SN Computer Science (2021) 2:391
https://doi.org/10.1007/s42979-021-00772-9

ORIGINAL RESEARCH

Two Class Pruned Log Message Anomaly Detection

Amir Farzad[1] · T. Aaron Gulliver[1]

Received: 14 September 2020 / Accepted: 11 July 2021

Abstract

Log messages are widely used in cloud servers and other systems. Millions of logs are generated each day which makes them important for anomaly detection. However, they are complex unstructured text messages which makes this task difficult. In this paper, a hybrid log message anomaly detection technique is proposed which employs pruning of positive and negative logs. Reliable positive log messages are first selected using a Gaussian mixture model algorithm. Then reliable negative logs are selected using the K-means, Gaussian mixture model and Dirichlet process Gaussian mixture model methods iteratively. It is shown that the precision for positive and negative logs with pruning is high. Anomaly detection is done using a deep learning long short-term memory network. The proposed model is evaluated using the well-known BGL, Openstack, and Thunderbird data sets. The results obtained indicate that the proposed model performs better than several well-known algorithms.

Keywords Anomaly detection · Deep learning · Log messages · Hybrid learning

Introduction

Companies and customers expect 24/7 connectivity to their cloud and software systems and loss of access can have serious consequences. Thus, significant investments have been made to preserve the quality and availability of these services. This is achieved by generating log messages which indicate the status of the system. Logging is the process of storing records for audit or security [46]. Log messages are unstructured text data that consist of time stamps, verbosity, and raw content concerning the system status. Logs are unstructured because developers typically use free text to record events for ease and flexibility [46]. Thus, the structure of these logs can vary considerably, making it hard to identify abnormalities [42]. Log messages are used for several purposes including anomaly detection [17] and performance monitoring [43]. Most techniques employ rules to identify anomalies in logs but this requires specialized knowledge of the area [41]. Some only consider one feature such as verbosity which limits the ability to detect abnormalities. Anomaly detection can be carried out manually, but for large systems this is not practical due to the amount of data and the complexity [25]. As a result, automated log analysis methods are required to identify anomalies.

Deep learning (DL) is a subclass of machine learning (ML) which employs a network with several layers. DL can identify similarities in data [14] which makes it desirable for big data applications. DL has been shown to provide excellent results for speech recognition, image processing and text classification [22, 44]. DL methods have good recognition capabilities for large amounts of data and are better than other ML methods for feature representation [5]. ML methods can be discriminative, generative, or hybrid. Discriminative methods are typically used for supervised classification while generative methods are employed for unsupervised classification. Hybrid methods combine generative and discriminative methods. One of the main issues in anomaly detection is handling unlabeled data. Millions of log messages are produced daily in cloud and other systems so it is typically not possible to label even a small portion of the data. Thus, unsupervised approaches should be considered to deal with this unlabeled data.

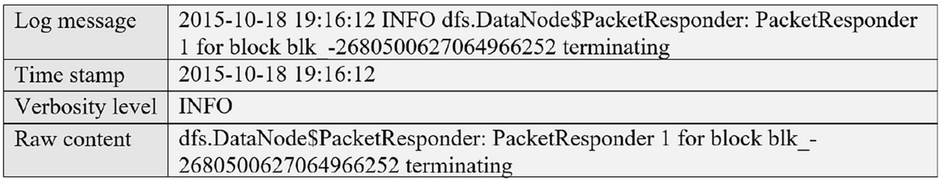
ML algorithms have been used to develop a range of anomaly detection methods. Elliptical envelope (EEnvelope)

✉ Amir Farzad
amirfarzad@uvic.ca

T. Aaron Gulliver
agullive@ece.uvic.ca

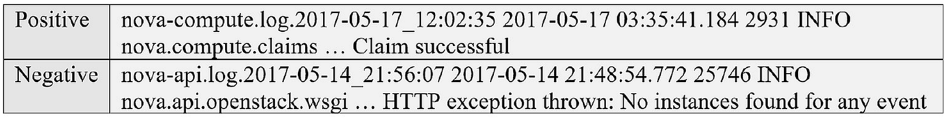
[1] Department of Electrical and Computer Engineering, University of Victoria, PO Box 1700, STN CSC, Victoria, BC V8W 2Y2, Canada

Published online: 24 July 2021



Log message	2015-10-18 19:16:12 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-2680500627064966252 terminating
Time stamp	2015-10-18 19:16:12
Verbosity level	INFO
Raw content	dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-2680500627064966252 terminating

Fig. 1 An example of a log message consisting of time stamp, verbosity level and raw content



Positive	nova-compute.log.2017-05-17_12:02:35 2017-05-17 03:35:41.184 2931 INFO nova.compute.claims … Claim successful
Negative	nova-api.log.2017-05-14_21:56:07 2017-05-14 21:48:54.772 25746 INFO nova.api.openstack.wsgi … HTTP exception thrown: No instances found for any event

Fig. 2 Examples of positive and negative log messages from Openstack

creates an elliptical area around the data mass center which is used in redshift estimation to detect anomalies [20]. Local outlier factor (LOF) uses local data deviation to detect network flow anomalies to reduce the risk of internet attacks [31]. Support vector machine (SVM) and one-class support vector machine (OC-SVM) have been employed to detect unknown computer activity [30] and anomalies in networks [28]. A convolutional neural network (CNN) combined with a graph convolutional network was used for abnormal breast detection in mammograms [45] and COVID-19 detection in CT images [38]. A decision tree model was considered in [32] to detect faults using log messages. An improved supervised K-nearest neighbors (IKNN) method was employed in [36] to detect anomalies in log messages. However, using supervised methods is not always possible because of the lack of labeled data. A method to detect anomalies in log messages using isolation forest with two autoencoder networks for feature extraction was presented in [11]. However, a detection threshold is required for each data set which is difficult to determine in practice.

Deeplog [7] uses a long short-term memory (LSTM) network [19] for anomaly detection. First, each log is mapped to its print statement (in the source code) using a log parser. Thus, each log is represented by a number (feature) and a session of logs is parsed to a sequence of numbers. Then, an LSTM network is trained as a multi-class classifier using a window with normal sequences (corresponding to normal system operation) [7]. The trained LSTM network is then used to predict the probabilities of numbers occurring at a given time step. If the actual number is unlikely to occur based on the LSTM prediction, then it is considered to be an anomaly. While this approach may be effective, it can be difficult to obtain labeled logs for normal system operation. The proposed model uses both normal (positive) and abnormal (negative) log messages for training and unsupervised algorithms are used to prune positive and negative logs. Once the reliable positive and negative logs are selected, an LSTM network is used for anomaly detection. Further, while log parsing is used by most ML/DL log message anomaly detection models such as Deeplog, only simple text log preprocessing is used in the proposed model.

Since the log messages are unstructured, they are usually parsed before being input to an ML model. There are several different log parsing methods including LogSig and IPLoM [18]. An example of a log message is given in Fig. 1. The purpose of log parsing is to differentiate between constant and variable parts of the message so that the constant elements are mapped to a list of log events. However, systems are continually changing, so it is difficult to develop effective automated log parsing methods [18]. Figure 2 shows examples of positive and negative logs from Openstack[1]. Although the verbosity level for both logs is INFO, the positive log is a declaration which may show a claim was successful on a node and the negative log may show the system has been terminated due to failing to start. This indicates that considering just one log message component may be insufficient for detecting anomalies.

K-means is a well-known clustering method which has been widely used in tasks such as detecting network intrusions [34]. Gaussian mixture model (GMM) is a clustering method that assumes the data was created from a combination of Gaussian distributions. It has been used to solve problems such as detecting anomalies in flight operation data [24]. The Dirichlet process Gaussian mixture model with variational inference (BGM) is a Bayesian mixture model (an extension of finite mixture models), which has been used for tasks such as anomaly detection in hyperspectral data [35]. An LSTM network [19] is a recurrent neural network (RNN) which uses a cell to preserve sequence information and recall long-term dependencies. LSTM networks have been used for various tasks such as text classification [37] and texture classification in images [4]. An LSTM network is suitable for sequential data such as log messages [10] and is effective with big data [8]. Another advantage of an LSTM network is robustness so it can be used with complex data.

[1] https://github.com/logpai/loghub/tree/master/OpenStack.

However, training an LSTM network can be time-consuming [16].

In this paper, a hybrid model is proposed which employs unsupervised K-means, GMM and BGM methods for data pruning and a supervised LSTM network for anomaly detection using reliable data. First, reliable positive logs are obtained using a GMM. Although GMMs are widely used for anomaly detection, we use a GMM to prune only positive logs in the first step. Then, reliable negative logs are obtained using pruning with the unsupervised K-means, GMM and BGM methods. Finally, a portion of the reliable positive and negative logs are used for anomaly detection using an LSTM network. The amount of data used for LSTM training is very small even though convergence with deep networks typically requires a significant amount of training data. The proposed model is evaluated using the accuracy, precision, recall and F-measure criteria [12], and three log message data sets, namely BlueGene/L (BGL)[2], Openstack and Thunderbird[3], are considered. The parameters of the proposed model are the same for all data sets to illustrate the robustness of this approach.

The main contributions of this paper are as follows.

1. An unsupervised algorithm is presented which uses a GMM method to select reliable positive logs.
2. An unsupervised algorithm is presented which employs K-means, GMM, and BGM methods iteratively to select reliable negative logs.
3. An LSTM network is used with the pruned logs for anomaly detection.

