Computer Systems Science & Engineering
DOI:10.32604/csse.2021.014030
Article

Tech Science Press

A Generative Adversarial Networks for Log Anomaly Detection

Xiaoyu Duan[1], Shi Ying[1,*], Wanli Yuan[1], Hailong Cheng[1] and Xiang Yin[2]

[1]School of Computer Science, Wuhan University, Wuhan, 430072, China
[2]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, 100093, China
[*]Corresponding Author: Shi Ying. Email: yingshi@whu.edu.cn
Received: 30 August 2020; Accepted: 22 September 2020

Abstract: Detecting anomaly logs is a great significance step for guarding system faults. Due to the uncertainty of abnormal log types, lack of real anomaly logs and accurately labeled log datasets. Existing technologies cannot be enough for detecting complex and various log point anomalies by using human-defined rules. We propose a log anomaly detection method based on Generative Adversarial Networks (GAN). This method uses the Encoder-Decoder framework based on Long Short-Term Memory (LSTM) network as the generator, takes the log keywords as the input of the encoder, and the decoder outputs the generated log template. The discriminator uses the Convolutional Neural Networks (CNN) to identify the difference between the generated log template and the real log template. The model parameters are optimized automatically by iteration. In the stage of anomaly detection, the probability of anomaly is calculated by the Euclidean distance. Experiments on real data show that this method can detect log point anomalies with an average precision of 95%. Besides, it outperforms other existing log-based anomaly detection methods.

Keywords: Generative adversarial networks; anomaly detection; data mining; deep learning

1 Introduction

Logs record system states and application behaviors, Operation & Maintenance Personnel (OPS) usually analyze system logs to locate the fault [1]. Besides, such log data is universally available in nearly all computer systems. Therefore, system logs are an essential data source for performance monitoring, understanding system status and anomaly detection.

System anomalies can be classified into three groups: point anomalies, execution sequence anomalies, and collective or group anomalies. This paper focuses on detecting log point anomalies. Traditional point anomaly detection methods are usually based on rules. There are many defects in traditional methods, for example, (1) it needs domain knowledge; (2) it is difficult to identify the unknown faults; (3) with the growth of log data and log update, it is challenging to build rule base effectively. Therefore, traditional anomaly detection methods are no longer workable. In log anomaly detection, it is not easy to get the labeled data. Therefore, most existing popular approaches that leverage system log data for log point

anomaly detection are based on unsupervised and semi-supervised technology, such as clustering [2], isolation forest [3], and support vector machines (SVM) [4,5]. The challenges of log point anomaly detection can be summarized as follows:

- Uncertainty: It is difficult to define the characteristics of all abnormal logs. Besides, the boundary between normal logs and abnormal logs is unclear. Therefore, the outlier observations close to the edge might be a normal log.
- Imbalance: The training samples belonging to normal logs are much more than ones belonging to abnormal logs, and it is not easy to gain the labeled log datasets.
- Unstructured: Log data has unstructured characteristics.

Our research is based on unsupervised technology Generative Adversarial Network [6], namely GAN-EDC (GAN based on Encoder-Decoder framework and Convolutional Neural Networks [7]). In recent years, GAN has made significant progress in many fields, such as image detection [8,9], image segmentation [10], and speech production [11]. To apply GAN in detecting abnormal logs, we use the Encoder-Decoder framework as the generator, Convolutional Neural Networks as the discriminator. The encoder maps the input log keywords to potential representation and then generates the corresponding log template through the decoder. CNN discriminator identifies the differences between the generated log template and the actual template and then gives a scalar (realistic or fake). The model parameters are optimized automatically by iteration. Finally, we use the Euclidean distance to calculate the abnormal value of logs. Compared with the existing unsupervised point anomaly detection methods, this method can (1) update the optimization model parameters through the confrontation between generator and discriminator; (2) no boundary needs to be found for anomaly detection, which avoids the influence of imbalanced data; (3) not limited to detecting known types of abnormal logs and has high versatility. We evaluated GAN-EDC on real log datasets with over 300 thousand log entries. The results show that GAN-EDC has high accuracy.

The primary contributions of this paper are as follows:

- We propose a detection model aimed at log point anomaly, namely GAN-EDC.
- We study the feasibility of GAN in log anomaly detection and the effect of parameter k value on the model.
- We summarize and compare a variety of log point anomaly detection methods and prove the superiority of GAN-EDC.

The rest of this paper is organized as follows: Section 2 presents related work on log anomaly detection. Section 3 introduces the basic structure and the pre-processing of the log dataset. Section 4 presents the design and construction of GAN-EDC. We evaluated the performance of GAN-EDC in Section 5. Finally, Section 6 presents the final remarks.

2 Related Work

Anomaly detection has long played an essential role in a wide variety of fields such as internet security [12], software security [13], web security [14], social network [15], big data [16]. Errors in the data can cause anomalies, but anomalies sometimes indicate a new, previously unknown, underlying process. As early as 1987, Paper [17] has defined the anomalies. Enderlin pointed out that the anomaly data is the data that deviates from other observation data seriously (i.e., abnormal instances are markedly different from normal instances). Therefore, anomaly detection is also called outlier detection. Existing approaches that leverage log data for log point anomaly detection can be broadly classified into four groups: rule-based methods, statistics-based methods, machine learning algorithm-based methods, and deep learning-based methods.

Rule-based methods are put forward early. Detection is a two-step process. These methods first use the algorithm to get the rules or make rules by the expert and then determine whether it is an abnormal log according to the rules. These methods have high detection accuracy but only successful in specific scenarios. For example, Ren et al. [18] uses the Apriori algorithm to find the association rules and frequent episodes in log datasets. The disadvantages of this method are limited by the domain knowledge. If there are new types of abnormal, the existing rule base is ineffective.

