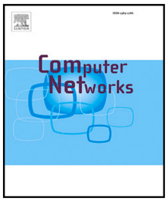
Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge

Zumin Wang [a], Jiyu Tian [a,*], Hui Fang [b], Liming Chen [c], Jing Qin [d]

[a] College of Information Engineering, Dalian University, Dalian, China
[b] Department of Computer Science, Loughborough University, Loughborough, UK
[c] School of Computing, Ulster University, Belfast, UK
[d] School of Software Engineering, Dalian University, Dalian, China

ARTICLE INFO

Keywords:
Log anomaly detection
Temporal convolutional network
Global average pooling
Pointwise-convolution
Edge computing

ABSTRACT

Log anomaly detection on edge devices is the key to enhance edge security when deploying IoT systems. Despite the success of many newly proposed deep learning based log anomaly detection methods, handling large-scale logs on edge devices is still a bottleneck due to the limited computational power on these devices to fulfil the real-time processing requirement for accurate anomaly detection. In this work, we propose a novel lightweight log anomaly detection algorithm, named LightLog, to tackle this research gap. In specific, we achieve real-time processing speed on the task via two aspects: (i) creation of a low-dimensional semantic vector space based on word2vec and post-processing algorithms (PPA); and (ii) design of a lightweight temporal convolutional network (TCN) for the detection. These two components significantly reduce the number of parameters and computations of a standard TCN while improving the detection performance. Experimental results show that our LightLog outperforms several benchmarking methods, namely DeepLog, LogAnomaly and RobustLog, by achieving 97.0 F1 score on HDFS Dataset and 97.2 F1 score on BGL with smallest model size. This effective yet efficient method paves the way to the deployment of log anomaly detection on the edge. Our source code and datasets are freely available on https://github.com/Aquariuaa/LightLog.

1. Introduction

Log anomaly detection plays an important role in maintaining normal operation and ensuring security of a network system. Routinely, logs record operational status of different devices in a network over a period of time. By detecting abnormal events in logs, it facilitates the identification of time points and root causes where system faults and malicious attacks occur [1–3]. Recently, many algorithms, including traditional machine learning methods, e.g., principal component analysis (PCA) [4] and Random Forest [5], and deep learning based networks, e.g., Autoencoder [6] and long short-term memory (LSTM) [7], have been proposed to achieve good performance for log anomaly detection.

Despite the progress made on log anomaly detection for high-performance computing platforms, it is still difficult to achieve accurate anomaly detection on edge devices in a real-time manner [8,9]. For log anomaly detection of IoT systems, log data generated on edge devices is collected and sent to cloud platform for storage through network. After this, high-performance computing cluster is used to detect anomalous logs [10–12]. Existing methods, especially deep learning based algorithms, require expensive computational power to detect anomalies with high accuracy. Thus, these state-of-the-art algorithms cannot be deployed on edge devices due to their heavy computational consumption [13]. Consequently, there is growing interest to design lightweight log anomaly detection method which is more suitable for its deployment on edge devices.

We propose a novel lightweight log anomaly detection method to meet the real-time and high-accuracy anomaly detection requirement with limited computational resources on edge devices. To achieve this goal, we create a low-dimensional semantic vector space by using word2vec [14] and post-processing algorithms (PPA) [15] algorithms to generate more compact and effective feature representations extracted from unstructured logs. Furthermore, a lightweight temporal convolutional network is designed by leveraging point-wise convolutional operation and global average pooling. Compared to state-of-the-art log anomaly detection methods proposed in recent years [7,16,17], our method has significant advantages in terms of detection accuracy

[*] Corresponding author.
E-mail addresses: wangzumin@dlu.edu.cn (Z. Wang), tianjiyu@s.dlu.edu.cn (J. Tian), h.fang@lboro.ac.uk (H. Fang), l.chen@ulster.ac.uk (L. Chen), qinjing@dlu.edu.cn (J. Qin).

https://doi.org/10.1016/j.comnet.2021.108616
Received 25 July 2021; Received in revised form 3 November 2021; Accepted 8 November 2021
Available online 10 November 2021

and computational consumption. We summarize the contributions of our proposed algorithm as follows:

- We propose a log anomaly detection algorithm for real-time processing on edge devices. The proposed method avoids uploading large-scale log files to network server for analysis, thus enabling local real-time detection processing which enhances the edge reliability and security.
- We create a new mapping method that significantly reduces the dimensionality of the transformed semantic vector of log templates. This method solves the neglection of semantic similarity of templates in existing methods using one-hot encoding. In addition, it reduces the huge computational burden of the downstream processing pipeline during training and detection.
- We design a lightweight temporal convolutional network (TCN) for the supervised classification of temporal log data. The model uses multi-kernel point-wise convolution to improve residual blocks in the TCN, and global average pooling to reduce the number of parameters of the model. This results in a significant reduction of floating point computations (FLOPs), number of parameters, detection model size and detection time of the algorithm while improving detection accuracy.