The rest of this paper is organized as follows. In the next section, the K-means, GMM, BGM, and LSTM architectures are presented and the proposed model is described. Experimental results for the three data sets are given in the third section along with a discussion of the model performance. Finally, the fourth section provides some concluding remarks.

System Model

In this section, the K-means, GMM, BGM, and LSTM architectures are given and the proposed model is described.

K-Means

K-means is an iterative clustering method. Given k classes, each cluster has a center which is the average of the samples in the cluster. The set of clusters is $S = \{S_1, S_2, \dots, S_k\}$ and a sample is assigned to the cluster whose center it is closest to. First, the cluster centers are initialized randomly. Next, the Euclidean distances between each sample and the cluster centers, c_i, are calculated. Then, each sample is reassigned to the closest cluster and new centers are calculated for each cluster. This process continues until the clusters do not change or the maximum number of iterations is attained. This corresponds to minimizing the objective function given by

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - c_i||^2, \tag{1}$$

where k is the number of clusters, c_i is the center of cluster S_i, x is a data sample, and $||x - c_i||^2$ is the Euclidean distance from x to c_i.

Gaussian Mixture Model (GMM)

Gaussian mixture model (GMM) is a clustering method. It is assumed that each cluster consists of data with a normal (Gaussian) distribution. The goal is to estimate the distribution parameters of each cluster and determine the labels for the samples, i.e., which cluster each sample belongs to. The expectation maximization (EM) algorithm [6] is used to obtain estimates for these parameters. The GMM probability density function (PDF) of sample x_j is given by

$$p(x_j|\theta) = \sum_{i=1}^{m} \pi_i \mathcal{N}(x_j|\mu_i, \Sigma_i), \tag{2}$$

where π_i, μ_i and Σ_i are the weight, mean, and covariance matrices of the ith distribution, respectively, m is the number of distributions (clusters), and $\theta = \{\mu_i, \Sigma_i, \pi_i\}$ is the parameter set of the mixture model. For d features, the Gaussian distribution of x_j is

$$\mathcal{N}(x_j|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma_i|}} \exp\left(-\frac{1}{2}(x_j - \mu_i)^T \Sigma_i^{-1}(x_j - \mu_i)\right). \tag{3}$$

The EM algorithm iterates two steps, expectation (E-step) and maximization (M-step). First, the model parameters θ are randomly initialized and the expectation and maximization steps are performed. The E-step is given by

$$\gamma_i^t(x_j) = \frac{\pi_i^t \mathcal{N}(x_j|\mu_i^t, \Sigma_i^t)}{\sum_{i=1}^{m} \pi_i^t \mathcal{N}(x_j|\mu_i^t, \Sigma_i^t)}, \tag{4}$$

where t is the iteration number. Then the parameters are estimated in the M-step as

[2] https://github.com/logpai/loghub/tree/master/BGL.

[3] https://github.com/logpai/loghub/tree/master/Thunderbird.

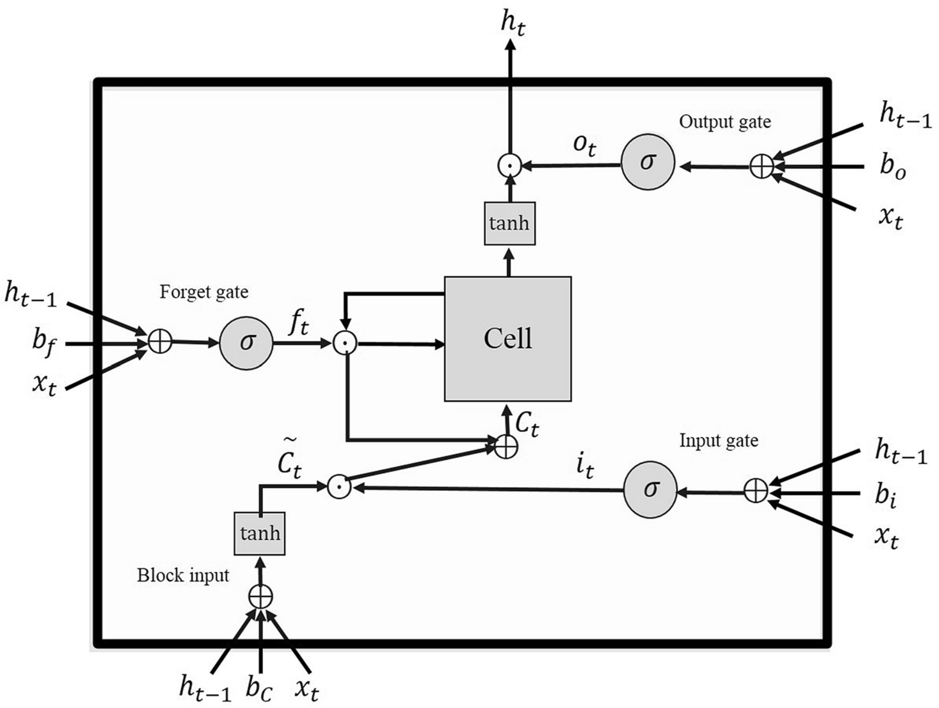


Fig. 3 A block of an LSTM network with input, input, output and forget gates [19]

$$\mu_i^{t+1} = \frac{\sum_{j=1}^{N} \gamma_i^t(x_j)x_j}{\sum_{j=1}^{N} \gamma_i^t(x_j)}, \tag{5}$$

$$\Sigma_i^{t+1} = \frac{\sum_{j=1}^{N} \gamma_i^t(x_j)(x_j - \mu_i^{t+1})(x_j - \mu_i^{t+1})^T}{\sum_{j=1}^{N} \gamma_i^t(x_j)}, \tag{6}$$

$$\pi_i^{t+1} = \frac{1}{N} \sum_{j=1}^{N} \gamma_i^t(x_j). \tag{7}$$

where N is the number of samples. These steps are repeated until the criteria are satisfied or a maximum number of iterations is reached.

Dirichlet Process Gaussian Mixture Model (BGM)

The Dirichlet process Gaussian mixture model (BGM) is a non-parametric Bayesian mixture model that is an extension of finite mixture models. The number of clusters (classes) does not need to be explicitly predefined because it is a non-parametric model. BGM uses the Dirichlet process (DP) which is a generalized form of a Dirichlet distribution [13]. A DP is composed of a base distribution G_0 and a positive concentration scaler α. Since this model is not a finite mixture model, variational inference is employed [3]. The model parameters are

$$\pi_i \sim DP(G_0, \alpha), \tag{8}$$

$$\Sigma_i \sim \mathcal{W}^{-1}(v, s), \tag{9}$$

$$\mu_i \sim \mathcal{N}(\mu_0, \Sigma_i), \tag{10}$$

where μ_0 and Σ_i are the mean and covariance of the Gaussian distribution, s is the scale matrix, and v is the number of degrees of freedom for the Inverse-Wishart distribution [39].

LSTM Architecture

An LSTM is a recurrent neural network [19] which has been used to solve sequential data problems [15]. Cells are used to store information and they are connected recurrently. The use of cells solves the vanishing gradient problem. Each LSTM block includes input, forget, and output gates. These gates can store information longer than feed-forward neural networks which improves performance [15]. A block of an LSTM network is shown in Fig. 3. The cell input at time t is x_t and the input, forget, and output gate outputs are

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{11}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{12}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{13}$$

respectively, where h_{t-1} is the previous block output, b is the bias vector, W and U are weight matrices, and σ is the activation function (usually the sigmoid function). The block input at time t is given by

$$\hat{C}_t = \tanh(W_C x_t + U_C h_{t-1} + b_C), \tag{14}$$

where b_C is the bias vector, W_C and U_C are the weight matrices, and tanh denotes the hyperbolic tangent activation function. The cell state at time t is

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t, \tag{15}$$

where $\odot$ denotes point-wise multiplication. The block output at time t is

$$h_t = o_t \odot \tanh(C_t). \tag{16}$$

Proposed Model

The proposed model architecture has three steps. First, positive logs are pruned using an unsupervised GMM method. Second, negative logs are pruned through multiple rounds of the unsupervised GMM, BGM, and K-means methods. Finally, anomalies are detected using an LSTM network with the selected (reliable) positive and negative logs.

First, simple text pre-processing including changing letters to lowercase, removing hyphens and tokenization are applied to the data set D. Next, the sentences are padded to 40 tokens, and sentences including less than five tokens are removed. Then, the number of appearances of each token in the data set is computed and the tokens are ordered from most frequent to least frequent. Each token is given an index starting from zero and the indices are used as the data set features. Next, the features are normalized using a min-max scaler so all values are between 0 and 1 and the entries in the data set are shuffled. Then D is divided into two sets, t_1 with 2% of the data for training and r_1 with the remaining 98% of the data. The set t_1 is small to keep the computational complexity low and have more data for the rest of the algorithm. The proportion of negative and positive logs in these sets is the same as in D.