Statistics-based methods considered that the normal log entries are in the high probability interval of the random model. On the contrary, the abnormal log entries are usually in the low probability interval. These methods first build a statistical model for the given normal dataset and determine whether the newly arrived data is the same as the statistical model. For example, Goldstein et al. [19] proposed an anomaly detection method based on the histogram. It first models univariate feature densities using histograms with a fixed or a dynamic bin width. Then, all histograms are used to compute an anomaly score for each data instance. These methods are over-reliant on strong independence between features, so it is difficult to detect anomalies in real log datasets. Statistical-based methods also include literature [20].

In recent years, machine learning-based methods and deep learning-based anomaly detection methods are popular research directions. These two methods can be divided into supervised, semi-supervised and unsupervised. Supervised methods have higher accuracy, such as decision trees [21] and random forest [22]. However, supervised anomaly detection methods generally require a lot of labeled samples to accomplish their training tasks. Besides, it is difficult for experts to label all data manually, and anomalies rarely occur in log datasets. Therefore, the semi-supervised method is widely used. The normal techniques include Recursive Neural Network (RNN) [23], CNN [24], and Restricted Boltzmann Machine (RBM) [25]. Compared with supervised anomaly detection methods, although semi-supervised anomaly detection methods need fewer labeled objects, it still needs a certain number of labeled objects to ensure accuracy. Unsupervised anomaly detection methods such as LSTM [1], Isolation Forest (IForest) [3] and Cluster [26] do not need the labeled data. Paper [2] proposed LogCluster, an approach that clusters the logs to ease log-based problem identification. LogCluster facilitates problem identification by clustering similar log sequences and retrieving recurrent issues. This model first clusters log sequence to construct an initial knowledge base. Second, it analyzes the log sequences and checks if the clusters can be found in the knowledge base. In the end, it only examines a small number of representative log sequences from the clusters that are previously unseen. LogCluster improves the effectiveness, but the strategy is too coarse-grained. Du proposes DeepLog [1], a deep neural network model utilizing Long Short-Term Memory, to model a system log as a natural language sequence. DeepLog uses the normal log dataset in the training stage to train a log key anomaly detection model for diagnosis purposes. In the detection stage, it compares the predicted results and the actual results to determine whether the system is abnormal or not. Papers that model log sequences as natural language sequences also include [27,28].

Ian Goodfellow proposed the Generative Adversarial Networks network in 2014, which is widely used in the field of image processing [29–32] and has an excellent performance. In 2017, Yu et al. [33] was the one who first applies GAN in text generation, namely SeqGAN. Inspired by SeqGAN, we use GAN in log point anomaly detection. We compare the generated log entry and the log entry generated by the system to determine whether the system is normal. The experimental results show the superiority of GAN in log anomaly detection.

3 Log Parser

Log parsing is the basis of anomaly detection. In log anomaly detection, it is not feasible to detect anomalies on massive unstructured logs directly. Therefore, we need to parse unstructured, free-text log entries into a structured representation to learn log patterns over a structured dataset.

The log dataset contains both template words and parameter words. Parameter words have no semantic information and usually appear in different forms. Template words reveal the event template of the log entry and remain the same for every event occurrence. Each log entry includes three parts: timestamp, log type, and log event. Timestamp records the time when an event occurs, log type usually shows the severity level of the event (e.g., "INFO" or "WARN"), and log event records the system execution information. The original log entries, the log templates of each raw log entry and classification log data are shown in Fig. 1.

No.	Raw log data
1	081109 204917 35 WARN dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: Redundant addStoredBlock request received for blk_3888635850409849568 on 10.251.107.227:50010 size 67108864
2	081110 103340 7159 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-8356640976087556977 terminating
3	081110 103340 7159 INFO dfs.DataNode$PacketResponder: Received block blk_-8356640976087556977 of size 67108864 from /10.251.71.16

Log classification

Log time stamp	081109 204917 35, 081110 103340 7159
Log type	INFO, WARN
Log subtypes	dfs. FSNamesystem, dfs. DataNode$PacketResponder
Log event	BLOCK* NameSystem.addStoredBlock: Redundant addStoredBlock request received for blk_3888635850409849568 on 10.251.107.227:50010 size 67108864
	PacketResponder 1 for block blk_-8356640976087556977 terminating
	Received block blk_-8356640976087556977 of size 67108864 from /10.251.71.16
Log templates	**dfs FSNamesystem BLOCK NameSystem addStoredBlock Redundant addStoredBlock request received for on size**
	dfs DataNode PacketResponder PacketResponder for block terminating
	dfs DataNode PacketResponder Received block of size from
Log keywords	*dfs FSNamesystem BLOCK*
	dfs DataNode PacketResponder PacketResponder
	dfs DataNode PacketResponder

Figure 1: Raw logs from HDFS

As seen in Fig. 1, the parameter words are highlighted in red. We first parses raw logs to find the log templates. Second, we extract the keywords of each log template. We select the log type, log subtype and the first word of log event as the log keywords. Note that each log keyword is a log template word. Finally, we build a dictionary C, which includes all log template words. The log keywords are shown in the log keywords item of Fig. 1.