The remainder of this paper is organized as follows: Section 2 presents related work, while Section 3 describes the proposed log anomaly detection approach, including the overall detection framework, log parsing and serialization methods, and the TCN network. In Section 4, we evaluate the performance of the proposed approach by comparing with several state-of-the-art algorithms. Finally, the work is concluded in Section 5.

2. Related work

2.1. Log anomaly detection

A typical log anomaly detection method has two phases: pre-processing of unstructured log data, including log parsing and mapping, and training of downstream anomaly detection model.

At the pre-processing stage, many different mapping strategies are designed to transform raw logs into meaningful representations. The DeepLog method [7] parsed unstructured logs into structured text and extracts templates and parameters from the structured logs for analysis respectively. It is worth noting that DeepLog uses the One-Hot encoding method in the process of mapping log templates to tags. In this encoding, the log template is converted into a sparse vector without any semantic information. To preserve more high semantic information, the LogAnomaly method proposed by Meng et al. [17] extracted semantic vectors via the Word2Vec algorithm [14]. Their further research [18] demonstrated an application-specific word embedding method for logs that extracts accurate semantic information from logs and has capacity of handling new words outside of predefined vocabularies. In the same way, to cope with the changing and unseen log formats in practice, the SwissLog method [19] used a BERT encoder to encode the log template to capture contextual information in the log statements, and the HitAnomaly method [20] utilized a hierarchical transformer structure to model both log template sequences and parameter values.

At the classification stage, both machine learning methods [21,22] and deep learning methods [7,17] are utilized to detect log anomalies. Deep learning based algorithms have outperformed traditional machine learning methods on the log anomaly detection task. In [7], stacked LSTM was used to capture dynamic patterns from the text sequences. Chen et al. [23] applied an LSTM network to extract the sequential patterns of logs and propose a novel transfer learning method sharing fully connected networks between source and target systems, to minimize the impact of noises in anomalous log sequences. The RobustLog method [16] introduced attention mechanism in Bi-LSTM to further improve the detection reliability via enhancing the recurrent neural network. Based on the experimental results, these deep learning based methods achieved significant performance improvement compared to classical detectors, such as PCA based method [21] and Invariant Mining (IM) based method proposed by Lou et al. [22]. As highlighted in [16], anomaly identification from real-world log sequences becomes unstable when both scale and complexity of log data in real industrial production scenarios increase although the above-mentioned approaches enable intelligent detection from log data.

However, the deployment of methods on edge computing devices requires more factors to be taken into account. The computational burden imposed by detection methods can negatively impact the operational performance of edge computing units [24]. Admittedly, the above methods are capable of effective detection from log data, but it remains to be explored how to achieve high accuracy detection within the constraints of limited computing resources on edge devices. For example, the One-Hot encoding approach in the DeepLog method allows for low-dimensional input when template types are scarce, but the lack of semantic information still limits the detection accuracy of the method. In scenarios where template types are abundant, the computing demands of the high-dimensional input generated by this encoding approach are enormous. Although the LogAnomaly and RobustLog methods take attention to the impact of missing information on anomaly detection, these methods are similarly based on high-dimensional semantic vectors. In addition, supervised or unsupervised detection models trained by LSTMs have a large number of FLOPs, parameters, and excessive detection times, making it difficult to deploy existing methods effectively on edge computing devices.

2.2. Temporal convolutional network

Temporal Convolutional Network (TCN) is a new convolutional neural network (CNN) based network architecture that is designed to deal with time-series prediction tasks [25,26]. It has been proven to have several distinctive advantages compared to recurrent neural network architectures which include: (1) TCNs can be used to model both short-range and long-range dependencies in time series data via multiple receptive fields; (2) it is easy to parallelize the processing compared to recurrent neural networks (RNNs); and (3) it solves the gradient explosion and disappearance problem when modelling long-range relationship. Thus, TCN raises significant attentions for temporal signal processing since the empirical study in [26].

TCN has been applied in various applications due to its superior performance on time-series analysis. Yan et al. [27] proposed to use TCN on weather forecasting. This study demonstrated that TCNs achieved higher accuracy in this prediction task when compared to LSTMs. Chen et al. [28] designed a TCN-based probabilistic prediction framework to estimate probability density of multi-attributes time series data under parametric and non-parametric settings, where stacked residual blocks based on dilated-causal convolutions were reconstructed to better capture the temporal correlation of sequences. He et al. [29] was the first to apply TCN to anomaly detection in an unsupervised manner by training a TCN model on normal sequences. After modelling normal sequences, a Gaussian distribution was used to fit the prediction error for determining whether the sequence values are anomalous. This method has been applied to some complex problem scenarios, such as electrocardiograms (ECG) anomaly detection and screen anomaly detection. Cheng et al. [30] proposed a hierarchical TCN to implement semi-supervised learning to alleviate the problem of insufficient number of labels in anomaly detection of IoT smart devices.

