World Wide Web (2023) 26:3137–3153
https://doi.org/10.1007/s11280-023-01174-y

SSDLog: a semi-supervised dual branch model for log anomaly detection

Siyang Lu[1] · Ningning Han[1] · Mingquan Wang[2] · Xiang Wei[2] · Zaichao Lin[1] · Dongdong Wang[3]

Received: 22 February 2023 / Revised: 31 March 2023 / Accepted: 8 April 2023 /
Published online: 13 June 2023

Abstract
With versatility and complexity of computer systems, warning and errors are inevitable. To effectively monitor system's status, system logs are critical. To detect anomalies in system logs, deep learning is a promising way to go. However, abnormal system logs in the real world are often difficult to collect, and effectively and accurately categorize the logs is an even time-consuming project. Thus, the data incompleteness is not conducive to the deep learning for this practical application. In this paper, we put forward a novel semi-supervised dual branch model that alleviate the need for large scale labeled logs for training a deep system log anomaly detector. Specifically, our model consists of two homogeneous networks that share the same parameters, one is called weak augmented teacher model and the other is termed as strong augmented student model. In the teacher model, the log features are augmented with small Gaussian noise, while in the student model, the strong augmentation is injected to force the model to learn a more robust feature representation with the guidance of teacher model provided soft labels. Furthermore, to further utilize unlabeled samples effectively, we propose a flexible label screening strategy that takes into account the confidence and stability of pseudo-labels. Experimental results show favorable effect of our model on prevalent HDFS and Hadoop Application datasets. Precisely, with only 30% training data labeled, our model can achieve the comparable results as the fully supervised version.

Keywords Log anomaly detection · Semi-supervised learning · Distributed system · Dual branch

1 Introduction

With the exponential growth of computing tasks and data, distributed parallel computing systems (DPCS) are increasingly widely adopted to make full use of hardware resources to achieve rapid and effective task deployment. While DPCS are effective in many ways, they

✉ Siyang Lu
sylu@bjtu.edu.cn

Extended author information available on the last page of the article

are not easy to maintain and manage, which can cause some serious system problems. To effectively monitor the system's health, system logs are usually collected for diagnosing.

Generally, the log-based anomaly detection method is realized by mining a large amount of system log data and conducting effective classification, which can be simply treated as a binary classification task. At present, for this task, there are mainly machine learning algorithms based on shallow method [1–5] and methods based on deep model [6–10]. For the traditional machine learning methods, they do not need a large amount of well annotated data to achieve satisfactory classification effect. For deep models, although their accuracy is much higher than that of traditional machine learning, the dependence on massive data often makes the model inadequate in real applications. For the problem we faced in log anomaly detection, due to the diversity of anomalies and the large number of logs, effective annotation of data is not energy-consuming, and in some real distributed systems, abnormal data is often difficult to collect, resulting in more scarce effective annotation.

In this paper, to tackle the above mentioned issue in log anomaly detection, we propose a innovative semi-supervised dual branch Log anomaly detection model dubbed SSDLog that alleviate the need for large scale labeled logs for training a deep system log anomaly detector. Specifically, in the training process of our model, we do not need a large amount of labeled data like the conventional deep models, but can achieve the comparative training effect as them with the support of only 30% label data and the remaining 70% unlabeled one. More explicitly our model consists of two homogeneous networks that share the same parameters, one is called weak augmented teacher model and the other is termed as strong augmented student model. In the teacher model, the log features are augmented with small Gaussian noise, while in the student model, the strong augmentation is injected to force the model to learn a more robust feature representation with the guidance of teacher model, which can provide a more stable soft pseudo-labels for supervised training. In the meantime, to further utilize unlabeled samples effectively, we propose a label screening strategy that takes into account the confidence and stability of pseudo-labels, which can further obtain reliable and learnable training samples.

To summarize, the following three contributions are made in this paper:

1. A novel semi-supervised dual branch model for efficient Log anomaly detection is proposed with only a few labels available in the training set. To the best of our knowledge, this is one of the latest attempts for the Log anomaly detection task in real scenario.
2. A novel teacher-student semi-supervised model integrated with flexible label screening strategy is established to efficiently detect system Log anomalies.
3. Our model has achieved superior performance on prevalent system Log anomaly detection datasets. In particular, the performance of our SSDLog is comparable to that of the full-labeled methods [7, 11] under the condition of only 30% labeled data available.

The remainder of the paper is organized as follows. In Section 2, we survey recent related work. Section 3 details our SSDLog approach. Sections 4 and 5 present the experimental results on multiple datasets. Finally, we conclude the paper in Section 6 with a summary and an outlook on future work.

2 Related work

This section will introduce related work from two aspects: log anomaly detection and semi-supervised deep learning.

2.1 Log anomaly detection approaches

Before deep learning take over computer vision and NLP, statistic approaches and non-deep machine learning approaches are mainly leveraged in log analysis field. On the one hand, classic statistical methods calculate specific features manually extracted from log data. There are several traditional statistic approaches, include PCA-based approach [12].In addition, Safyallah et al. [13] analyze frequent and common log sequence execution path to detect anomalies. Fu et al. [14] use rule-based method to identify log templates and detect anomalies in distributed system logs.

To prevent certain features extracted by statistic approaches from affecting the effect of log anomaly detection, many studies emerged who using non-deep machine learning methods to detect anomalies. The related approaches include, SVM-based approaches [1, 2], Bayesian Learning-based model [3], Decision Tress-based model [4], HMM(Hidden Markov Model)-based approach [5]. For example, Fulp et al. [1] leverage sliding window to analyze system logs and support vector machine (SVM) to predict anomalies. Liang et al. [2] use support vector machine (SVM), RIPPER (a rule-based classifier) and custom nearest neighbor method to establish three classifiers for anomaly prediction. Lou et al. [5] apply Bayesian Learning method to extract structural diagrams from system logs. Hen et al. [4] identified system failures by using decision tree to model labeled log information with higher interpretability compared with other classification methods. Morever, Yadwadkar et al. [3] use Hidden Markov Model-based (HMM) methods to detect anomalies.