Let $LD = \{d_1,\ d_2, \cdots, d_n\}$ be the log dataset, each d_i in LD is a log entry. For example, the dataset in Fig. 1 is $LD = \{d_1,\ d_2, d_3\}$. Each log entry d_i is composed of multiple words, clearly $d_i = \{w_1,\ w_2, \cdots, w_n\}$, where each w_i is a word (template word or parameter word). If w_i occurs frequently (i.e. w_i has a large support), w_i will be a template word. In log template extraction, (1) the first step is to scan the whole log dataset and use the regular expression to delete symbols and parameter words with a fixed format, such as IP address, mac address, and file path. Remarkably, special symbols are not in the research scope of this paper. We purposely count the symbols of six data sets with 10MB. The results are shown in Tab. 1.

There are no special symbols in these data sets, and we can easily delete these symbols by using the regular expression. (2) Then, we compute the number of times that word w_i in log event has occurred in the dataset (i. e., Word Frequency) and derive a list F of words in descending order of their Word Frequency. Each word has a fixed serial number (ID). (3) When a new log entry arrives. We create the root of a tree, which is labeled with the log type. In our case, it is "INFO" and "WARN". Then we construct the first path of the tree according to the ID of the words in list F. When the next log entry is processed, if the current log entry has the common prefix word with processed log entries, then share a common prefix with the existing path; otherwise, a new branch is created as a sub-tree of the node. Finally, we prune the tree. For example, if a node is not in the top 5 nodes and has too many children (exceeding a threshold k), all its children (or sub-trees) are deleted from the tree, and the node becomes leaf itself. Each root-to-leaf path is a log template. The algorithm of extract log template is shown in Algorithm 1.

Table 1: The count result of symbol on six log datasets

	HDFS	OpenStack	Spark	Hadoop	Zookeeper	BGL
.	400450	1005200	205750	728900	228850	623250
:	250500	615000	333050	528200	501450	225550
/	177800	352750	223850	30600	69950	34350
−	67350	1514500	8900	394100	407250	1179250
_	163700	181350	68850	178850		7000
$	52850			650	71700	
*	32950			100	14250	3500
(	50	15800	37900	16050	6850	11800
)	50	15800	37900	16050	6850	11800
[		127500	100	118100	108150	
]		127500	100	118100	108150	
"		101700				
,		23700	88200	122100	134900	28250
=		6250	75000	25150	37950	14150
‘		700	3200	100		
%		400				
&		150				
?		100				
@			100	32750	100050	
;			350	7800	2000	
+			2250			550
<				7400		
>				7400		50
#				3600		
\				350		
{				150		
}				150		
!				50		100

4 Design and Construction

In this section, we first describe traditional Generative Adversarial Networks. Second, we introduce the generator model based on the Encoder-Decoder framework and the discriminator model based on Convolutional Neural Networks. Finally, we present the process of anomaly detection by using the anomaly detection model. The GAN-EDC architecture is shown in Fig. 2 with three key components: generator model, discriminator model and anomaly detection model.

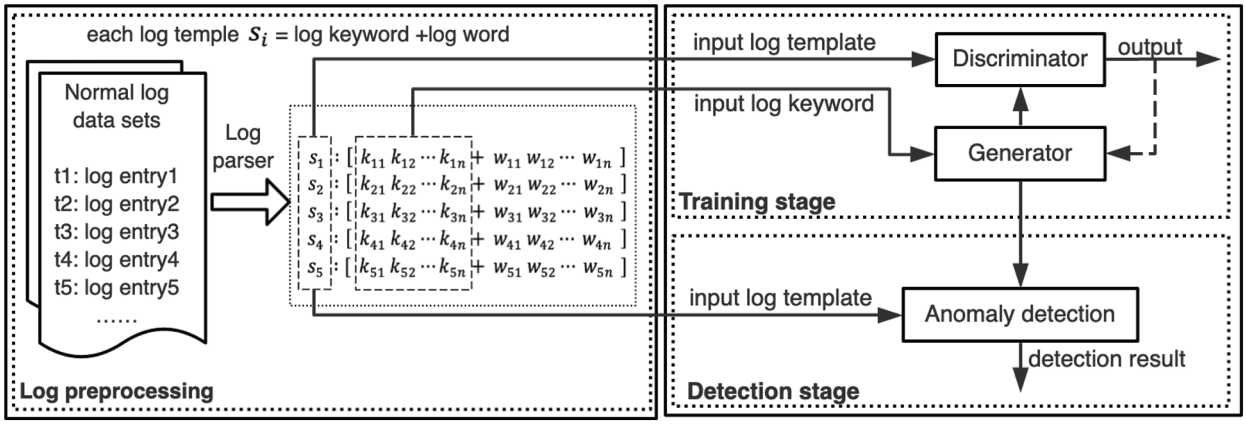

Figure 2: GAN-EDC architecture

Training stage. Training data for GAN-EDC are log entries from the normal system. Each log entry is parsed to a log template. Then GAN-EDC extracts log keywords of each log entry. We input the log keyword into the generator and train the generator to generate instances of log template. The input of the discriminator is the log template generated by the generator and the real log template. The discriminator is to distinguish between the real log template and the generated sample.

Detection stage. A newly arrived log entry is parsed to log keywords and log words. The anomaly detection model uses the log template and the generated sample to check whether the incoming log entry is normal. The output of the anomaly detection model is the detection result (normal log entry or abnormal log entry).

Algorithm 1: extract log template

Input: A log dataset D and a threshold k

output: A log template tree T

1: Scan the log dataset D.
2: Calculate the Word Frequency for each log word
3: Derive a list F of log words in descending order of their Word Frequency.
4: Create the root.
5: **for** each log entry in D **do**
6: Sort the log words according to the order in list F
7: **end for**
8: **for** Child node N in T **do**
9: **if** N is not the top 5 node and has more than k children **then**
10: Eliminate all the children of N
11: **end if**
12: **end if**
13: **return** T

4.1 Generative Adversarial Networks

GAN is a dynamic game model, which comprises two adversarial models: a generator G that captures the data distribution, and a discriminator D that estimates the probability that a sample came from the training data rather than generator. In the process of dynamic game training, the purpose of the generator is to increase the error probability of discriminator, the purpose of the discriminator is distinguishing between the real data and the generated samples. Training of GANs involves both finding the parameters of a discriminator that maximize its classification accuracy and finding the parameters of a generator which maximally confuse the discriminator.