Select Reliable Positive Logs

A GMM is used to prune the positive logs. It is trained with t_1 and tested with r_1. The negative predicted logs (predicted output $y = 0$) and positive predicted logs (predicted output $y = 1$) are counted and labeled c_0 and c_1, respectively. If the number of logs predicted as positive is less than the number predicted as negative, then c_0 and c_1 are swapped. This is because it is known that the number of anomalies (negative logs) is much less than the number of positive logs (around 10%). The variance is given by

$$var = \frac{\sum_{i=1}^{F} (x_i - \hat{x})^2}{F}, \tag{17}$$

where x_i is the ith feature, $\hat{x}$ is the average of the features and F is the total number of features in the data set. Let $a = \frac{c_1}{c_0}$ and the variances of the negative and positive predicted logs be z_{var} and o_{var}, respectively. If $a > 3$ and $o_{var} \times c < z_{var}$ (c is a constant), then the positive and negative predicted logs are added to the sets o_0 (reliable positive logs) and z_0 (rest of the data), respectively. The threshold for a was chosen considering that the majority of the logs are positive. A high value of c increases the probability of getting only positive logs but if it is too high the algorithm criterion may not be satisfied. It was set to $c = 1.6$ for all data sets based on the experimental results obtained.

The variance measures the spread of a data set. If the model predicted most of the positive logs correctly (small number of false positives), then the variance of the positive logs should be lower than that of the negative logs. A high variance may indicate that there is a mix of positive and negative logs whereas a small variance indicates that there are mostly positive logs predicted correctly. If the criteria are met, the results are kept, otherwise the process is repeated.

Select Reliable Negative Logs

The GMM, K-means and BGM methods are now used to select negative logs. These models were chosen because they are efficient unsupervised models for text data [1]. There are n rounds and in each round, GMM, K-means and BGM are run m times. In the first round, the models are trained with z_0 from the previous step. In subsequent rounds, the results from the previous round z are used for this purpose. The entropy of sample x_j is given by

$$H = -\sum_{i=1}^{d} \frac{n_i}{M} \ln(\frac{n_i}{M}), \tag{18}$$

where n_i is the ith feature of the sample, d is the length of the sample and M is the sum of the sample features.

For a model run, denote the average entropy of the logs predicted as negative and positive as sh_0 and sh_1, respectively. If $sh_0 < sh_1$, then the predicted negative logs are appended to z_1 and the predicted positive logs are appended to o_1. This is because small features appear frequently in positive logs so they are more uniform and thus have a higher entropy. At the end of a round, z_1 is assigned to z for use in the next round. The logs in z are counted and ordered from most frequent to least frequent and the repetitions discarded.

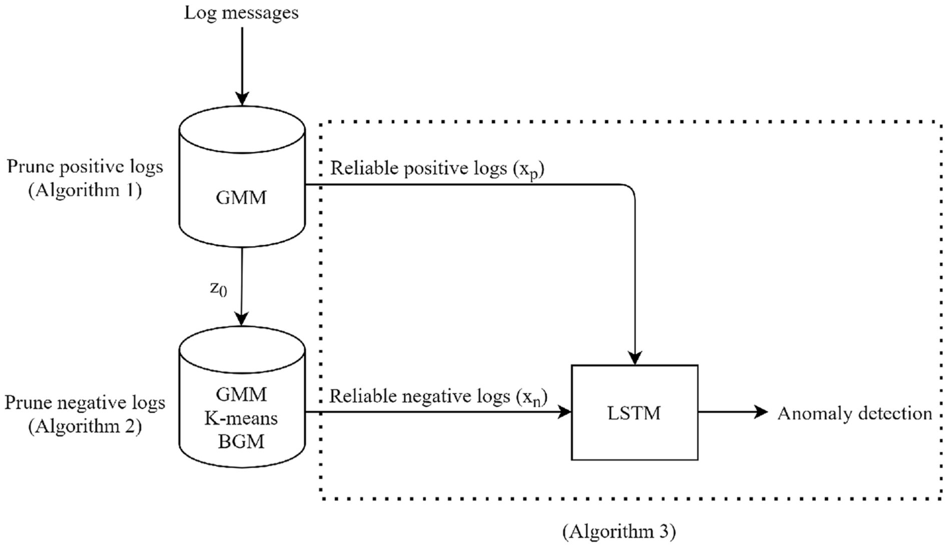

Log messages
Prune positive logs (Algorithm 1)
GMM
Reliable positive logs (x_p)
z_0
Prune negative logs (Algorithm 2)
GMM
K-means
BGM
Reliable negative logs (x_n)
LSTM
Anomaly detection
(Algorithm 3)

Fig. 4 The proposed model architecture

This is done so that each log appears at most once in z and the logs that appear more often are used earlier so the models in the next round can predict better. The logs in z_1 are discarded at the start of each round.

In each round, the prediction of positive and negative logs is done using z. Thus, the number of positive logs is reduced and only reliable negative logs are kept. The final z_1 contains the predicted negative logs from the last round and these are used in the next step.

Anomaly Detection

In this step, an LSTM network is used with the reliable negative (z_1) and positive (o_0) logs from the previous two steps for anomaly detection. Each log in z_1 and o_1 is counted and ordered from most frequent to least frequent and the repetitions are discarded. The first L logs in z_1 are selected and assigned to z_2 (most reliable predicted negative logs), and the remaining logs are assigned to o_2. Logs which appear in o_1 but not in z_1 are placed in o_3. The reliable positive logs o_0 obtained in the first step are shuffled, and 10% are randomly assigned to o_4 and the remainder to o_5. The logs in z_2 are repeated four times and assigned to x_n. A portion of o_4 which is the same size as the number of elements in x_n is randomly chosen and assigned to x_p. Thus, the reliable negative logs are oversampled so the number of reliable positive logs and negative logs is the same. This is because LSTM networks work better with balanced data and should be trained with a sufficient number of positive and negative logs. The logs in x_n and x_p are labeled with $y = 0$ and $y = 1$ indicating negative and positive logs, respectively, and x_n and x_p are assigned to t_2. The remaining logs in o_4 are assigned to o_6, and o_2, o_3, o_5 and o_6 are assigned to t_3. The data set was initially scaled so all values are between 0 and 1, but this is reversed for t_2 and t_3 to provide training and testing sets, respectively, for the LSTM network.

The LSTM network is trained with 90% of t_2, validated with the remaining 10% of t_2, and tested with t_3. The parameters used are $k = 20$, $n = m = 5$, and $L = 10000$ for the BGL and Thunderbird data sets and $L = 3000$ for the Openstack data set. A different value of L is used because the Openstack data set is much smaller than the BGL and Thunderbird data sets. For training the BGL and Thunderbird data sets, an LSTM network with three hidden layers of size 256, batch size 128 and a maximum of 10 training epochs is used. To prevent overfitting, dropout with probability 0.5 and early stopping are used. The softmax activation function is applied in the last dense layer. The cross-entropy loss function and Adam optimizer are used for training. The Adam optimizer is used because it has been shown to provide good performance and fast convergence in DL algorithms [33]. For the Openstack data set, an LSTM network with a single hidden layer of size 512 and embedding dimension of size 512 is used. A single-layer network is used for this data set because it is smaller than the other data sets. The rest of the architecture for the Openstack data set is the same as above. All network parameters were chosen based on the experimental results obtained. An LSTM network is used for anomaly detection because it has been shown to provide good results in classifying sequential data [15]. However, other DL discriminative networks such as a CNN can be employed. The proposed model algorithms are given in Algorithms 1–3 and shown in Fig. 4. The data preparation for Algorithm 3 is shown in Fig. 5.