Recently years, Deep Learning-based approaches are popularly used for abnormal log detection with better performance. AutoEncoder is an unsupervised neural network which effectively compresses and encodes data. Farzad et al. [15] proposed an unsupervised model for anomaly detection of log messages by using isolated forest and two AutoEncoder networks, the isolated forest is used to detect normal logs with thresholds, and the AutoEncoder networks are used for training and feature extraction. Afterwards, due to the fact that LSTM-based approaches [6] could extract sequence features and detect anomaly log in real time. Du et al. [7] design a LSTM-based model to detect abnormal for log workflows and log values. Brown et al. [8]extend an attention mechanism to LSTM for improving detection performance. Moreover, another LSTM-based model combines with generative adversarial networks (GANs) is proposed by Xia et al. [9]. For offline approaches, a CNN-based model [10] is proposed by us to achieve state-of-art in abnormal detection with Hadoop dataset. Currently, we propose a black-box based attacking approach to attack some of above detection models. The results show that the robustness of those detection models is weak. Hence, we propose a Self-Knowledge Distillation (SKD) approach to improve the robustness and accuracy for typical deep learning-based log detection models [16].

Nowadays, new methods derived from machine learning have also been used for log anomaly detection. Rui et al. [17] put forward a framework called LogTransfer, which uses transfer learning to transfer abnormal log information in a cross-system way, thus avoiding similar log systems to retrain the detection model, and greatly reducing the manpower, material resources and time expenses on the premise of ensuring the detection effect. Duan et al. [18] designed a method called QLLog, which integrates Q-Learning algorithm and feedback mechanism in reinforcement learning to improve the understanding of anomaly log patterns and detect anomalies quickly and effectively.

Nevertheless, in actual scenarios, the logs of different systems are different. Therefore, manual labeling of massive log information takes time and effort. The above related algorithms do not take into account the semi-supervised learning conditions, which greatly limits the applicability of current Log anomaly detection algorithms. This paper innovatively

proposes a semi-supervised log anomaly detection algorithm to further promote the application of this technology.

2.2 Semi-supervised learning

At present, the latest semi-supervised deep learning (SSL) methods focus mainly on the following three aspects:1. Consistency regularization [19–23]; 2. Pseudo-labeling [24–26]; 3. Data augmentation [27–30].

For consistency regularization strategy, the basic idea is to make the output of the same disturbed sample through the network as consistent as possible, so as to enhance the representation smoothness of the model around the sample. In [19], the consistency penalty term is first introduced in to semi-supervised learning optimization. The method constrained the consistency of the output of two versions of the same input data. In addition, to enhance the robustness of the model to label noise and shorten the training time, temoral ensembling is proposed to get one of the branch's output in a non-parametric way by exponential moving average (EMA). Considering the delay in updating the sample output, Meam Teacher [31] adopted the method of EMA on the model to further enhance the performance on the large dataset. In order to carry out data disturbance more effectively and find the best consistency constraint direction, [32] proposed to use virtual adversarial training strategy to carry out data disturbance. Recently, [33] combined MixUp [34] and consistency strategies in SSL to make consistency act between different classes to obtain a more reasonable classification hyperplane. Similarly, in [21], Berthelot et al. proposed a MixUp-based consistency method combined with auto-augmentation for SSL. Most recently, Jiang et al. [35] proposed a worst case consistency regularization technique that minimizes the largest inconsistency between an original unlabeled sample and its multiple augmented variants.

For Pseudo-labeling strategy, in brief, it uses the prediction of unlabeled samples derived from the same or a dedicated model's prediction as the training label to expand the dataset, and such approaches commonly adopt a high threshold mask to alleviate the confirmation bias. Arazo et al. [24] proposed a soft pseudo-labeling strategy combined with MixUp augmentation for SSL. Then Wei et al. [25] proposed an incremental self-labeling strategy based on Generative adversarial network for efficient label propagation. Recently, considering that it is not reliable to directly assign a fixed threshold for filtering pseudo-labels, Wang et al. proposed [26] FreeMatch that to define and adjust the confidence threshold according to the model's learning status. Similarly, in [36] Huang et al. proposed a percentile-based threshold adjusting scheme, named PercentMatch, that is to dynamically alter the score thresholds of positive and negative pseudo-labels for each class during the training.

Data augmentation methods typically improve the generalization ability of the model by generating various variant data through image-based transformation and Mixed Data Augmentation (MDA) [29, 33]. For the image-based transformation augmentation, the most representative approach is FixMatch [28], which operated by adopting weak and strong auto-augmentation for SSL. For MDA-based augmentation, in [29] Wei et al. proposed a novel FMixCut algorithm for effective data augmentation. In [27], to deal with the problem of class-imbalance in SSL (CISSL), Kong et al. proposed a MixUp-based data augmentation and label reassignment strategy for CISSL.

The mechanisms of the above three strategies are different; however, in some cases, they can be fused effectively to further improve the accuracy of the model. Whereas, none of the aforementioned literature has studied SSL in Log anomaly detection, in which the accuracy of the detector is significantly threatened by the small amount of labeled data. In this paper,

we propose a novel framework dubbed SSDLog for semi-supervised log anomaly detection. To the best of our knowledge, this is one of the latest attempts for the Log anomaly detection task in real semi-supervised scenario.

3 Methodology

In this section, we first present the problem setup for semi-supervised Log anomaly detection problem. Based on this, we introduce our SSDLog, a novel teacher-student semi-supervised model integrated with flexible label screening strategy for semi-supervised Log anomaly detection.

3.1 Problem setup

For Log-based abnormal detection task, as the raw data is usually a sequence of logs with semantic information. In this study, the target log data sets are HDFS log and Hadoop application log. Following most works, we preprocess the raw data with log template approach, and turn it into log key sequences. The log sequence is denoted by $x_i \in (R)^{L\times 1}$, where L indicates the fixed length of each log code. As shown in Figure 1.