Our work is distinctive with other TCN models [29,30] on both computational efficiency and accuracy. It is found that the fusion of multi-channel feature maps with multi-convolutional kernels to extract underlying features of sequential data leads to more stable prediction results compared to the original TCN model. Further, the use of point-wise convolutional operation in convolution layers and Global Average Pooling (GAP) to replace fully-connected layer significantly reduces the parameter number of the model, thus allowing affordable computation on edge devices.

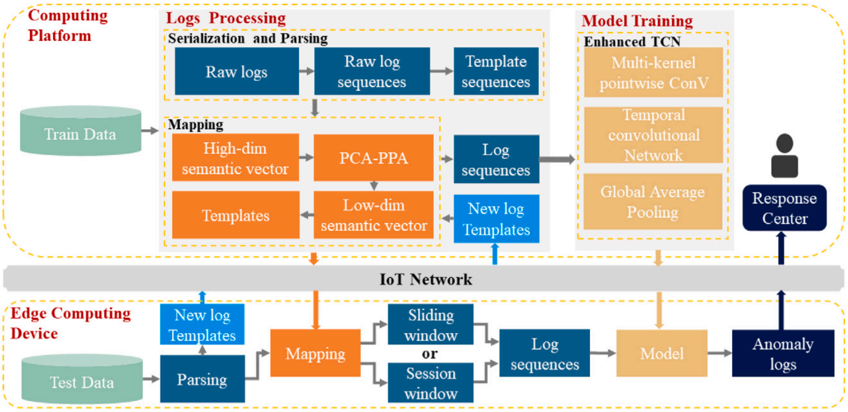


Fig. 1. The framework of LightLog.

3. The LightLog method

In this section, we describe the details of our proposed LightLog method. After introducing the framework, we present the preprocessing step of transforming unstructured log sequence data to compact feature representations and the implementation of the Lightweight TCN network.

3.1. The LightLog framework

The proposed LightLog aims to provide real-time local anomaly detection of logs with low consumption of computing resources, while achieving high detection accuracy. As illustrated in Fig. 1, the LightLog is composed of two phases: the training phase is conducted on cloud computing platform which is supported by high-performance computing clusters, and at the deployment phase, the mapping and enhanced TCN model are implemented on edge computing devices for real-time log processing.

At the training phase, the log data collected from edge devices is first serialized and transmitted to cloud computing platform based on a sliding window or a session window consisting of unique identifiers such as its block ID, thread number and task number etc. Following this step, the unstructured raw logs are transformed into log sequences from which low-dimensional semantic vectors and log templates are obtained for analysis (details are provided in 3.2). Then, the lightweight TCN model is trained for detecting log anomalies from the data (details are provided in 3.3). The transformation process is carried out through the Mapping component and the result of the transformation is used as input to the downstream detection algorithm. In particular, the Mapping component uses the PCA-PPA algorithm to reduce the dimensionality of the high-dimensional semantic vectors generated by the log templates, and ultimately to establish a mapping relationship between the templates and the low-dimensional semantic vectors. This mapping relationship is maintainable and can be reconstructed on the computing platform when a new log template is evolved. Then, the original TCN is enhanced by multi-kernel pointwise convolution and global average pooling to train lightweight models to detect log anomalies from the data (details are provided in 3.3).

At the deployment phase, the mapping function and enhanced TCN model are implemented on edge computing devices. Due to its lightweight architecture, the LightLog algorithm is capable of detecting anomalies from logs reliably on local devices with affordable computational consumption. Thus, it facilitates instant responses when anomalies occur. As a result, this deployment strategy avoids lengthy processing pipeline to transmit all log files to server and wait in a queue for anomaly detection. In addition, it is important to be aware that the new logs are merged into the log sequence as unique identifier-based session windows or sliding windows to form a new test sequence. To better fits the application scenario, the test task also includes a component for uploading exception logs that will be uploaded to the response centre of the computing platform for alerting.