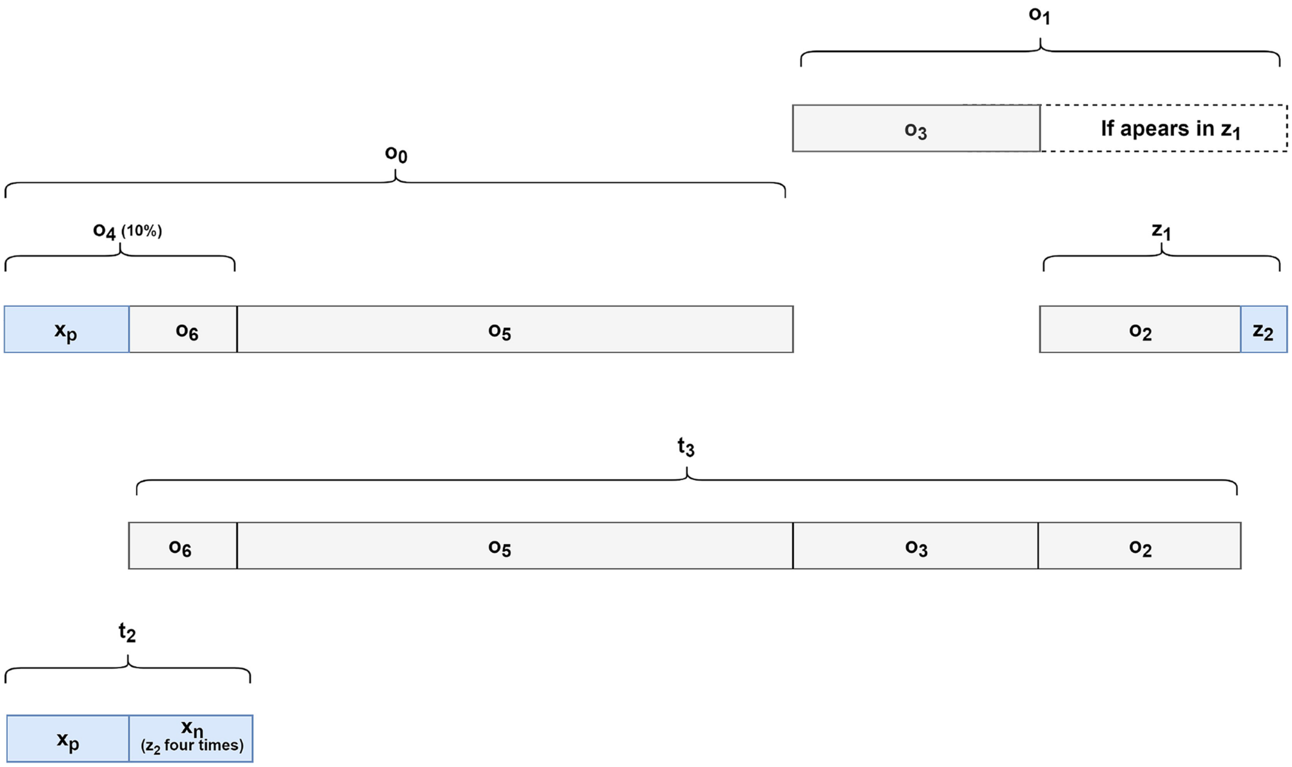


Fig. 5 The data preparation for Algorithm 3

```
Algorithm 1 Prune Positive Logs
Input: Data set $D$ scaled between $[0, 1]$, training set $t_1 = 2\%$ of $D$, remaining set
    $r_1 = 98\%$ of $D$
for $i \leftarrow 1$ to $k$ do
    Train GMM with $t_1$ and test with $r_1$
    Count negative predicted logs $(y = 0)$ $(c_0)$
    Count positive predicted logs $(y = 1)$ $(c_1)$
    if $c_1 < c_0$ then swap $c_0$ and $c_1$
    end if
    $a = \frac{c_1}{c_0}$
    Compute variance of negative predicted logs $(z_{var})$
    Compute variance of positive predicted logs $(o_{var})$
    if $(a > 3)$ and $(o_{var} \times c < z_{var})$ then
        $z_0 \leftarrow$ Selected negative predicted logs
        $o_0 \leftarrow$ Selected positive predicted logs
        break for loop
    end if
end for
```

Algorithm 2 Prune Negative Logs

```
Model ← [GMM, K-means, BGM]
for $round$ ← 1 to $n$ do
    $z_1$ ← $Empty$
    for each Model do
        for $j$ ← 1 to $m$ do
            if $round = 1$ then
                Train on $z_0$. Get negative and positive predicted logs
            else
                Train on $z$. Get negative and positive predicted logs
            end if
            Compute average entropy of negative predicted logs ($sh_0$)
            Compute average entropy of positive predicted logs ($sh_1$)
            if $sh_0 < sh_1$ then
                Append negative predicted logs to $z_1$
                Append positive predicted logs to $o_1$
            end if
        end for
    end for
    $z = z_1$
    Count each log in $z$, then order from most frequent to least frequent and delete the repetitions
end for
```

Algorithm 3 Anomaly Detection

```
Count each log in $o_1$, then order from most frequent to least frequent and delete the repetitions
Count each log in $z_1$, then order from most frequent to least frequent and delete the repetitions
$z_2$ ← first $L$ logs of $z_1$
$o_2$ ← rest of $z_1$
$o_3$ ← $o_1$ minus any logs that also appear in $z_1$
$o_4$ ← 10% of $o_0$ chosen randomly
$o_5$ ← rest of $o_0$
$x_n$ ← $z_2$ repeated four times, label as $y = 0$ (oversampling)
$x_p$ ← a random portion of $o_4$ so the length is the same as $x_n$, label as $y = 1$
$t_2$ ← $x_n$ and $x_p$
$o_6$ ← rest of $o_4$
$t_3$ ← $o_2$, $o_3$, $o_5$, and $o_6$
Reverse the scaling of $t_2$ and $t_3$
Train the LSTM network using $t_2$ and test using $t_3$
```

Results

In this section, the proposed model is evaluated using the BGL, Openstack and Thunderbird data sets. Four performance criteria are considered, namely accuracy, precision, recall and F-measure [12]. The percentage of data correctly predicted is called the accuracy and is given by

$$A = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}, \tag{19}$$

where T_p is the number of positive samples predicted by the model to be positive, F_p is the number of negative samples predicted to be positive, T_n is the number of negative samples predicted to be negative, and F_n is the number of positive samples predicted to be negative. Then, the precision is

Table 1 The proposed model testing accuracy, precision, recall, F-measure, and average time with (a) GMM for positive log pruning and (b) BGM for positive log pruning

Data set	Testing accuracy	Label	Precision	Recall	F-measure	Time (s)
(a)						
BGL	99.3%-(**99.5%**)-99.6%	0	93.2%-(**95.6%**)-97.7%	97.1%-(**97.8%**)-98.7%	95.6%-(**96.7%**)-97.6%	3725
		1	99.8%-(99.8%)-99.9%	99.4%-(99.6%)-99.8%	99.6%-(99.7%)-99.8%	
Openstack	99.8%-(**99.9%**)-100%	0	99.3%-(**99.9%**)-100%	97.9%-(**99.7%**)-100%	99.0%-(**99.8%**)-100%	177
		1	99.7%-(99.9%)-100%	99.9%-(99.9%)-100%	99.9%-(99.9%)-100%	
Thunderbird	99.6%-(**99.8%**)-99.9%	0	97.1%-(**98.9%**)-99.9%	99.6%-(**99.6%**)-99.7%	98.3%-(**99.3%**)-99.8%	3550
		1	99.9%-(99.9%)-99.9%	99.5%-(99.8%)-99.9%	99.7%-(99.9%)-99.9%	
(b)						
BGL	99.4%-(**99.5%**)-99.7%	0	94.0%-(**95.4%**)-97.4%	97.2%-(**98.5%**)-99.4%	96.0%-(**96.9%**)-98.2%	3649
		1	99.8%-(99.9%)-99.9%	99.5%-(99.6%)-99.8%	99.7%-(99.7%)-99.8%	
Openstack	99.6%-(**99.9%**)-100%	0	96.7%-(**99.7%**)-100%	99.6%-(**99.9%**)-100%	98.3%-(**99.8%**)-100%	134
		1	99.9%-(99.9%)-100%	99.6%-(99.9%)-100%	99.8%-(99.9%)-100%	
Thunderbird	99.7%-(**99.8%**)-99.9%	0	97.5%-(**99.0%**)-99.9%	99.6%-(**99.6%**)-99.7%	98.6%-(**99.3%**)-99.8%	3136
		1	99.9%-(99.9%)-99.9%	99.7%-(99.8%)-99.9%	99.8%-(99.9%)-99.9%	

The minimum, maximum and average (in parenthesis) values are given for 10 runs with the BGL, Openstack and Thunderbird data sets. Positive labels are denoted by 1 and negative labels by 0

Table 2 The precision, recall and F-measure for negative logs for (a) BGL, (b) Openstack and (c) Thunderbird data sets with the Auto-LSTM, IKNN, nLSALog and Deeplog algorithms

Algorithm	Precision (%)	Recall (%)	F-measure (%)
(a)			
Auto-LSTM	98.0	91.3	94.5
IKNN	92.0	91.0	92.0
nLSALog	82.5	94.7	88.2
(b)			
Auto-LSTM	99.4	92.8	96.0
Deeplog	94.0	99.0	97.0
(c)			
Auto-LSTM	98.4	99.8	99.1
IKNN	96.0	96.0	96.0

$$P = \frac{T_p}{T_p + F_p}, \tag{20}$$

the recall is

$$R = \frac{T_p}{T_p + F_n}, \tag{21}$$

and the F-measure is

$$F = \frac{2 \times P \times R}{P + R}. \tag{22}$$

All experiments were run on the Compute Canada Graham cluster with 32 CPU cores, two P100 GPUs and 124 GB of memory. The algorithms were implemented using Python, Keras[4] and Scikit-learn[5].

The hyperparameters of the proposed model were not tuned so the default values were used in all experiments. Each experiment was repeated 10 times and the minimum, maximum and average testing accuracy, precision, recall, F-measure and computation time were obtained. Table 1a gives the proposed model results for the BGL, Openstack and Thunderbird data sets using GMM for positive pruning. For comparison, the proposed model results using BGM for positive pruning are given in Table 1b with the order for negative pruning changed to K-means, GMM, and BGM. The results for negative logs with the Auto-LSTM [10], IKNN, nLSALog [40] and Deeplog algorithms for the (a) BGL, (b) Openstack, and (c) Thunderbird data sets are given in Table 2. Table 3 gives the average testing accuracy, precision, recall, F-measure and computation time with the BGM, EEnvelope, GMM, K-means, LOF and OC-SVM methods for the (a) BGL, (b) Openstack, and (c) Thunderbird data sets. Table 4 presents the positive log pruning results for the BGL, Openstack and Thunderbird data sets with (a) GMM and (b) BGM. Tables 5, 6, 7 give the negative log pruning results for the BGL, Openstack and Thunderbird data sets, respectively, with (a) GMM, (b) K-means, and (c) BGM for $n = 5$ rounds.

[4] https://github.com/keras-team/keras.

[5] https://github.com/scikit-learn/scikit-learn.