For a normal Log, we set its label y_i to 1, otherwise to 0.

Formally, suppose that for a binary abnormal detection problem, we have a labeled dataset $\mathcal{X} = \{(x_m, y_m) : m \in (1, ...\mathcal{M})\}$, and an unlabeled dataset $\mathcal{U} = \{(u_n) : n \in (1, ..., \mathcal{N})\}$. We express the labeled ratio as $\beta = \mathcal{M}/(\mathcal{M}+\mathcal{N})$ and $\mathcal{M} \ll \mathcal{N}$. As we usually assume that the labeled data are randomly selected from the whole dataset, unlabeled data and labeled data are all obey the same distribution. Consequently, our goal is to train a binary classifier $f(\cdot)$ that performs effectively to detect anomalies from the normal Logs.

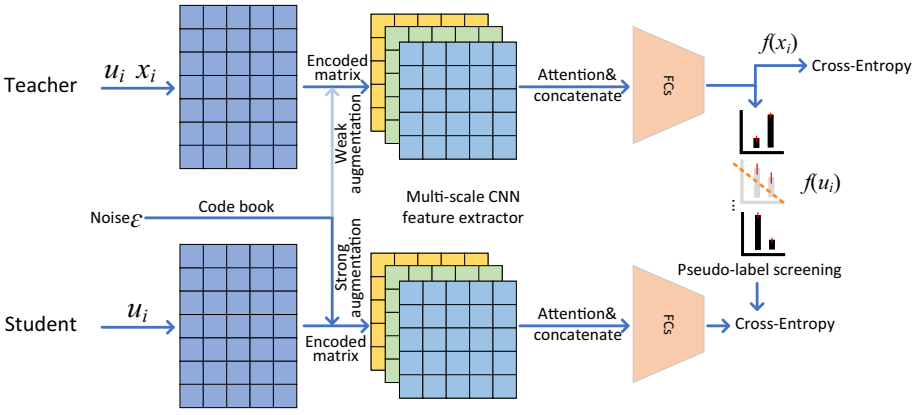

Figure 1 The schematic of SSDLog. Two branches are carried out including a teacher model and a student model. For the teacher model, both labeled and unlabeled data are used as input, and weak augmentation is injected into the encoded vector; at the end of the classification head of teacher model, the pseudo-label screening strategy is carried out to filter out more stable and confidential samples. For the student model, only unlabeled data is used as input, and strong augmentation is injected into the encoded vector; for the classification head of student model, the Cross-Entropy is conducted in the pseudo-labeled screened training sample

3.2 SSDLog anomaly detector

The framework of the proposed SSDLog is presented in Figure 1. As we can see, our whole framework consists of two networks that share the same network structure and parameters. The model at the top is called teacher model, which takes both labeled set $\mathcal{X}$ and unlabeled set $\mathcal{U}$ as input. Inside teacher model, the weak augmentation is injected to get reliable outputs for both sets' data. The model at the bottom is called student model, which only takes unlabeled set $\mathcal{U}$ as input for training. Noting that, for the label provided from the teacher model to train the student, only samples with high confidence and stable output can be selected. In addition, in the training of student model, in order to make its generalization ability stronger, stronger noise disturbance is injected into its encoded vector than the teacher's model. In the following, we will first detail our dual branch model, then explain our label screening process.

A. Dual branch architecture

Since our two-branch model belongs to the siamese neural network, we'll use the teacher model as an example here, and we'll declare the differences where necessary.

As shown in Figure 1, for the teacher model $f^T(\cdot)$, we will feed both u_i and x_i as input. Then we declare a learnable code book matrix $C(\cdot) \in \mathbb{R}^{N\times K}$ for encoding discrete Log values, where N indicates the number of discrete codes, and K is the encoding length for each code scalar. As each Log is an encoding sequence with fixed length L, the output $e(x_i)$ after encoding for each sample is a matrix with size $L \times K$. Formally, we express the obtained encoded matrix for sample x_i as:

$$e(x_i) = x_i \odot C \tag{1}$$

where $\odot$ represents the encoding operation of x_i by code book C.

After the encoded matrix is obtained, we then add noise into each matrix. Since we did not limit the output range of the encoding, it is difficult to obtain a relatively reasonable noise by simply random a noise matrix. Also, Dropout operation is also not suitable for our log anomaly detection as some code is fatal to identify abnormal and there may not have strong dependence from other encoded values. Then, here, we make innovative use of a MixUp-like feature augmentation approach that mix feature pairs with fusion factor ε. Formally, we express our augmentation process as:

$$\widetilde{e(x_i)} = (1-\varepsilon)e(x_i) + \varepsilon\widehat{e(x_i)} \tag{2}$$

where, ε is sampled from beta(α, α) distribution and can be expressed as:

$$\varepsilon \sim \max(beta(\alpha,\alpha), 1 - beta(\alpha,\alpha)) \tag{3}$$

where α is set to 0.95 for teacher model, so as to keep more information from the original encoded feature $e(x_i)$. For student mode, α in (3) is set to 0.9 to give a relative stronger augmentation. For simplicity, we express all the data derived from teacher model with superscript T, and S for student's, otherwise there is no need to specify them.

It's worth noting that in (2), $\widehat{e(x_i)}$ is obtained by a random shuffle of $e(x_i)$.

After the augmented feature matrix is obtained, it then passed into three parallel convolutional layer with different kernel size. Following [10], we implement three kernel with size $3 \times K, 4 \times K$ and $5 \times K$ to catch the long and short dependencies among encoded features. In order to further enable the model to automatically catch effective features in different ranges,

we also use the attention mechanism for feature weighting. Since this part is not the key part of this paper, we will not expand the description, and relevant operations can be viewed in our previous work [16].