Set a prior on input noise variables $P_z(z)$, then represent a mapping to data space as $G\left(z;\theta_g\right)$, where G is a differentiable function represented by a multilayer perceptron with parameters θ_g. Set second multilayer perceptron $D(x;\theta_d)$, x is the data, θ_d is the parameter. $D(x)$ represents the probability that x came from the data rather than the generator's distribution p_g. D and G play the following two-player minimax game with value function $V(G,D)$.

$$\min_G \max_D V(D,G) = \mathbb{E}_{x-P_{data}(x)}[\log D(x)] + \mathbb{E}_{z-P_z(z)}[log(1-D(G(z)))] \tag{1}$$

4.2 Generative Based on Encoder-Decoder

In our research, we choose the Encoder-Decoder framework based on LSTM to build the generative model G. First, we input the log keyword into the LSTM of the encoder phase and get the hidden state with fixed dimension. After encoder processing, input the hidden state into the LSTM of the decoder phase. Finally, we get the log template from the generator (i.e., we train the generator to generate the instances of log template).

Encoder. The LSTM encoder learns fixed-length vector representation of the input. Given a log dataset $S=\{s_1,\ s_2,\cdots,s_n\}$, where n is the total number of log templates, s_i is a log template. Each log template s_i is composed of multiple words, clearly s_i=$\{w_1,\ w_2,\cdots,w_n\}$, where each w_i denotes the i-th word in log template s_i. Some of these words are log keywords, let log keyword of s_i is $K=\{k_1,\ k_2,\cdots,k_n\}$ ($K\in s_i$), k_i is a log keyword. Consider a keyword sequence $K=\{k_1,\ k_2,\cdots,k_L\}$ of length L, $h_E^{(i)}$ is the hidden state of the encoder at time t_i for each $i\in\{1,2,\cdots,L\}$, where $h_E^{(i)}\in R^c$, c is the number of LSTM units in the hidden layer of the encoder. Enter the log keywords into the LSTM encoder. Memory Cell is the core part of the LSTM, composed of parameters and a gating unit system. It is used to judge whether the information is useful or not. Each gating unit system includes three gates, namely input gate, forgetting gate and output gate. LSTM uses a separate memory cell to remember long term dependencies, which can be updated depending on the current input. At each time step, an LSTM takes current input x_i and previous memory cell state $h_E^{(i-1)}$ as input and computes the current cell state $h_E^{(i)}$. Finally, we get the end hidden state $h_E^{(L)}$.

Decoder. The final state $h_E^{(L)}$ of the encoder is used as the initial state for the decoder. For example, the decoder takes x_1 and $h_E^{(1)}\left(h_E^{(1)}=h_E^{(L)}\right)$ as input to obtain the next hidden state $h_E^{(2)}$. Processing continues in this same way. We can get the hidden state of each time. The hidden state of each time inputs into a Softmax function, and the output is a probability distribution describing the probability for each template word from the dictionary C. We choose the word with the highest probability as the output. The output can be expressed as

$$y_i = g\left(y_{i-1}, h_D^{(i)}, h_E^{(L)}\right) \tag{2}$$

where $h_E^{(i)}$ is the hidden state of decoder LSTM at the time t, y_{i-1} is the output of the previous time and take as the input as time t. g is a non-linear multi-layer perceptron, which produces the probability that each word in dictionary C belongs to y_i.

4.3 Discriminator Based on CNN

We use CNN to build the discriminator model of GAN, and the discriminator is a binary classifier. The input of the discriminator is the generated log template and the real log template. The output is a number between 0 and 1. The number represents the probability that the input log template is real.

The discriminator consists of a convolution layer and a max-pooling operation over the entire sentence for each feature map. Set the log template generated as $Y = \{y_1, y_2, \cdots, y_n\}$, each word y_t is embedded into a k-dimensional word vector $x_t = W_e[y_t]$, where $W_e \in \mathbb{R}^{k \times V}$ is a word embedding matrix (to be learned). Then a log template of length T (padded where necessary) is represented as a matrix $X \in \mathbb{R}^{k \times T}$, by concatenating its word embeddings as columns, i.e., the t-th column of X is x_t. Set the convolution kernel is $W_c \in \mathbb{R}^{k \times h}$, where h is the number of words. For example, the convolution kernel with a window size of one word can be expressed as $W_c \in \mathbb{R}^{k \times 1}$. First, we get the feature map $c = f(X * W_c + b)$ by convolution of the input matrix X, where $f(\cdot)$ is a nonlinear activation function, $*$ denotes the convolutional operator, b is a bias vector. Second, we apply a max-over-time pooling operation to the feature map and take its maximum value $\hat{c} = \max\{c\}$. Third, we apply the fully connected layer of sigmoid function to output a scalar, the scalar represents the probability of X is from the log data distribution, rather than from adversarial G.

Training. G and D share a set of parameters, which are updated alternately by a stochastic gradient descent method. G and D are trained simultaneously. The training strategy is to train the k-steps D and then train the 1-step G, i.e., The discriminator is trained until optimal with respect to the current generator. Then, the generator is updated again. We adjust parameters for G to minimize $log(1 - D(G(S)))$ and adjust parameters for D to minimize $logD(x)$, as if they are following the two-player min-max game with value function $V(D, G)$.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x - P_{data}(x)}[\log D(x)] + \mathbb{E}_{S - P_S}[log(1 - D(G(S)))] \tag{3}$$

where S is the keyword sequence.