Table 3 Average testing accuracy, precision, recall, F-measure, and time for (a) BGL, (b) Openstack and (c) Thunderbird data sets with the BGM, EEnvelope, GMM, K-means, LOF and OC-SVM methods using 10-fold cross-validation

Algorithm	Testing accuracy	Label	Precision	Recall	F-measure	Time (s)
(a)						
BGM	50.3%	0	37.8%	50.0%	43.0%	2632
		1	63.9%	50.3%	52.2%	
EEnvelope	86.2%	0	17.6%	23.9%	20.3%	5287
		1	93.8%	91.1%	92.4%	
GMM	50.3%	0	38.2%	50.0%	43.3%	1894
		1	63.9%	50.4%	52.2%	
K-means	50.0%	0	6.6%	50.0%	11.7%	7759
		1	93.4%	50.0%	64.9%	
LOF	83.6%	0	7.1%	10.3%	8.4%	605
		1	92.6%	89.4%	91.0%	
OC-SVM	84.3%	0	8.5%	11.4%	9.7%	28469
		1	92.8%	90.1%	91.4%	
(b)						
BGM	50.3%	0	17.0%	50.0%	21.9%	90
		1	82.9%	50.3%	59.8%	
EEnvelope	88.8%	0	53.4%	44.9%	48.8%	244
		1	92.7%	94.7%	93.7%	
GMM	48.7%	0	35.2%	60.0%	38.9%	77
		1	68.2%	47.1%	53.0%	
K-means	30.0%	0	30.0%	30.0%	30.0%	208
		1	30.0%	30.0%	30.0%	
LOF	80.3%	0	14.1%	13.1%	13.6%	985
		1	88.4%	89.3%	88.9%	
OC-SVM	38.5%	0	0.3%	1.3%	0.5%	13502
		1	76.6%	43.5%	55.5%	
(c)						
BGM	57.9%	0	25.9%	60.0%	35.2%	1901
		1	83.6%	57.6%	63.4%	
EEnvelope	82.1%	0	43.8%	26.2%	32.8%	3872
		1	86.3%	93.3%	89.6%	
GMM	66.2%	0	27.1%	70.0%	37.2%	760
		1	92.8%	65.8%	71.5%	

Table 3 (continued)

Algorithm	Testing accuracy	Label	Precision	Recall	F-measure	Time (s)
K-means	50.0%	0	5.9%	50.0%	10.6%	4787
		1	94.1%	50.0%	58.3%	
LOF	75.4%	0	17.5%	10.3%	12.9%	1141
		1	82.2%	89.5%	85.7%	
OC-SVM	40.6%	0	8.5%	23.7%	12.5%	20353
		1	72.8%	44.3%	55.1%	

Positive labels are denoted by 1 and negative labels by 0

Table 4 Positive pruning testing accuracy, precision, recall, and F-measure with (a) GMM and (b) BGM for the BGL, Openstack and Thunderbird data sets

Data set	Testing accuracy	Label	Precision	Recall	F-measure
(a)					
BGL	97.1%-(97.8%)-98.4%	0	71.6%-(76.8%)-81.9%	99.9%-(99.9%)-100%	83.4%-(86.9%)-90.0%
		1	99.9%-(**99.9**%)-100%	96.8%-(97.6%)-98.2%	98.4%-(98.8%)-99.1%
Openstack	99.8%-(99.9%)-100%	0	98.8%-(99.8%)-100%	99.3%-(99.8%)-100%	99.1%-(99.8%)-100%
		1	99.9%-(**99.9**%)-100%	99.8%-(99.9%)-100%	99.9%-(99.9%)-100%
Thunderbird	89.9%-(90.0%)-90.5%	0	49.3%-(50.4%)-59.6%	99.9%-(99.9%)-100%	66.0%-(67.0%)-74.7%
		1	99.9%-(**99.9**%)-100%	88.8%-(88.9%)-89.0%	94.1%-(94.1%)-94.2%
(b)					
BGL	97.3%-(97.6%)-98.2%	0	73.1%-(75.3%)-80.6%	99.9%-(99.9%)-100%	84.5%-(85.9%)-89.2%
		1	99.9%-(**99.9**%)-100%	97.1%-(97.4%)-98.1%	98.5%-(98.7%)-99.0%
Openstack	99.8%-(99.9%)-99.9%	0	98.5%-(99.4%)-99.9%	99.6%-(99.8%)-99.9%	99.1%-(99.6%)-99.9%
		1	99.9%-(**99.9**%)-99.9%	99.8%-(99.9%)-99.9%	99.9%-(99.9%)-99.9%
Thunderbird	89.9%-(90.0%)-90.1%	0	49.3%-(49.5%)-49.8%	99.9%-(99.9%)-100%	66.0%-(66.2%)-66.5%
		1	99.9%-(**99.9**%)-100%	88.9%-(88.9%)-89.1%	94.1%-(94.1%)-94.2%

The minimum, maximum, and average (in parenthesis) values are given for 10 runs. Positive labels are denoted by 1 and negative labels by 0

Table 5 Negative pruning testing accuracy, precision, recall, and F-measure for the BGL data set with (a) GMM, (b) K-means and (c) BGM methods for five rounds (with a GMM for positive pruning)

Round	Testing accuracy	Label	Precision	Recall	F-measure
(a)					
1	76.2%-(82.6%)-88.9%	0	75.0%-(82.4%)-100%	81.5%-(99.0%)-100%	85.7%-(89.7%)-93.1%
		1	55.8%-(97.4%)-100%	16.1%-(29.1%)-100%	27.8%-(40.9%)-71.9%
2	85.4%-(96.0%)-99.2%	0	99.9%-(99.9%)-100%	81.5%-(94.9%)-99.0%	89.8%-(97.3%)-99.5%
		1	58.6%-(86.2%)-97.1%	99.9%-(99.9%)-100%	73.9%-(91.8%)-98.5%
3	81.9%-(90.6%)-99.5%	0	99.3%-(99.9%)-100%	81.0%-(90.2%)-99.5%	89.5%-(94.6%)-99.7%
		1	19.7%-(47.3%)-89.8%	86.6%-(99.3%)-100%	32.9%-(59.1%)-94.6%
4	81.9%-(91.1%)-99.5%	0	99.9%-(100%)-100%	81.0%-(90.7%)-99.5%	89.5%-(94.9%)-99.7%
		1	19.4%-(50.9%)-90.1%	99.9%-(99.9%)-100%	32.5%-(62.1%)-94.8%
5	81.9%-(88.7%)-99.0%	0	100%-(**100%**)-100%	81.0%-(88.2%)-98.9%	89.5%-(93.5%)-99.5%
		1	19.6%-(43.7%)-82.3%	100%-(100%)-100%	32.7%-(55.1%)-90.3%
(b)					
1	76.1%-(81.8%)-87.1%	0	75.0%-(80.8%)-86.4%	99.9%-(99.9%)-100%	85.7%-(89.4%)-92.7%
		1	99.9%-(99.9%)-100%	16.0%-(21.7%)-28.9%	27.6%-(35.5%)-44.8%
2	96.0%-(96.2%)-96.3%	0	95.3%-(95.5%)-95.7%	99.9%-(99.9%)-100%	97.6%-(97.7%)-97.8%
		1	99.9%-(99.9%)-100%	72.6%-(80.8%)-85.3%	84.1%-(89.3%)-92.1%
3	86.9%-(87.7%)-91.1%	0	94.9%-(95.8%)-99.9%	91.1%-(91.2%)-91.2%	93.0%-(93.4%)-95.3%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
4	86.9%-(87.5%)-90.8%	0	94.9%-(95.6%)-99.9%	90.8%-(91.1%)-91.2%	93.0%-(93.3%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
5	87.0%-(87.7%)-90.8%	0	95.0%-(**95.9%**)-99.9%	90.8%-(91.1%)-91.2%	93.0%-(93.4%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
(c)					
1	76.2%-(88.2%)-96.1%	0	75.0%-(88.1%)-100%	81.5%-(98.4%)-100%	85.7%-(92.7%)-97.6%
		1	55.8%-(94.3%)-100%	16.1%-(57.5%)-100%	27.8%-(67.5%)-89.7%
2	85.4%-(96.0%)-99.2%	0	99.9%-(99.9%)-100%	81.5%-(94.9%)-99.0%	89.8%-(97.3%)-99.5%
		1	58.6%-(86.2%)-97.1%	99.9%-(99.9%)-100%	73.9%-(91.8%)-98.5%
3	81.9%-(90.6%)-99.5%	0	99.3%-(99.9%)-100%	81.0%-(90.2%)-99.5%	89.5%-(94.6%)-99.7%
		1	19.7%-(47.3%)-89.8%	86.6%-(99.3%)-100%	32.9%-(59.1%)-94.6%
4	67.3%-(89.8%)-99.5%	0	99.3%-(99.9%)-100%	67.3%-(89.4%)-99.5%	80.4%-(94.1%)-99.7%
		1	0.1%-(47.2%)-90.1%	86.6%-(99.3%)-100%	0.1%-(57.9%)-94.8%
5	81.9%-(88.7%)-99.0%	0	100%-(**100%**)-100%	81.0%-(88.2%)-98.9%	89.5%-(93.5%)-99.5%
		1	19.6%-(43.7%)-82.3%	100%-(100%)-100%	32.7%-(55.1%)-90.3%

The minimum, maximum and average (in parenthesis) values are given for 10 runs. Positive labels are denoted by 1 and negative labels by 0

Table 6 Negative pruning testing accuracy, precision, recall, and F-measure for the Openstack data set with (a) GMM, (b) K-means and (c) BGM methods for five rounds (with a GMM for positive pruning)