Finally, after a simple fully-connected classifier head, we get the final output of the model. Then we implement Cross-Entropy loss to train the model with corresponding labeled samples x_i along with their labels, as shown in (4). It should be noted that the ratio between labeled samples and unlabeled samples is fixed to 1:1 in teacher model's training to take into account the accuracy of the model and the ability to explore unknown samples.

$$\mathcal{L}_{ce}(x_i, y_i) = CE(f_t^T(x_i), y_i) \tag{4}$$

B. Flexible label screening

This part is both related to teacher model and student model. First, using the teacher model, we obtain the output of the unlabeled sample u_i at epoch t, denoted as $f_t(u_i)$. We then weighted the historical output using an exponential moving average (EMA) to get $M_t(u_i)$ at epoch t. Meanwhile, the output variances of the latest 30 epochs are calculated for each sample(u_i), and expressed as $S_t(u_i)$, respectively.

Next, we effectively filter the unlabeled data by declaring an evaluation expression that takes into account the confidence and stability of the teacher's output of the unlabeled sample:

$$F_t^T(u_i) = M_t^T(u_i) - \mu S_t^T(u_i) \tag{5}$$

where μ is the hyper-parameter that controls the proportion of confidence and stability importance, in our paper, we set it to 0.1 throughout our experiments.

$M_t^T(u_i)$ in (5) is calculated by:

$$\begin{aligned} m_t^T(u_i) &= \lambda m_{(t-1)}^T(u_i) + (1-\lambda) f_t^T(u_i) \\ M_t^T(u_i) &= m_t^T(u_i)/(1-\lambda^t) \end{aligned} \tag{6}$$

where λ is the momentum weight used to control the updating rate, in our paper, we set it to 0.4 by default. $f_t^T(u_i)$ is the output of the teacher model for the unlabeled sample u_i at epoch t.

For the calculation of $S_t^T(u_i)$, all the initial value is set to 0, and each $S_t^T(u_i)$ is updated by:

$$S_t^T(u_i) = std(\{f_{t-29}^T(u_i), f_{t-28}^T(u_i), ..., f_t^T(u_i)\}) \tag{7}$$

When we get $F^T(u_i)$ for each unlabeled data u_i, we then conduct pseudo-label screening by choosing the value of $F^T(u_i)$ greater than a pre-defined threshold τ as pseudo-labeled sample for Cross-Entropy optimization. For a tradeoff between safety and exploration, we set τ as a dynamic changing hyper-parameter along with training epoch, and expressed as:

$$\tau_t = 0.95 \times \frac{t}{epochs} \tag{8}$$

where *epochs* denotes the total epochs to be trained. From (8), we can conclude that with the training goes by, the threshold will grow slowly. This manner will guarantee that the model will receive more data for the initial training so as to avoid the over-fitting and confirmation bias problem. As the training processes, the model will concentrate on more safe and stable samples for training and finally reach a desired anomaly detector.

From (5) to (7) we can see that our screening strategy can take both confidence and stability of the output into consideration, so as to get more valuable unlabeled samples for training. The Cross-Entropy loss operated in the student side is formally expressed a in (9)

$$\mathcal{L}_{ce}(u_i) = \mathbb{I}(F_t^T(u_i)) \times CE(f_t^S(u_i), Onehot(M_t^T(u_i))) \tag{9}$$

where $\mathbb{I}(\cdot)$ is a 0-1 indicator that defined as:

$$\mathbb{I}(F_t^T(u_i)) = \begin{cases} 1, if \quad F_t^T(u_i) > \tau_t \\ 0, otherwise \end{cases} \tag{10}$$

In (9), $Onehot(\cdot)$ denotes the operation of converting the probability vector to an Onehot 0-1 vector.

We summarize SSDLog anomaly detection algorithm in Algorithm 1. Note that, the teacher model is responsible for training the student model with unlabeled data, which transfers more latent information to detection model. The student model shares the same weight as the teacher model but with different augmentation injected into the training process.

Algorithm 1 SSDLog anomaly detection.

Input: prepossessed log sequences set $\mathcal{X}$ and $\mathcal{U}$. $\alpha^T = 0.95$; $\alpha^S = 0.9$. $\mu = 0.1$. λ=0.4.
1: **for** t in R iterations (or Epochs) **do**:
2: sample a batch of labeled data $\{(x_1, y_1),...,(x_B, y_B)\}$
3: sample a batch of unlabeled data $\{u_1,...,u_B\}$
4: $f_t^T(x_i)$, $f_t^T(u_i) \leftarrow$ Teacher model (x_i, u_i)
5: $f_t^S(u_i) \leftarrow$ Student model (u_i)
6: $\mathcal{L}_{ce}(x_i, y_i) = CE(f_t^T(x_i), y_i)$ ▷ (4)
7: $M_t^T(u_i) \leftarrow$ (6)
8: $S_t^T(u_i) \leftarrow$ (7)
9: $F_t^T(u_i) \leftarrow$ (5)
10: $\mathcal{L}_{ce}(u_i) = \mathbb{I}(F_t^T(u_i))CE(f_t^S(u_i), Onehot(M_t^T(u_i)))$ ▷ (9)
11: Update $f(\cdot)$ through $\mathcal{L}_{ce}(x_i, y_i)$ and $\mathcal{L}_{ce}(u_i)$
12: **end for**

3.3 Efficiency analysis

Through the introduction of our framework, it can be found that we made two copies of the same model for the effective SSL through combining consistency and pseudo-labeling strategy, and also adaptively added the feature augmentation strategy for Log in the training process. Therefore, compared with the traditional single branch network, the running time of our algorithm is bound to increase. To put it simply, we are still training only one network model, and the two-branch structure can be seen as passing through the network twice for the unlabeled sample. Therefore, the theoretical increase in training time is simply to run the unlabeled sample once more during the forward propagation process.

It is worth noting that in the test phase of our model, feature augmentation is removed, and the inference time of the network is consistent with that of the traditional single-branch network, that is, there is no performance degradation in our model.