4.4 4.4 Anomaly Detection

GAN-EDC has learned the feature distributions of normal log entries. Therefore, we design an abnormal score to evaluate the error between the input templates and templates from G. The abnormal score can be expressed as

$$A(X, \theta) = \| X - G(S) \|_2 \tag{4}$$

where $A(X, \theta)$ is the abnormal score, X is the input log entry, θ is the parameters of G. We select a suitable threshold ρ. If $A(X, \theta)$ is less than ρ, the input log entry is normal; otherwise, it is abnormal.

5 Evaluation

In this section, we first introduce the datasets and experimental environment. All log datasets are collected from the real-world. Second, we evaluate the performance of GAN-EDC and compare it with those methods, such as Cluster, SVM, decision tree, PCA. Finally, we assess the effect of the k value.

5.1 Log Data Sets and Experiment Setup

We use distributed system log datasets as training sets and test sets, including the HDFS dataset and BGL dataset. HDFS dataset is generated through running Hadoop MapReduce jobs on more than 100 Amazon's EC2 nodes. There are 29 log templates and 575056 execution sequences in the HDFS dataset, including normal and abnormal log. The Blue Gene/L supercomputer system log dataset contains 4747963 log entries and 385 log templates. There are 348698 (7%) log entries labeled as anomalies. These two datasets are labeled by experts. Details of the HDFS dataset and Blue Gene/L dataset could be found in these papers [1,34,35,36]. Zhu et al. [34] has officially released these datasets, and we have made no modification to these datasets in this paper. These datasets have been downloaded over 7000 times. Tab. 2 summarizes the two datasets.

Table 2: Log data sets

Dataset	Data Size	Log Entry	Log template
HDFS	1.45 GB	11197954	29
BGL	708MB	4747963	385

The platform used in sections 6.2, 6.3 and 6.4 is Python 3.7. All experiments use the same machine with an Intel (R) Xeon (R) Silver 2.20 GHz CPU, 128 G memory, and Windows 10 operating system.

5.2 Accuracy Evaluation of Anomaly Detection

In this section, we evaluate the accuracy of the GAN-EDC. The following three intuitive metrics usually assess a method's effect to detect abnormal log: *Precision*, *Recall* and $F-measure$. These evaluation metrics are used in previous studies, frequently [37–39]. Eqs. (5), (6) and (7) show the formulas of these three metrics.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F - measure = \frac{2Recall \cdot Precision}{Recall + Precision} \tag{7}$$

where TP is true positive, TN is true negative, FP is false positive, FN is false negative. $F-measure$ is the harmonic mean of *Precision* and *Recall*. We ran two supervised methods (i.e., decision tree and SVM) and two unsupervised methods (i.e., clustering and PCA) to detect the log point anomalies for the HDFS dataset and BGL dataset, respectively. For all methods, we choose the first 80% data as the training data and the remaining 20% as the testing data. Fig. 3 shows the comparison using *Precision*, *Recall* and $F-measure$.

As illustrated in Fig. 3. We can observe that supervised methods (namely decision tree and SVM) and GAN-EDC achieve high accuracy (over 0.95) in the HDFS dataset. Clearly, GAN-EDC has achieved the best overall performance in two datasets, especially in the BGL dataset with *Precision* 96%, *Recall* 89% and $F-measure$ 92%. The result proves that GAN-EDC can effectively separate normal instances and abnormal instances by using feature representation. Besides, the overall accuracy on the HDFS dataset is higher than the accuracy on the BGL dataset. Because the HDFS dataset only records 29 log templates, which is much less than that in the BGL dataset (it records 385 log templates). The detection strategy of

the cluster is too coarse-grained, so it has the lowest accuracy. Compared with the supervised methods, all unsupervised methods relatively low accuracy on two datasets.