Round	Testing accuracy	Label	Precision	Recall	F-measure
(a)					
1	61.5%-(64.3%)-94.0%	0	99.1%-(99.9%)-100%	61.5%-(64.3%)-94.0%	76.2%-(78.0%)-96.9%
		1	0.0%-(0.1%)-1.6%	0.0%-(4.2%)-52.8%	0.0%-(0.2%)-3.1%
2	61.5%-(83.9%)-94.0%	0	100%-(100%)-100%	61.5%-(83.8%)-94.0%	76.2%-(90.7%)-96.9%
		1	0.0%-(0.7%)-9.3%	0.0%-(7.5%)-100%	0.0%-(1.3%)-17.1%
3	64.9%-(88.8%)-98.3%	0	100%-(100%)-100%	64.9%-(88.8%)-98.3%	78.7%-(93.8%)-99.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
4	71.6%-(92.0%)-98.2%	0	100%-(100%)-100%	71.6%-(92.0%)-98.2%	83.4%-(95.7%)-99.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
5	58.7%-(88.6%)-98.2%	0	100%-(**100%**)-100%	58.7%-(88.6%)-98.2%	74.0%-(93.6%)-99.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
(b)					
1	62.7%-(62.8%)-62.9%	0	99.0%-(99.9%)-100%	62.7%-(62.8%)-62.9%	76.9%-(77.1%)-77.3%
		1	0.0%-(0.1%)-1.4%	0.0%-(4.5%)-46.7%	0.0%-(0.3%)-2.8%
2	62.8%-(84.1%)-90.4%	0	100%-(100%)-100%	62.8%-(84.1%)-90.4%	77.1%-(90.9%)-95.0%
		1	0.0%-(1.0%)-9.3%	0.0%-(11.4%)-100%	0.0%-(1.9%)-17.1%
3	90.4%-(91.6%)-94.6%	0	100%-(100%)-100%	90.4%-(91.6%)-94.6%	95.0%-(95.6%)-97.2%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
4	91.4%-(93.2%)-94.6%	0	100%-(100%)-100%	91.4%-(93.2%)-94.6%	95.5%-(96.5%)-97.2%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
5	78.6%-(91.8%)-94.6%	0	100%-(**100%**)-100%	78.6%-(91.8%)-94.6%	88.0%-(95.7%)-97.2%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
(c)					
1	61.5%-(64.3%)-94.0%	0	99.1%-(99.9%)-100%	61.5%-(64.3%)-94.0%	76.1%-(78.0%)-96.9%
		1	0.0%-(0.1%)-1.6%	0.0%-(4.3%)-53.8%	0.0%-(0.2%)-3.2%
2	61.5%-(82.0%)-90.4%	0	100%-(100%)-100%	61.5%-(82.0%)-90.4%	76.1%-(89.6%)-95.0%
		1	0.0%-(0.8%)-9.3%	0.0%-(8.8%)-100%	0.0%-(1.5%)-17.1%
3	53.6%-(83.3%)-98.3%	0	100%-(100%)-100%	53.6%-(83.3%)-98.3%	69.8%-(90.1%)-99.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
4	71.6%-(89.8%)-98.2%	0	100%-(100%)-100%	71.6%-(89.8%)-98.2%	83.4%-(94.5%)-99.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
5	54.4%-(84.8%)-98.2%	0	100%-(**100%**)-100%	54.4%-(84.8%)-98.2%	70.5%-(91.1%)-99.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%

The minimum, maximum and average (in parenthesis) values are given for 10 runs. Positive labels are denoted by 1 and negative labels by 0

Table 7 Negative pruning testing accuracy, precision, recall, and F-measure for the Thunderbird data set with (a) GMM, (b) K-means and (c) BGM methods for five rounds (with a GMM for positive pruning)

Round	Testing accuracy	Label	Precision	Recall	F-measure
(a)					
1	99.7%-(99.8%)-99.8%	0	99.9%-(99.9%)-100%	99.6%-(99.6%)-99.6%	99.8%-(99.8%)-99.8%
		1	99.4%-(99.6%)-99.6%	99.9%-(99.9%)-100%	99.7%-(99.8%)-99.8%
2	84.6%-(88.1%)-88.4%	0	99.9%-(99.9%)-100%	84.6%-(88.1%)-88.4%	91.7%-(93.7%)-93.9%
		1	0.0%-(0.1%)-0.1%	0.0%-(10.0%)-100%	0.0%-(0.1%)-0.1%
3	90.0%-(90.6%)-90.7%	0	99.9%-(99.9%)-100%	90.0%-(90.6%)-90.7%	94.7%-(95.1%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
4	45.2%-(82.2%)-90.7%	0	99.9%-(99.9%)-100%	45.2%-(82.2%)-90.7%	62.3%-(89.0%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
5	45.2%-(83.0%)-90.7%	0	99.9%-(**99.9**%)-100%	45.2%-(83.0%)-90.7%	62.3%-(89.9%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
(b)					
1	70.9%-(78.0%)-99.8%	0	100%-(100%)-100%	51.2%-(57.0%)-99.6%	67.7%-(72.1%)-99.8%
		1	58.1%-(69.4%)-99.6%	100%-(100%)-100%	73.5%-(81.7%)-99.8%
2	35.0%-(35.3%)-35.8%	0	100%-(100%)-100%	35.0%-(35.3%)-35.8%	51.8%-(52.2%)-52.7%
		1	0.0%-(0.1%)-0.1%	0.0%-(17.4%)-100%	0.0%-(0.1%)-0.1%
3	36.1%-(36.5%)-37.9%	0	100%-(100%)-100%	36.1%-(36.5%)-37.9%	53.0%-(53.4%)-55.0%
		1	0.0%-(0.1%)-0.1%	0.0%-(5.9%)-100%	0.0%-(0.1%)-0.1%
4	36.1%-(36.6%)-38.0%	0	100%-(100%)-100%	36.1%-(36.6%)-38.0%	53.0%-(53.6%)-55.0%
		1	0.0%-(0.1%)-0.1%	0.0%-(5.5%)-100%	0.0%-(0.1%)-0.1%
5	29.7%-(38.2%)-61.2%	0	100%-(**100**%)-100%	29.7%-(38.2%)-61.2%	45.8%-(54.9%)-76.0%
		1	0.0%-(0.1%)-0.1%	0.0%-(4.5%)-100%	0.0%-(0.1%)-0.1%
(c)					
1	99.7%-(99.8%)-99.8%	0	99.9%-(99.9%)-100%	99.6%-(99.6%)-99.6%	99.8%-(99.8%)-99.8%
		1	99.4%-(99.6%)-99.6%	99.9%-(99.9%)-100%	99.7%-(99.8%)-99.8%
2	84.6%-(88.1%)-88.4%	0	99.9%-(99.9%)-100%	84.6%-(88.1%)-88.4%	91.7%-(93.7%)-93.9%
		1	0.0%-(0.1%)-0.1%	0.0%-(4.7%)-40.0%	0.0%-(0.1%)-0.1%
3	90.0%-(90.6%)-90.7%	0	99.9%-(99.9%)-100%	90.0%-(90.6%)-90.7%	94.7%-(95.1%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
4	45.2%-(83.0%)-90.7%	0	99.9%-(99.9%)-100%	45.2%-(83.0%)-90.7%	62.3%-(89.6%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%
5	45.2%-(83.7%)-90.7%	0	99.9%-(**99.9**%)-100%	45.2%-(83.7%)-90.7%	62.3%-(90.4%)-95.1%
		1	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%	0.0%-(0.0%)-0.0%

The minimum, maximum and average (in parenthesis) values are given for 10 runs. Positive labels are denoted by 1 and negative labels by 0

BGL

The BlueGene/L (BGL) data set has 4,399,502 positive logs and 348,460 negative logs. From these, 94,960 logs are used for the training set t_1 and 4,653,002 for the remaining set r_1. Using GMM for positive pruning, the final average testing accuracy is 99.5% with average precision, recall and F-measure of 95.6%, 97.8% and 96.7% for negative logs, and 99.8%, 99.6% and 99.7% for positive logs, respectively. Using BGM for positive pruning, the final average testing accuracy is 99.5% with average precision, recall and F-measure of 95.4%, 98.5% and 96.9% for negative logs, and 99.9%, 99.6% and 99.7% for positive logs, respectively.

The results with Auto-LSTM, IKNN and nLSALog for the BGL data set are given in Table 2a. The precision, recall and F-measure results for negative logs are better than the 92%, 91% and 92%, respectively, with the improved K-nearest neighbors (IKNN) supervised algorithm [36]. The precision, recall and F-measure results for negative logs are also better than the 82.5%, 94.7% and 88.2%, respectively, with the nLSALog algorithm [40]. The precision, recall and F-measure results for negative logs are also better than the 98%, 91.3% and 94.5%, respectively, with the Auto-LSTM algorithm [10] (however, the precision with Auto-LSTM is higher). Several well-known models were also evaluated for anomaly detection. The average testing accuracy, precision, recall, F-measure and time with the BGM, EEnvelope, GMM, K-means, LOF, and OC-SVM methods for the BGL data set using 10-fold cross-validation are given in Table 3a. Among existing methods, the GMM results for negative logs are the highest with precision, recall and F-measure of 38.2%, 50% and 43.3%, but these values are lower than those for the proposed model. The proposed model results are better because of pruning positive and negative logs and using DL. Because of the high complexity of the LOF and OC-SVM methods [2, 9, 29], only 5% of the data set was used for these models.