4 Evaluation

In this section, we first detail the experiment setup and datasets. Then, we compare the proposed SSDLog method with other methods on the semi-supervised version of HDFS and Hadoop Application datasets. Finally, we apply our model to a real-world industry dataset and an unstable dataset to validate the robustness and efficiency.

4.1 Experimental setup and dataset

We implement the model through Pytorch [37]. We evaluate our model and the other relevant approaches on HDFS and Hadoop, two prevalent benchmarks on distributed system log. We set β = 10%, 20%, 30% and 50%, respectively to construct the semi-supervised version of HDFS and Hadoop Application. We also show the fully-supervised results under two datasets. In addition, we establish a novel dataset, i.e, a real world industry dataset, to validate the effectiveness of our proposed method in real SSL condition with $\beta = 50\%$. The data distribution of these datasets is listed in Table 1. Their detailed information is introduced as follows:

HDFS dataset is first produced by Xu et al. [12], and this dataset is from more than 200 Amazon EC2 nodes for Hadoop distributed file system in runtime, which collects 11,175, and 629 logs, plus 29 unique log keys. First, the raw data size is over 1.5 GB, and then the raw data is divided into log sessions based on $block\_id$. Finally, there are 558,223 normal sessions and 16,838 anomalous sessions in the dataset.

Hadoop Application dataset [38] contains two Hadoop applications from Hadoop cluster, including Word Count and Page Rank. The whole cluster has five nodes with 16GB RAM for each. The abnormal logs consist of the errors caused by machine malfunction, network disconnection, and full disk.

Industry Dataset We collect real-world industry dataset from the Windows Event Tracing for Windows (ETW) framework, which records 135 different log sessions. Different log sessions represent differnt system behaviors. The industry dataset consists of 4,878 abnormal log events and 130,411 normal log events.

Unstable Dataset We define noisy dataset as unstable dataset since it is hard predicted system performance due to unstable log pattern. We simulate unstable log data with stochastic system variation pattern. The noise of log events generally contains three categories of information. The log sequences may lose some log events, repeat some log events, or mess

Table 1 The summary of adopted benchmark datasets

Datasets	# of labeled blocks	# of train	# of test
HDFS(10%)	55,822	1,684	1,117,563
HDFS(20%)	111,644	3,368	2,235,126
HDFS(30%)	167,466	5,052	3,352,689
HDFS(50%)	279,110	8,420	5,587,815
Hadoop Application(10%)	76	14	5,538
Hadoop Application(20%)	152	28	11,076
Hadoop Application(30%)	228	42	16,614
Hadoop Application(50%)	380	70	27,690

up log events. Therefore, we randomly perturb the original HDFS dataset with different ratio. Figure 2 demonstrates the three different ways of injecting noises into log sequences.

In terms of evaluation metrics, we consider three types of metrics including precision ($Precision = \frac{TP}{TP+FP}$), recall ($Recall = \frac{TP}{TP+FN}$), and F1-score ($F1 - score = \frac{2(Precision \times Recall)}{Precision+Recall}$), to assess the performance of SSDLog.

4.2 Evaluation on HDFS

First of all, we compare our proposed framework with our previous approach [10], which is implemented by the vanilla CNN. To quantify model performance and assess their difference, three metrics are selected, including model precision, recall, and F1-score. The results are represented in Table 2. SSDLog outperforms by a large margin.

In addition, we compare SSDLog with the other two log-based anomaly detection approaches, including DeepLog [7] and LogAnomaly [11], on HDFS dataset. DeepLog is an LSTM-based model for log sequence prediction and efficient anomaly detection. LogAnomaly is an end-to-end model with LSTM network to detect abnormal log sequences.

Noting that, as the abovementioned three approaches did not consider the semi-supervised training conditions, we only give them labeled data for training.

From the given comparison results represented in Table 2, we observe that our proposed SSDLog outperforms vanilla CNN, DeepLog and LogAnomaly with different on all scores. Specifically, when β =50%, SSDLog gains the best performance on recall of 99.4%, exceeding DeepLog by 0.5%, LogAnomaly by 1.7%, and the vanilla CNN model by 3%. This indicates that SSDLog can better identify the abnormal behaviours in the system procedure and make accurate anomaly prediction. In addition, SSDLog obtains the highest F1-score of 98.7%, outperforming DeepLog significantly by 4.4%, LogAnomaly by 2.8%, and the vanilla CNN model by 1.6%. This implies that our model consistently outperforms these approaches across all metrics.

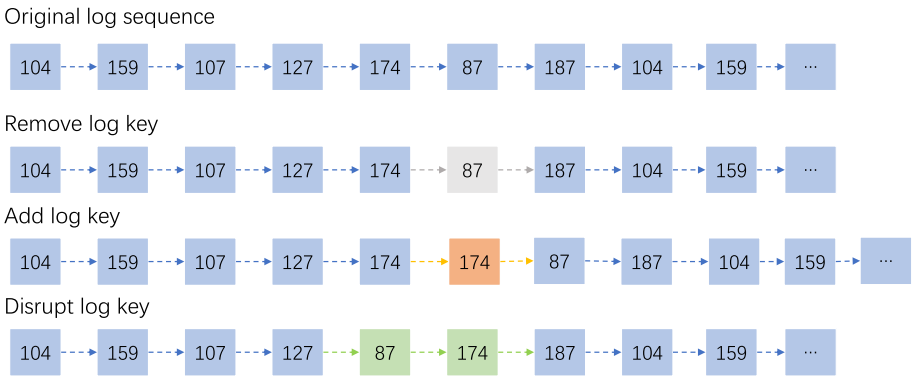

Figure 2 Illustration of log events in the synthetic data. The synthetic data can be divided into three categories: remove log event, add log event, and disrupt log event. The log sequence in the first line is the original log sequence

Table 2 The comparison of our previous models on HDFS dataset with various SSL conditions