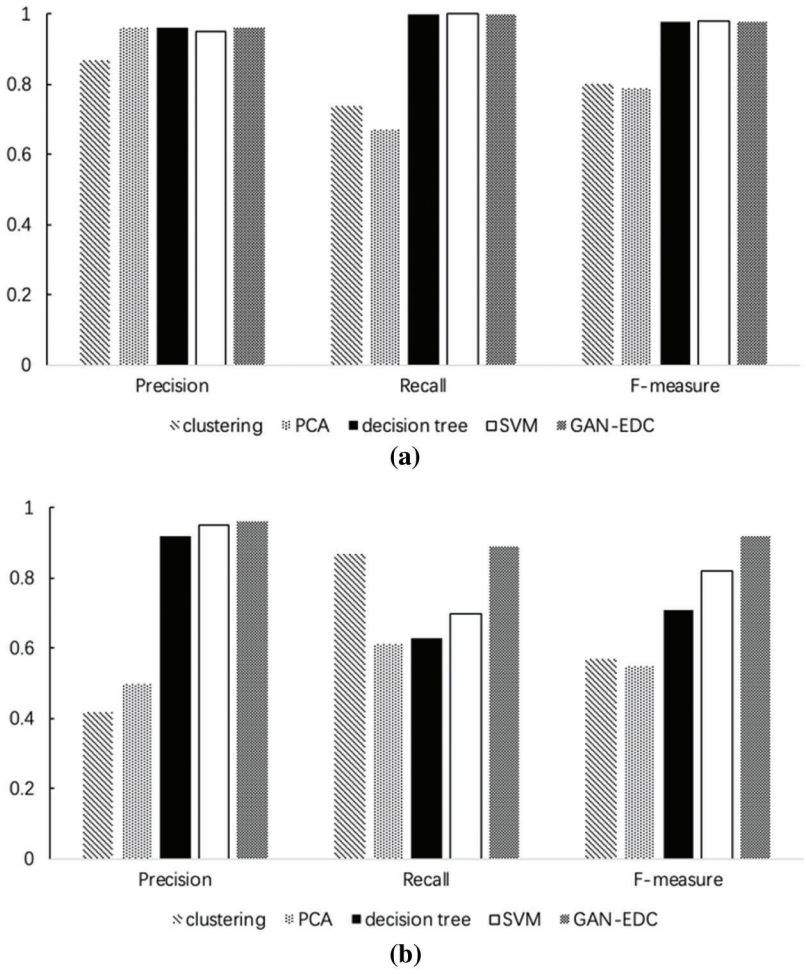

Figure 3: Evaluation on HDFS dataset and BGL dataset. (a) Accuracy on HDFS dataset. (b) Accuracy on BGL dataset

5.3 Efficiency Evaluation of Anomaly Detection

In Fig. 4, the efficiency of all these anomaly detection methods is evaluated on both HDFS and BGL datasets with varying log sizes. Because the clustering cannot handle large-scale datasets in an acceptable time, the running time results of clustering are not fully plotted. All methods can detect anomalies in a short time (except clustering). The time complexity of clustering is $O(n^2)$. The other detection methods scale linearly as the log size increases. Furthermore, the decision tree is the fastest detection method and slightly quicker than GAN-EDC, but GAN-EDC is higher than the decision tree in detection accuracy. Combine detection accuracy, GAN-EDC has achieved the best overall performance than other methods in two datasets. It is proved that GAN-EDC is effective for large-scale datasets.

5.4 The Effect of K Value

In this section, we evaluate the effect of k value on the generator by using the evaluation metric NLL_{oracle} [33]. In our case, we assume that the expert has learned an accurate model of the real data distribution

$p_{expert}(x)$. Only the negative log-likelihood is minimized, i.e., $-\mathbb{E}_{x\sim p} logq(x)$, the real data and the generated data are difficult to distinguish. Therefore, we use the LSTM network to capture the relationship between words in real log templates. We first initialize the parameters of an LSTM network following the normal distribution $N(0, 1)$ as the oracle (namely G_{oracle}) describing the real log distribution $G_{oracle}(x_t|x_1, \cdots, x_{t-1})$. Then, we use it to generate 10,000 sequences as the training dataset for the generative G. Finally, we evaluate G by using the oracle model. The evaluation function can be expressed as

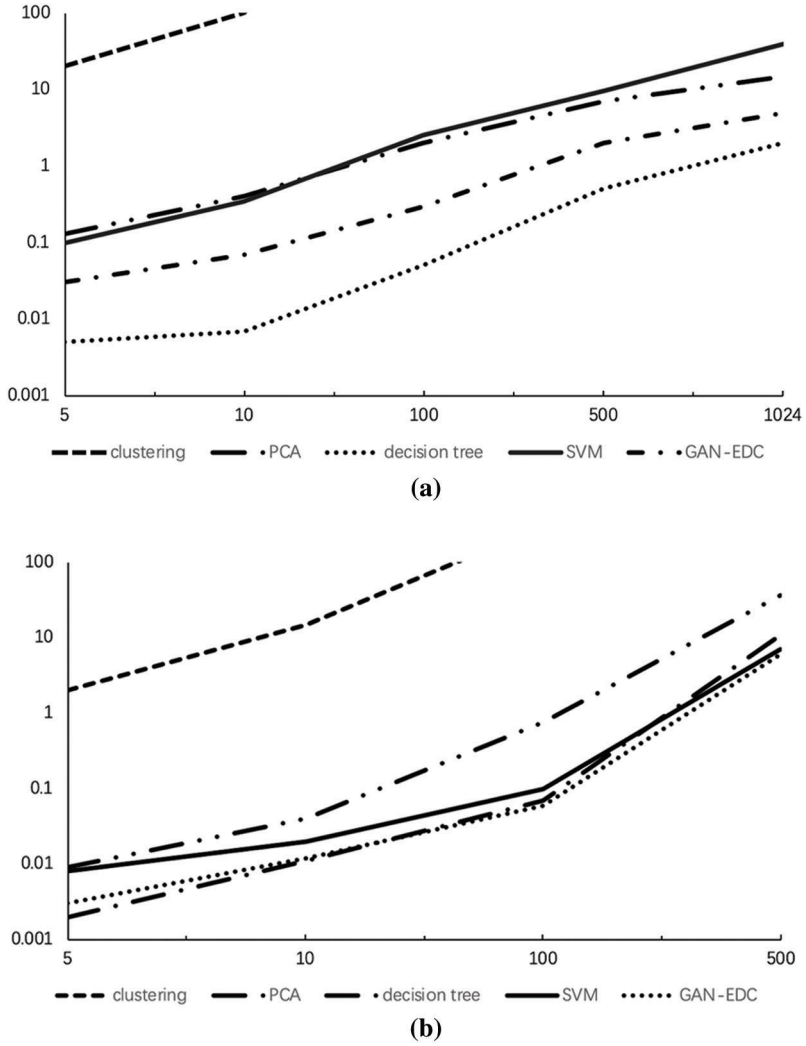

Figure 4: Evaluation running time with increasing log size. (a) HDFS dataset. (b) BGL dataset

$$NLL_{oracle} = -E_{Y_{1:T}\sim G_\theta}\left[\sum_{t=1}^{T} G_{oracle}(y_T|Y_{1:T-1})\right] \tag{8}$$

where $Y_{1:T-1}$ is the generated series before time T. G_{oracle} is the oracle model and G_θ is the generative model.