Openstack

The Openstack data set has 137,074 positive log messages and 18,434 negative log messages. From these, 3111 logs are used for the training set t_1 and 152,397 for the remaining set r_1. Using GMM for positive pruning, the final average testing accuracy is 99.9% with average precision, recall and F-measure of 99.9%, 99.7% and 99.8% for negative logs, and 99.9%, 99.9% and 99.9% for positive logs, respectively. Using BGM for positive pruning, the final average testing accuracy is 99.9% with average precision, recall and F-measure of 99.7%, 99.9% and 99.8% for negative logs, and 99.9%, 99.9% and 99.9% for positive logs, respectively.

The results with Auto-LSTM and Deeplog for the Openstack data set are given in Table 2b. The precision, recall and F-measure results for negative logs are better than the 94%, 99% and 97% obtained with the Deeplog network [7]. The precision, recall and F-measure results for negative logs are also better than the 99.4%, 92.8% and 96%, respectively, with the Auto-LSTM algorithm [10]. Several well-known models were also evaluated for anomaly detection. The average testing accuracy, precision, recall, F-measure and time with the BGM, EEnvelope, GMM, K-means, LOF, and OC-SVM methods for the Openstack data set using 10-fold cross-validation are given in Table 3b. Among existing methods, the EEnvelope results for negative logs are the highest with precision, recall and F-measure of 53.4%, 44.9% and 48.8%, but these values are lower than those for the proposed model. The proposed model results are better because of pruning positive and negative logs and using DL.

Thunderbird

From the Thunderbird data set, 3,000,000 positive log messages and 324,824 negative log messages are used. Of these, 66,497 messages are used for the training set t_1 and 3,258,327 for the remaining set r_1. Using GMM for positive pruning, the final average testing accuracy is 99.8% with average precision, recall and F-measure of 98.9%, 99.6% and 99.3% for negative logs, and 99.9%, 99.8% and 99.9% for positive logs, respectively. Using BGM for positive pruning, the final average testing accuracy is 99.8% with average precision, recall and F-measure of 99%, 99.6% and 99.3% for negative logs, and 99.9%, 99.8% and 99.9% for positive logs, respectively.

The results with Auto-LSTM and IKNN for the Thunderbird data set are given in Table 2c. The precision, recall and F-measure results for negative logs are better than the 96% for all criteria with the IKNN supervised algorithm [36]. The precision, recall and F-measure results for negative logs are about the same as the 98.4%, 99.8% and 99.1%, respectively, with the Auto-LSTM algorithm [10]. Several well-known models were also evaluated for anomaly detection. The average testing accuracy, precision, recall, F-measure and time with the BGM, EEnvelope, GMM, K-means, LOF, and OC-SVM methods for the Thunderbird data set using 10-fold cross-validation are given in Table 3c. Among existing methods, the GMM results for negative logs are the highest with precision, recall and F-measure of 27.1%, 70% and 37.2%, but these values are lower than those for the proposed model. The proposed model results are better because of pruning positive and negative logs and using DL. Because of the high complexity of the LOF and OC-SVM methods, only 5% of the data set was used for these models.

Discussion

Gaussian mixture model (GMM), Dirichlet process Gaussian mixture model (BGM), and K-means are well-known clustering algorithms. Clustering algorithms have been shown to provide good results with text data [1] and logs are mostly text. In addition, clustering algorithms are faster to train than DL algorithms [26]. However, an unsupervised GMM is used here for pruning positive logs and the unsupervised GMM, BGM, and K-means methods are used for pruning negative logs. This eliminates the need to label log messages to detect anomalies. The positive and negative logs are selected unsupervised using Algorithms 1 and 2, respectively. If the conditions in these algorithms are satisfied, then the logs predicted to be positive and negative are added to o_0 and z_0, respectively, for Algorithm 1, and o_1 and z_1, respectively, for Algorithm 2. Then reliable positive and negative logs are selected using o_0 and z_1, respectively, in Algorithm 3. The amount of positive data is far greater than the amount of negative data, so the positive data can be accurately predicted using clustering algorithms. However, the negative cluster contains a lot of positive data and this is a disadvantage of using clustering methods with imbalanced data [23]. We take advantage of this data imbalance as GMM can easily predict positive data. Another disadvantage of unsupervised clustering is that clusters may be labeled incorrectly [27]. Thus, not only do negative clusters include positive logs but clusters may be incorrectly labeled in different runs. As a consequence, the average results shown in Table 3 for the BGM, GMM and K-means methods are poor. Comparing the results in Tables 1 and 3, it is evident that BGM, GMM and K-means alone do not provide good log message anomaly detection results. This is also due to the complexity of the unstructured log messages. For negative pruning, our experimental results indicate that using just one model can limit the pruning process so that many unreliable logs are retained. Thus, multiple models are employed.

The precision for positive logs is the percentage of true positive logs predicted of all logs predicted to be positive. Table 4 shows that with GMM and BGM for positive pruning, the average precision of positive logs is 99.9% for the BGL, Openstack and Thunderbird data sets, which is very high. Thus, most logs that are predicted to be positive are correct, and the number of negative log messages predicted to be positive (F_P) is low. This indicates that pruning positive logs using Algorithm 1 is effective. However, the average precision for negative logs for the BGL and Thunderbird data sets is around 76% and 50%, respectively, which is quite low.

In separate experiments, the value of the constant c and the threshold for a were varied to determine their effect on Algorithm 1. The first set of experiments considered $c = 1.4, 1.5, 1.6, 1.7$, and 1.8 with $a > 3$ and the second set of experiments considered $a > 3$, 4, 6, and 10 with $c = 1.6$. The criterion for this algorithm is that the set z_0 is not empty. This is because Algorithm 2 requires this data for training. In the first set of experiments, the criterion was satisfied for all values of c with the Openstack and Thunderbird data sets and for all values except $c = 1.8$ with the BGL data set. In the second set of experiments, the criterion was satisfied for $a = 3$, 4, and 6 with the BGL and Openstack data sets and for $a = 3$ and 4 with the Thunderbird data set.

The effect of the negative log pruning algorithm is shown in Tables 5, 6 and 7 for the BGL, Openstack and Thunderbird data sets (with a GMM for positive pruning), respectively. Here, precision for the negative logs is the most important criterion. For the BGL data set, the average precision of the negative logs with the GMM, K-means and BGM methods increased from 82.4 to 100%, 80.8–95.9% and 88.1– 100%, respectively, over the five rounds. For the Openstack data set, the average precision of the negative logs with the GMM, K-means and BGM methods increased from 99.9 to 100% for all models over the five rounds. For the Thunderbird data set, the average precision of the negative logs with the GMM, K-means and BGM methods was approximately the same, 99.9–100%, over the five rounds. These results indicate that Algorithm 2 is very effective in pruning negative logs. Further, the BGL data set required five rounds to obtain good results but only two rounds were sufficient for the Openstack and Thunderbird data sets. The GMM, K-means and BGM methods were used here to prune negative logs, but other unsupervised models can be employed.

The adjusted Rand index [21] is a measure of the similarity of results and has a value between 0 and 1. A high index value means the results are very similar. For negative log pruning with the BGL data set, the average adjusted Rand index of all rounds for GMM and BGM was 0.94, for GMM and K-means was 0.33 and for BGM and K-means was 0.28. For negative log pruning with the Openstack data set, the average adjusted Rand index of all rounds for GMM and BGM was 0.74, for GMM and K-means was 0.49, and for BGM and K-means was 0.37. For negative log pruning with the Thunderbird data set, the average adjusted Rand index of all rounds for GMM and BGM was 0.93, for GMM and K-means was 0.19, and for BGM and K-means was 0.17. These values show that the GMM and BGM results are more similar than the GMM and K-means results and the BGM and K-means results.

The anomaly detection results with the LSTM network using the reliable positive and negative logs are shown in Table 1. The final results with GMM and BGM positive log pruning were similar. The proposed model results are better than with Auto-LSTM because the data was balanced before it was input to the LSTM network for anomaly detection whereas in [10], imbalanced data was used in the network.

The amount of data used for LSTM training was very small (less than 2% for BGL, 3% for Thunderbird, and 18% for Openstack) whereas deep networks typically require a significant amount of training data for convergence. For the Openstack data set, a greater percentage of training data was required for convergence because it is small (more than 20 times smaller than Thunderbird and BGL).

The proposed hybrid model (with unsupervised selection of reliable logs), has three advantages over supervised methods. First, it is suitable for many practical applications as there is no need to label data. Second, labeling data is a time-consuming task and in many cases is not feasible. Third, using an unsupervised method eliminates the human error inherent in labeling. The default hyperparameters were used with the proposed model so better results may be obtained with hyperparameter tuning.

Conclusion

Many millions of log messages are generated each day in cloud and other systems. These messages are important for system maintenance which includes anomaly detection. Log messages consist of unstructured data which is mostly text. Thus, machine learning (ML) is a good choice for anomaly detection. In this paper, a hybrid log message anomaly detection technique using deep learning (DL) was proposed with pruning of positive and negative log messages. An unsupervised algorithm with a Gaussian mixture model (GMM) was used to prune positive logs. Then, an unsupervised algorithm was used to prune negative logs using the K-means, GMM, and Dirichlet Process Gaussian mixture model (BGM) methods iteratively. The precision with the pruning algorithms for positive and negative logs was high, i.e., there were few false positives (F_P). The proposed model was tested on three different log message data sets, namely BGL, Openstack and Thunderbird. The results obtained show that this model is better than other well-known approaches. Future research can consider the effect of adding other unsupervised methods such as isolation forest to the proposed model. Further, a CNN network can be used for anomaly detection instead of an LSTM network and hyperparameter tuning can be investigated.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declarations

Conflict of Interest The authors declare no conflict of interest with regards to this paper.