β	Approach	Precision	Recall	F1-score
5%	CNN	0.939	0.789	0.858
	DeepLog	0.793	0.603	0.680
	LogAnomaly	0.924	0.534	0.677
	SSDLog	**0.958**	**0.946**	**0.951**
10%	CNN	0.962	0.858	0.907
	DeepLog	0.763	0.870	0.825
	LogAnomaly	0.777	0.829	0.800
	SSDLog	**0.971**	**0.981**	**0.976**
20%	CNN	0.975	0.904	0.938
	DeepLog	0.869	0.975	0.919
	LogAnomaly	0.922	0.960	0.941
	SSDLog	**0.978**	**0.986**	**0.982**
30%	CNN	0.972	0.940	0.954
	DeepLog	0.913	0.967	0.939
	LogAnomaly	0.925	0.965	0.945
	SSDLog	**0.978**	**0.992**	**0.985**
50%	CNN	0.978	0.964	0.971
	DeepLog	0.901	0.989	0.943
	LogAnomaly	0.941	0.977	0.959
	SSDLog	**0.980**	**0.994**	**0.987**

The best results are in bold

4.3 Evaluation on hadoop

To further assess the effectiveness of SSDLog, we compare our model with other competitors on the dataset of Hadoop Application. As listed in Table 3, our method outperforms the other approaches on precision and F1-score. Although recall and precision have slightly lower than the other methods when $beta = 5\%$ and 10%, separately, F1-score of SSDLog is always higher than the other methods. The SSDLog obtains the highest F1-score of 89.1% outperforming DeepLog by 13.1%, LogAnomaly by 9.2%, and vanilla CNN by 4.5%. This observation justifies that SSDLog exhibits more accurate anomaly detection and higher model robustness. The recall scores on Hadoop Application are lower than HDFS datasets because there are some challenging anomalous events, which interferes the detection process. For instance, an unexpected system shut-down may result in an incomplete sequence lacking of some logkeys. However, this kind of errors is difficult to be detected, since all the patterns in sliding windows in the model would be identified as correct.

4.4 Evaluation on industry dataset

To demonstrate the capability of SSDLog, we compare SSDLog with three state-of-art methods on real-world industry dataset. The results are shown in Figure 3. It clearly depicts that SSDLog obtains the best F1-score among all approaches, outperforming CNN by

Table 3 The comparison of our previous models on Hadoop Application dataset with various SSL conditions

β	Approach	Precision	Recall	F1-score
5%	CNN	0.683	0.700	0.691
	DeepLog	0.648	0.607	0.627
	LogAnomaly	0.528	**0.769**	0.626
	SSDLog	**0.862**	0.625	**0.725**
10%	CNN	**0.935**	0.642	0.761
	DeepLog	0.725	0.565	0.635
	LogAnomaly	0.521	**0.775**	0.623
	SSDLog	0.922	0.701	**0.797**
20%	CNN	0.875	0.700	0.778
	DeepLog	0.791	0.584	0.672
	LogAnomaly	0.722	0.605	0.658
	SSDLog	**0.895**	**0.850**	**0.872**
30%	CNN	0.886	0.775	0.827
	DeepLog	0.870	0.626	0.728
	LogAnomaly	0.802	0.684	0.739
	SSDLog	**0.932**	**0.821**	**0.873**
50%	CNN	0.929	0.776	0.846
	DeepLog	0.926	0.644	0.760
	LogAnomaly	0.831	0.770	0.799
	SSDLog	**0.934**	**0.851**	**0.891**

The best results are in bold

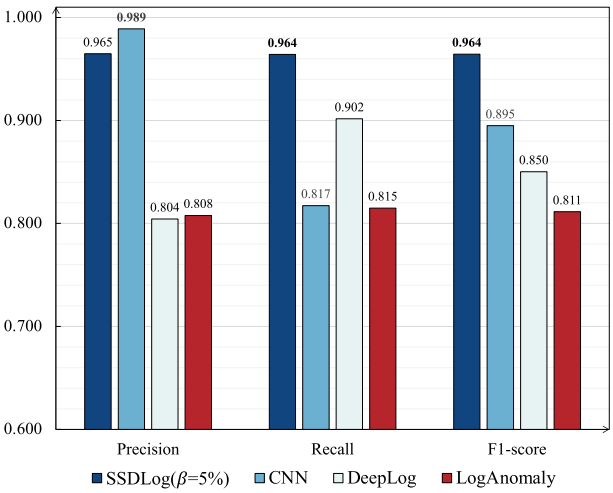

Figure 3 F1-score of different models on the industry dataset. The best results are in bold

6.9%, DeepLog by 11.4% and LogAnomaly by 15.3%. What's more, SSDLog achieves a higher recall score than the others, outperforming CNN by 14.7%, DeepLog by 6.2% and LogAnomaly by 14.9%. As observed, the proposed SSDLog has a superior performance in anomaly detection. It is noteworthy that SSDLog only applies 5% labeled training set, further proving the effectiveness of SSDLog.

4.5 Evaluation on real-world unstable dataset

In addition, to verify the capability of anomaly detection in real-world SSL conditions, we compare SSDLog with vanilla CNN, DeepLog and LogAnomaly on the aforementioned unstable dataset. The comparison results on the unstable dataset are presented in Figure 4. It is observed that our proposed SSDLog significantly archives better performance than the other three methods. With the increase of the noise ratio, SSDLog exhibits remarkable anomaly detection accuracy than the other three methods. In addition, DeepLog and LogAnomaly show weaker detection ability on the noisy data, since the classifier from Deeplog and LogAnomaly has some deviation which may consider all unseen log patterns as anomalies.