We use G_θ to generate 10,000 samples and calculate NLL_{oracle} with different k (k = 1, 3, 5) for each sample by G_{oracle} and their average score. We vary the values of k while using the same values for others. Fig. 5 shows the evaluation result.

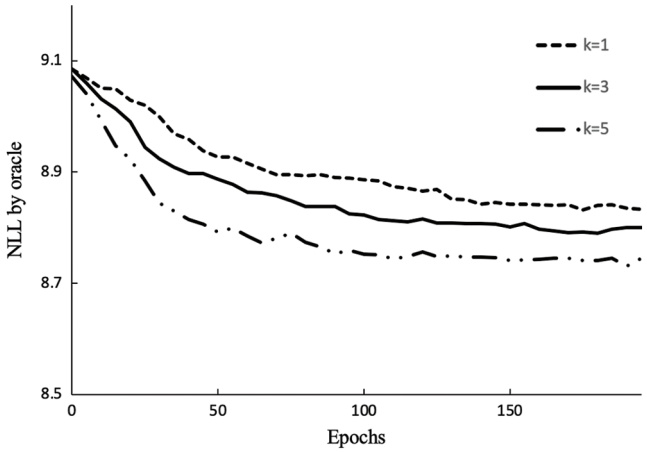

Figure 5: Negative log-likelihood convergence performance of GAN-EDC with different *k*-steps

We can see the impact of *k*. when $k = 3$, the baselines outperform other baselines significantly and pay equal attention to convergence speed. The baselines of $k = 1$ and $k = 3$ also converge to stable values, but the results are imperfect 8.83 and 8.79 respectively. When $k = 5$, the convergence speed of the model is the fast, and the result is the best 8.74. This proofs that the selection of the *k* has great effects on the convergence time of the model and the quality of generated data. Therefore, we set $k = 5$ in this paper.

6 Conclusion

In this paper, we propose a log anomaly detection method based on GAN. This method uses the Encoder-Decoder framework based on LSTM as the generator. The discriminator uses the CNN network to identify the difference between the generated log template and the real log template. In the phase of anomaly detection, the probability of anomaly is calculated by the Euclidean distance. Experiments on real data set show the effectiveness of GAN-EDC. Compared with the state of the art, GAN-EDC can not only detect log point anomalies with high accuracy but also outperform other existing log-based anomaly detection methods. Our future work considers using a reinforcement learning method to select *k* value instead of a human setting automatically. For example, we consider but are not limited to combine the Q-learning algorithm [40] or the Actor-Critic algorithm (A3C) [41].

Funding Statement: The work was supported by National Natural Science Foundation of China under grant NO.61672392 and NO.61373038, the National Key Research and Development Program of China under grant NO.2016YFC1202204.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Du, F. Li, G. Zheng and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. of the ACM Conf. on Computer and Communications Security*, Dallas, Texas, USA, pp. 1285–1298, 2017.

[2] R. Vaarandi and M. Pihelgas, "Logcluster - a data clustering and pattern mining algorithm for event logs, " in *Int. Conf. on Network & Service Management*, IEEE, Barcelona, Spain, pp. 1–7, 2015.

[3] F. T. Liu, K. M. Ting and Z. H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, no. 1, pp. 1–39, 2012.

[4] S. M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognition*, vol. 58, pp. 121–134, 2016.

[5] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning, " in *2017 IEEE Int. Conf. on Data Mining Workshops (ICDMW)*, IEEE, Los Alamitos, CA, USA, pp. 1058–1065, 2017.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley *et al.*, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

[7] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing*, Doha, Qatar, pp. 1746–1754, 2014.

[8] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng *et al.*, "Perceptual generative adversarial networks for small object detection," in *Proc. 2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, pp. 1951–1959, 2017.

[9] K. Ehsani, R. Mottaghi and A. Farhadi, "Segan: Segmenting and generating the invisible, " in *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Los Alamitos, CA, USA, 6144–6153, 2018.

[10] Y. Xue, T. Xu, H. Zhang, R. Long and X. Huang, "Segan: Adversarial network with multi-scale l1 loss for medical image segmentation," *Neuroinformatics*, vol. 16, no. 3–4, pp. 383–392, 2018.

[11] C. C. Hsu, H. T. Hwang, Y. C. Wu, Y. Tsao and H. M. Wang, "Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks," *18th Annual Conference of the Speech Communication Associatio (INTERSPEECH 2017)*, Stockholm, Sweden, pp. 3364–3368, 2017.

[12] D. Kwon, H. Kim, J. Kim, S. Suh, I. Kim *et al.*, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22, no. S1, pp. 949–961, 2019.

[13] A. N. Jahromi, S. Hashemi, A. Dehghantanha, K. K. R. Choo, H. Karimipour *et al.*, "An improved two-hidden-layer extreme learning machine for malware hunting," *Computers & Security*, vol. 89, pp. 101655, 2020.

[14] A. Javed, P. Burnap and O. Rana, "Prediction of drive by download attacks on twitter," *Information Processing & Management*, vol. 56, no. 3, pp. 1133–1145, 2019.

[15] R. Kaur and S. Singh, "A comparative analysis of structural graph metrics to identify anomalies in online social networks," *Computers & Electrical Engineering*, vol. 57, pp. 294–310, 2017.

[16] B. Mudassar, J. Ko and S. Mukhopadhyay, "An unsupervised anomalous event detection framework with class aware source separation, " in *2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, Calgary, AB, Canada, pp. 2671–2675, 2018.