References

1. Aggarwal CC, Zhai C. A survey of text clustering algorithms. In: Mining text data. Berlin: Springer; 2012. p. 77–128. https://doi.org/10.1007/978-1-4614-3223-4_4.
2. Baoyi W, Xiangyu L, Shaomin Z. An improved outlier detection algorithm K-LOF based on density. Comput Perform Commun Syst. 2017;2(1):1–7. https://doi.org/10.23977/cpcs.2017.21001.
3. Blei DM, Jordan MI. Variational inference for Dirichlet process mixtures. Bayesian Anal. 2006;1(1):121–43. https://doi.org/10.1214/06-BA104.
4. Byeon W, Liwicki M, Breuel TM. Texture classification using 2D LSTM networks. In: International Conference on Pattern Recognition, pp. 1144–1149. 2014.
5. Dargan S, Kumar M, Ayyagari MR, Kumar G. A survey of deep learning and its applications: a new paradigm to machine learning. Arch Comput Methods Eng. 2020;27(4):1071–92. https://doi.org/10.1007/s11831-019-09344-w.
6. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B Methodol. 1977;39(1):1–22. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.
7. Du M, Li F, Zheng G, Srikumar V. DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: ACM Conference on Computer and Communications Security, pp. 1285–1298. 2017. https://doi.org/10.1145/3133956.3134015.
8. Du W, Zhu Z, Wang C, Yue Z. The real-time big data processing method based on LSTM for the intelligent workshop production process. In: IEEE International Conference on Big Data Analytics, pp. 63–67. 2020. https://doi.org/10.1109/ICBDA49040.2020.9101345.
9. Erfani SM, Rajasegarar S, Karunasekera S, Leckie C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. Pattern Recogn. 2016;58:121–34. https://doi.org/10.1016/j.patcog.2016.03.028.
10. Farzad A, Gulliver TA. Log message anomaly detection and classification using auto-B/LSTM and auto-GRU. 2019. arXiv:1911.08744 [Preprint].
11. Farzad A, Gulliver TA. Unsupervised log message anomaly detection. ICT Exp. 2020;6(3):229–37. https://doi.org/10.1016/j.icte.2020.06.003.
12. Fawcett T. An introduction to ROC analysis. Pattern Recogn Lett. 2006;27(8):861–74. https://doi.org/10.1016/j.patrec.2005.10.010.
13. Ferguson TS. A Bayesian analysis of some nonparametric problems. Ann Stat. 1973;1(2):209–30.
14. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge: MIT Press; 2016.
15. Graves A. Supervised sequence labelling with recurrent neural networks. In: Studies in computational intelligence. Berlin: Springer; 2012. https://doi.org/10.1007/978-3-642-24797-2.
16. Guo W, Che L, Shahidehpour M, Wan X. Machine-learning based methods in short-term load forecasting. Electr J. 2021;34(1):106884. https://doi.org/10.1016/j.tej.2020.106884.
17. Harada Y, Yamagata Y, Mizuno O, Choi E. Log-based anomaly detection of CPS using a statistical method. In: International Workshop on Empirical Software Engineering in Practice, pp. 1–6. 2017. https://doi.org/10.1109/IWESEP.2017.12.
18. He P, Zhu J, He S, Li J, Lyu MR. An evaluation study on log parsing and its use in log mining. In: IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 654–661. 2016. https://doi.org/10.1109/DSN.2016.66.
19. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

20. Hoyle B, Rau MM, Paech K, Bonnett C, Seitz S, Weller J. Anomaly detection for machine learning redshifts applied to SDSS galaxies. Mon Not R Astron Soc. 2015;452(4):4183–94. https://doi.org/10.1093/mnras/stv1551.
21. Hubert L, Arabie P. Comparing partitions. J Classif. 1985;2(1):193–218. https://doi.org/10.1007/BF01908075.
22. Kaur H, Kumar M. A comprehensive survey on word recognition for non-Indic and Indic scripts. Pattern Anal Appl. 2018;21(4):897–929. https://doi.org/10.1007/s10044-018-0731-2.
23. Kumar C, Rao K, Govardhan A, Reddy K. Imbalanced K-means: an algorithm to cluster imbalanced-distributed data. Int J Eng Tech Res. 2014;2(2):114–22.
24. Li L, Hansman RJ, Palacios R, Welsch R. Anomaly detection via a Gaussian mixture model for flight operation and safety monitoring. Transp Res Part C: Emerg Technol. 2016;64:45–57. https://doi.org/10.1016/j.trc.2016.01.007.
25. Lin Q, Zhan, H, Lou J, Zhang Y, Chen X. Log clustering based problem identification for online service systems. In: IEEE/ACM International Conference on Software Engineering, pp. 102–111. 2016.
26. Majumder S, Balaji N, Brey K, Fu W, Menzies T. 500+ times faster than deep learning (a case study exploring faster methods for text mining StackOverflow). arXiv:1802.05319 [Preprint] 2018. http://arxiv.org/abs/1802.05319
27. Maligo A, Lacroix S. Classification of outdoor 3D lidar data based on unsupervised Gaussian mixture models. IEEE Trans Autom Sci Eng. 2017;14(1):5–16. https://doi.org/10.1109/TASE.2016.2614923.
28. Miao X, Liu Y, Zhao H, Li C. Distributed online one-class support vector machine for anomaly detection over networks. IEEE Trans Cybern. 2019;49(4):1475–88.
29. Nalepa J, Kawulok M. Selecting training sets for support vector machines: a review. Artif Intell Rev. 2019;52(2):857–900. https://doi.org/10.1007/s10462-017-9611-1.
30. Nissim N, Moskovitch R, Rokach L, Elovici Y. Detecting unknown computer worm activity via support vector machines and active learning. Pattern Anal Appl. 2012;15(4):459–75. https://doi.org/10.1007/s10044-012-0296-4.
31. Paulauskas N, Bagdonas AF. Local outlier factor use for the network flow anomaly detection. Secur Commun Netw. 2015;8(18):4203–12. https://doi.org/10.1002/sec.1335.
32. Reidemeister T, Jiang M, Ward PAS. Mining unstructured log files for recurrent fault diagnosis. In: IFIP/IEEE International Symposium on Integrated Network Management and Workshops, pp. 377–384. 2011.
33. Sun Y, Xu W, Zhang J, Xiong J, Gui G. Super-resolution imaging using convolutional neural networks. In: Communications, signal processing, and systems. Berlin: Springer Nature; 2020. p. 59–66.
34. Syarif I, Prugel-Bennett A, Wills G. Unsupervised clustering approach for network anomaly detection. In: Networked digital technologies. Berlin: Springer; 2012. p. 135–45.
35. Veracini T, Matteoli S, Diani M, Corsini G. Fully unsupervised learning of Gaussian mixtures for anomaly detection in hyperspectral imagery. In: International Conference on Intelligent Systems Design and Applications; 2009. pp. 596–601.
36. Wang B, Ying S, Cheng G, Wang R, Yang Z, Dong B. Log-based anomaly detection with the improved K-nearest neighbor. Int J Softw Eng Knowl Eng. 2020;30(2):239–62. https://doi.org/10.1142/S0218194020500114.
37. Wang J, Cao Z. Chinese text sentiment analysis using LSTM network based on L2 and Nadam. In: IEEE International Conference on Communication Technology, pp. 1891–1895. 2017.
38. Wang SH, Govindaraj VV, Górriz JM, Zhang X, Zhang YD. COVID-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network. Inform Fusion. 2021;67:208–29. https://doi.org/10.1016/j.inffus.2020.10.004.
39. West M, Harrison J. Bayesian forecasting and dynamic models, chap 16: multivariate modelling and forecasting. Berlin: Springer; 1997. p. 581–630.
40. Yang R, Qu D, Gao Y, Qian Y, Tang Y. nLSALog: an anomaly detection framework for log sequence in security management. IEEE Access. 2019;7:181152–64.
41. Yen TF, Oprea A, Onarlioglu K, Leetham T, Robertson W, Juels A, Kirda E. Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In: Annual Computer Security Applications Conference, pp. 199–208. 2013. https://doi.org/10.1145/2523649.2523670.
42. Yuan D, Mai H, Xiong W, Tan L, Zhou Y, Pasupathy S. SherLog: error diagnosis by connecting clues from run-time logs. In: Architectural support for programming languages and operating systems, pp. 143–154. 2010. https://doi.org/10.1145/1736020.1736038.
43. Zhang L. The research of log-based network monitoring system. In: Advances in intelligent systems. Berlin: Springer; 2012. p. 315–20.
44. Zhang Q, Yang LT, Chen Z, Li P. A survey on deep learning for big data. Inform Fusion. 2018;42:146–57. https://doi.org/10.1016/j.inffus.2017.10.006.
45. Zhang YD, Satapathy SC, Guttery DS, Górriz JM, Wang SH. Improved breast cancer classification through combining graph convolutional network and convolutional neural network. Inform Process Manag. 2021;58(2):102439. https://doi.org/10.1016/j.ipm.2020.102439.
46. Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, Lyu MR. Tools and benchmarks for automated log parsing. In: International Conference on Software Engineering: Software Engineering in Practice, pp. 121–130. 2019. https://doi.org/10.1109/ICSE-SEIP.2019.00021.