5 Ablation study and analysis

Though SSDLog performs well on anomaly detection, a small part of log sequences are still classified incorrectly. To reveal the reasons of these misjudgments, we further analyze the misclassified cases of HDFS. We find out that when log key are identical, the log can also be either normal or abnormal. This suggests that the property of abnormal logs does not only rely on log keys. Obviously, the latent information hidden in the log values and log semantics also have certain effects in the process of log anomaly detection. Therefore, we employ one type of log values, i.e, timestamp, to distinguish the aforementioned type of log sequences with an unsupervised method. For these log cases with the same log key sequence, we extract timestamp intervals between relevant log events. These timestamp intervals are embedded

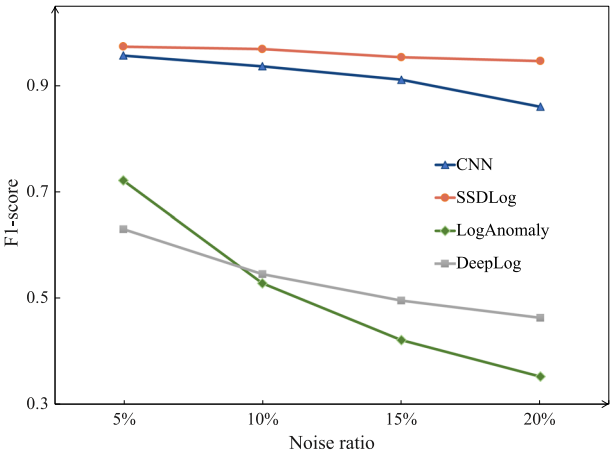

Figure 4 F1-score of different models on the synthetic unstable dataset. Different marked curve denotes different approaches

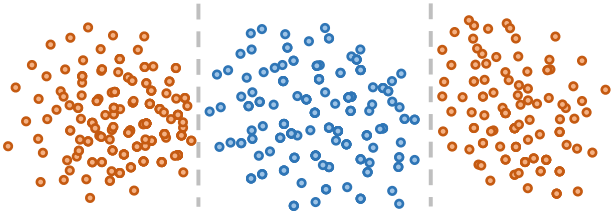

Figure 5 Distribution of log timestamp intervals produced by identical log key sequences. The blue and orange dots represent the time intervals of abnormal log and normal log, respectively. The gray dashed lines denote the decision boundaries generated by GMM

into vectors, which are fed to a Gaussian Mixture Model (GMM). The GMM assumes that all vectors are generated from a mixture of Gaussian distributions. This GMM can separate normal and abnormal logs according to different probabilistic distributions.

An instance of timestamp intervals' distribution is illustrated in Figure 5. It is observed that the clusters of normal and abnormal sequences have an obvious distance between each other. The log events are divided by decision boundaries generated by GMM. Though these logs have identical log key sequences, they can be correctly classified into abnormal and normal logs due to different timestamp interval distributions.

It implies that better anomaly identification can be obtained by further integrating log value with log keys. Among different log values, the timestamp interval is a good choice since it reflects system response and delay, and facilitates efficient anomaly detection.

6 Conclusion and future work

In this paper, we proposed a novel approach named SSDLog to detect log anomaly in semi-supervised condition. our model adopted two-branch teacher-student model for SSL. In addition, to further utilize unlabeled samples effectively, we proposed a flexible label screening strategy that takes into account the confidence and stability of pseudo-labels. To validate the effectiveness, we compare SSDLog with other state-of-art approaches. The experimental results shown that our SSDLog achieved a higher performance than the other methods in semi-supervised log anomaly detection task.

In the future, to further alleviate the need for data collection, we will explore the few-shot learning for log anomaly detection. Besides, more datasets with valuable information will be suggested to extend usage scenarios of SSDLog, such as CPS data, IoT data, and other system monitoring data.

Author Contributions Siyang Lu wrote the main manuscript, prepared Figures 1- 2, and proposed methodology. Ningning Han wrote the main manuscript text, prepared Figures 3–4, and conducted experiments. Mingquan Wang prepared dataset, wrote, reviewed and edited the main manuscript. Xiang Wei reviewed and edited the manuscript, validated the methodology. Zaichao Lin conducted investigation and wrote the related work. Dongdong Wang conducted investigation, reviewed and edited the manuscript. All authors reviewed the manuscript.

Funding This research was supported by the National Natural Science Foundation of China (No.62006016) and the National Key Research and Development Program of China (No.2021YFB2900704).

Data Availability All the log datasets leveraged in the experiment can be found at https://zenodo.org/record/3227177

Declarations

Competing interests The authors have no competing interests as defined by Springer, or other interests that might be perceived to influence the results and/or discussion reported in this paper.

Ethical Approval Not applicable.

References

1. Fulp, E.W., Fink, G.A., Haack, J.N.: Predicting computer system failures using support vector machines. WASL **8**, 5–5 (2008)
2. Liang, Y., Zhang, Y., Xiong, H., Sahoo, R.: Failure prediction in ibm bluegene/l event logs. In: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference On, pp. 583–588. IEEE (2007)
3. Yadwadkar, N.J., Ananthanarayanan, G., Katz, R.: Wrangler: predictable and faster jobs using fewer resources. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 1–14 (2014). ACM
4. Chen, M., Zheng, A.X., Lloyd, J., Jordan, M.I., Brewer, E.: Failure diagnosis using decision trees. In: International Conference on Autonomic Computing (2004)
5. Lou, J.-G., Fu, Q., Yang, S., Xu, Y., Li, J.: Mining invariants from console logs for system problem detection. In: 2010 USENIX Annual Technical Conference (USENIX ATC 10) (2010)
6. Yang, L., Chen, J., Wang, Z., Wang, W., Jiang, J., Dong, X., Zhang, W.: Semi-supervised log-based anomaly detection via probabilistic label estimation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1448–1460 (2021). IEEE
7. Du, M., Li, F., Zheng, G., Srikumar, V.: Deeplog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298 (2017)
8. Brown, A., Tuor, A., Hutchinson, B., Nichols, N.: Recurrent neural network attention mechanisms for interpretable system log anomaly detection. arXiv:1803.04967 (2018)
9. Xia, B., Bai, Y., Yin, J., Li, Y., Xu, J.: Loggan: a log-level generative adversarial network for anomaly detection using permutation event modeling. Inf. Syst. Front., 1–14 (2020)
10. Lu, S., Wei, X., Li, Y., Wang, L.: Detecting anomaly in big data system logs using convolutional neural network. In: 2018 4th Intl Conference on Cyber Science and Technology Congress (CyberSciTech), pp. 151–158. IEEE (2018)
11. Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., Chen, Y., Zhang, R., Tao, S., Sun, P., et al.: Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI, vol. 19, pp. 4739–4745 (2019)
12. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 117–132 (2009)
13. Safyallah, H., Sartipi, K.: Dynamic analysis of software systems using execution pattern mining. In: Program Comprehension, 2006. ICPC 2006. 14th IEEE International Conference On (2006)
14. Fu, Q., Lou, J.G., Wang, Y., Li, J.: Execution anomaly detection in distributed systems through unstructured log analysis. In: Ninth IEEE International Conference on Data Mining, pp. 149–158 (2009)
15. Farzad, A., Gulliver, T.A.: Unsupervised log message anomaly detection. ICT Express (2020)
16. Han, N., Lu, S., Wang, D., Wang, M., Tan, X., Wei, X.: Skdlog: self-knowledge distillation-based cnn for abnormal log detection. The 19th IEEE International Conference on Ubiquitous Intelligence and Computing (2022)