[17] G. Enderlein, "Identification of outliers," *Biometrical Journal*, vol. 29, no. 2, pp. 198, 1987.

[18] R. Ren, X. Fu, J. Zhan and W. Zhou, "Logmaster: Mining event correlations in logs of large-scale cluster systems," in *Proc. of the IEEE Sym. on Reliable Distributed Systems*, Irvine, CA, USA, pp. 71–80, 2012.

[19] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS One*, vol. 11, no. 4, pp. 1–31, 2016.

[20] Y. Watanabe, H. Otsuka, M. Sonoda, S. Kikuchi and Y. Matsumoto, "Online failure prediction in cloud datacenters by real-time message pattern learning," in *4th IEEE Int. Conf. on Cloud Computing Technology and Science Proceedings*, Taipei, Taiwan, pp. 504–511, 2012.

[21] W. Xu, L. Huang, A. Fox, D. Patterson and M. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. of the ACM SIGOPS 22nd Sym. on Operating Systems Principles*, Taipei, Taiwan, pp. 37–46, 2010.

[22] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[23] H. Wu and S. Prasad, "Semi-supervised deep learning using pseudo labels for hyperspectral image classification," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1259–1270, 2018.

[24] E. Racah, C. Beckham, T. Maharaj, S. E. Kahou, Prabhat *et al.*, "Extreme weather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events," in *Proc. of the 31st Int. Conf. on Neural Information Processing Systems NIPS'17*, Long Beach, CA, USA, pp. 3405–3416, 2017.

[25] X. Jia, K. Li, X. Li and A. Zhang, "A novel semi-supervised deep learning framework for affective state recognition on EEG signals, " in *2014 IEEE Int. Conf. on Bioinformatics and Bioengineering*, IEEE, Boca Raton, FL, USA, pp. 30–37, 2014.

[26] G. Yuan, B. Li, Y. Yao and S. Zhang, "A deep learning enabled subspace spectral ensemble clustering approach for web anomaly detection, " in *2017 Int. Joint Conf. on Neural Networks (IJCNNI)*, IEEE, Anchorage, AK, USA, pp. 3896–3903, 2017.

[27] Z. Fengming, L. Shufang, G. Zhimin, W. Bo, T. Shiming *et al.*, "Anomaly detection in smart grid based on encoder-decoder framework with recurrent neural network," *Journal of China Universities of Posts and Telecommunications*, vol. 24, no. 6, pp. 67–73, 2017.

[28] A. Chawla, "Host based intrusion detection system with combined cnn/rnn model," in *Proc. ECML PKDD 2018*, Dublin, Ireland, pp. 149–158, 2019.

[29] W. Fang, F. Zhang, Y. Ding and J. Sheng, "A new sequential image prediction method based on LSTM and DCGAN," *CMC-Computers, Materials & Continua*, vol. 64, no. 1, pp. 217–231, 2020.

[30] C. Li, Y. Jiang and M. Cheslyar, "Embedding image through generated intermediate medium using deep convolutional generative adversarial network," *CMC-Computers, Materials & Continua*, vol. 56, no. 2, pp. 313–324, 2018.

[31] T. Li, S. Zhang and J. Xia, "Quantum generative adversarial network: A survey," *CMC-Computers, Materials & Continua*, vol. 64, no. 1, pp. 401–438, 2020.

[32] K. Fu, J. Peng, H. Zhang, X. Wang and F. Jiang, "Image super-resolution based on generative adversarial networks: A brief review," *CMC-Computers, Materials & Continua*, vol. 64, no. 3, pp. 1977–1997, 2020.

[33] L. Yu, W. Zhang, J. Wang and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proc. of the Thirty-First AAAI Conf. on Artificial Intelligence (AAAI'17)*, San Francisco, California, USA, pp. 2852–2858, 2017.

[34] J. Zhu, S. He, J. Liu, P. He, Q. Xie *et al.*, "Tools and benchmarks for automated log parsing, " in *2019 IEEE/ACM 41st Int. Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, IEEE/ACM, Montreal, QC, Canada, pp. 121–130, 2019.

[35] W. Xu, L. Huang, A. Fox, D. Patterson and M. Jordan, "Online system problem detection by mining patterns of console logs, " in *2009 Ninth IEEE Int. Conf. on Data Mining*, IEEE, Miami, Florida, USA, pp. 588–597, 2009.

[36] X. Duan, S. Ying, H. Cheng, W. Yuan and X. Yin, "OILog: An online incremental log keyword extraction approach based on MDP-LSTM neural network," *Information Systems*, vol. 95, pp. 101618, 2020.

[37] C. Bertero, M. Roy, C. Sauvanaud and G. Tredan, "Experience report: Log mining using natural language processing and application to anomaly detection, " in *2017 IEEE 28th Int. Sym. on Software Reliability Engineering (ISSRE)*, IEEE, Toulouse, France, pp. 351–360, 2017.

[38] A. Siddiqui, A. Fern, T. Dietterich, R. Wright, A. Theriault *et al.*, "Feedback-guided anomaly discovery via online optimization," in *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining KDD '18*, London, UK, pp. 2200–2209, 2018.

[39] P. He, J. Zhu, S. He, J. Li and M. R. Lyu, "An evaluation study on log parsing and its use in log mining, " in *2016 46th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN)*, IEEE, Toulouse, France, pp. 654–661, 2016.

[40] V. Mnih, K. Kavukvuoglu, D. Silver, A. Graves, I. Antonoglou *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[41] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. of the 33rd Int. Conf. on Machine Learning*, New York City, NY, USA, pp. 1928–1937, 2016.