployed cloud applications with log,” in *2017 IEEE 14th International Conference on e-Business Engineering (ICEBE)*, 2017, pp. 282–287.

[40] A. Babenko, L. Mariani, and F. Pastore, “Ava: Automated interpretation of dynamically detected anomalies,” in *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, ser. ISSTA ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 237–248.

[41] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, “Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks,” in *Proceedings of the 29th Annual Computer Security Applications Conference*, ser. ACSAC ’13. New York, NY, USA: Association for Computing Machinery, 2013, p. 199–208.

[42] B. C. Tak, S. Tao, L. Yang, C. Zhu, and Y. Ruan, “Logan: Problem diagnosis in the cloud using log-based reference models,” in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, 2016, pp. 62–67.

[43] A. Graves, “Long short-term memory,” *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[44] J.-C. Su, Y.-H. Tsai, K. Sohn, B. Liu, S. Maji, and M. Chandraker, “Active adversarial domain adaptation,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 739–748.

[45] 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)*, 2017, pp. 33–40.

[46] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[48] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, “A tutorial on energy-based learning,” *Predicting structured data*, vol. 1, no. 0, 2006.

[49] 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.

[50] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, “{ZooKeeper}: Wait-free coordination for internet-scale systems,” in *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[51] R. Vaarandi and M. Pihelgas, “Logcluster-a data clustering and pattern mining algorithm for event logs,” in *2015 11th International conference on network and service management (CNSM)*. IEEE, 2015, pp. 1–7.