17. Chen, R., Zhang, S., Li, D., Zhang, Y., Liu, Y.: Logtransfer: cross-system log anomaly detection for software systems with transfer learning. In: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) (2020)
18. Duan, X., Ying, S., Yuan, W., Cheng, H., Yin, X.: Qllog: a log anomaly detection method based on q-learning algorithm. Inf. Process. Manag. **58**(3), 102540 (2021)
19. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. arXiv:1610.02242 (2016)
20. Wei, X., Gong, B., Liu, Z., Lu, W., Wang, L.: Improving the improved training of wasserstein gans: a consistency term and its dual effect. In: International Conference on Learning Representations(ICLR) (2018)
21. Berthelot, D., Carlini, N., Cubuk, E.D., Kurakin, A., Sohn, K., Zhang, H., Raffel, C.: Remixmatch: semi-supervised learning with distribution alignment and augmentation anchoring. In: International Conference on Learning Representations(ICLR) (2020)
22. Wang, D., Liu, Q., Wu, D., Wang, L.: Meta domain generalization for smart manufacturing: tool wear prediction with small data. J. Manuf. Syst. **62**, 441–449 (2022)
23. Wang, D., Gong, B., Wang, L.: On calibrating semantic segmentation models: analysis and an algorithm. arXiv:2212.12053 (2022)
24. Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Pseudo-labeling and confirmation bias in deep semi-supervised learning. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
25. Wei, X., Wei, X., Xing, W., Lu, S., Lu, W.: An incremental self-labeling strategy for semi-supervised deep learning based on generative adversarial networks. IEEE Access **8**, 8913–8921 (2020)
26. Wang, Y., Chen, H., Heng, Q., Hou, W., Savvides, M., Shinozaki, T., Raj, B., Wu, Z., Wang, J.: Freematch: self-adaptive thresholding for semi-supervised learning. arXiv:2205.07246 (2022)
27. Kong, X., Wei, X., Liu, X., Wang, J., Lu, S., Xing, W., Lu, W.: 3lpr: a three-stage label propagation and reassignment framework for class-imbalanced semi-supervised learning. Knowledge-Based Systems **253**, 109561 (2022)
28. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.-L.: Fixmatch: simplifying semi-supervised learning with consistency and confidence. Adv. Neural Inf Process. Syst. **33**, 596–608 (2020)
29. Wei, X., Wei, X., Kong, X., Lu, S., Xing, W., Lu, W.: Fmixcutmatch for semi-supervised deep learning. Neural Netw **133**, 166–176 (2021)
30. Wang, D., Li, Y., Wang, L., Gong, B.: Neural networks are more productive teachers than human raters: active mixup for data-efficient knowledge distillation from a blackbox model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1498–1507 (2020)
31. Tarvainen, A., Valpola, H.: Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results. arXiv:1703.01780 (2017)
32. Miyato, T., Maeda, S.-I., Koyama, M., Ishii, S.: Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE Trans Pattern Anal Mach Intell **41**(8), 1979–1993 (2018)
33. Verma, V., Kawaguchi, K., Lamb, A., Kannala, J., Solin, A., Bengio, Y., Lopez-Paz, D.: Interpolation consistency training for semi-supervised learning. Neural Netw **145**, 90–106 (2022)
34. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: Mixup: beyond empirical risk minimization. In: International Conference on Learning Representations(ICLR) (2017)
35. Jiang, Y., Li, X., Chen, Y., He, Y., Xu, Q., Yang, Z., Cao, X., Huang, Q.: Maxmatch: semi-supervised learning with worst-case consistency. IEEE Trans Pattern Anal Mach Intell (2022)
36. Huang, J., Huang, A., Guerra, B.C., Yu, Y.-Y.: Percentmatch: percentile-based dynamic thresholding for multi-label semi-supervised classification. arXiv:2208.13946 (2022)
37. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
38. Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y., Chen, X.: Log clustering based problem identification for online service systems. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 102–111 (2016)

Authors and Affiliations

Siyang Lu[1] · Ningning Han[1] · Mingquan Wang[2] · Xiang Wei[2] · Zaichao Lin[1] · Dongdong Wang[3]

Ningning Han
22120493@bjtu.edu.cn

Mingquan Wang
20121736@bjtu.edu.cn

Xiang Wei
xiangwei@bjtu.edu.cn

Zaichao Lin
lzc_eastchina@163.com

Dongdong Wang
daniel.wang@knights.ucf.edu

1 School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China

2 School of Software Engineering, Beijing Jiaotong University, Beijing, China

3 Department of Civil Environmental and Construction Engineering, University of Central Florida, Orlando, USA