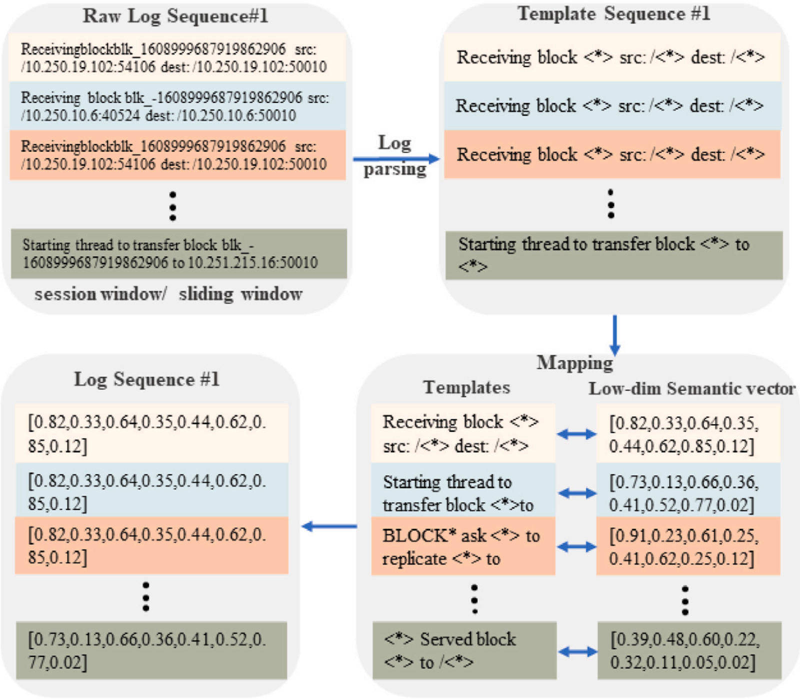
Overall, the training stage of the LightLog method is conducted on a central computing platform. While the mapping function and the enhanced TCN model are then distributed to edge computing devices for deployment. This strategy enables local real-time detection of online logs through the lightweight detection model with consistently uploading abnormal logs and new templates to the cloud computing platform for future model refinement. In addition, our method significantly reduces the data transmission between edge devices and cloud computing platform. For example, in the log datasets HDFS [21] and BGL [31] built in a real environment, abnormal logs only account for 2 percent and 1 percent of the total logs, respectively.

3.2. Logs processing

A log file contains a sequence of textual records and each record is consisted of a timestamp and a text message, that indicates operational status of a device. As log format is a set of specialized vocabularies or abbreviations defined by different equipment manufacturers, it requires a standardization stage to make the data well-structured for analysis. A typical raw log message contains two parts, i.e. log templates and event parameters. We provide an illustration of raw log sequence in Fig. 2. In this example, the event parameters *blk-1608999687919862906*, *10.250.19.102:54106* and *10.250.19.102:50010* generated in the production environment are concatenated with the log template *Receiving block <*>src: /<*>dest: /<*>* to get a specific log message *Receiving block blk-1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010*. When a new set of event parameters *blk-1608999687919862906*, *10.250.10.6:40524*, *10.250.10.6:50010* is generated, another log message *Receiving block blk-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010* is recorded. In the LightLog, we follow the previous work [16,17,32], and focus on the analysis of log templates.

In a nutshell, log parsing is equivalent to the process of reversing log printing, that is, reversing the processing of log messages. Log anomaly detection relies on parsed log templates. As this is not the focus of this study, the log parsing techniques used for the process can be implemented by referring to the methods given in the literature [33].

As illustrated in Fig. 2, after a structured sequence of log templates is generated, we utilize Word2Vec model [34] to convert template sequence into semantic vectors. This mapping function projects the templates into a sequence of 300-dimensional vectors by preserving semantic similarities of the templates in sequences. It provides better representations of log content compared to one-hot encoding method as explained in [35–38]. Despite the advantages Word2Vec method, the length of the sequence representations by using this 300-dimensional



```
Receivingblockblk_1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010
Receiving block blk_-1608999687919862906 src: /10.250.10.6:40524 dest: /10.250.10.6:50010
Receivingblockblk_1608999687919862906 src: /10.250.19.102:54106 dest: /10.250.19.102:50010
⋮
Starting thread to transfer block blk_-1608999687919862906 to 10.251.215.16:50010
```

```
Receiving block <*> src: /<*> dest: /<*>
Receiving block <*> src: /<*> dest: /<*>
Receiving block <*> src: /<*> dest: /<*>
⋮
Starting thread to transfer block <*> to <*>
```

```
Receiving block <*> src: /<*> dest: /<*>    ↔  [0.82,0.33,0.64,0.35,0.44,0.62,0.85,0.12]
Starting thread to transfer block <*>to     ↔  [0.73,0.13,0.66,0.36,0.41,0.52,0.77,0.02]
BLOCK* ask <*> to replicate <*> to          ↔  [0.91,0.23,0.61,0.25,0.41,0.62,0.25,0.12]
⋮                                              ⋮
<*> Served block <*> to /<*>                ↔  [0.39,0.48,0.60,0.22,0.32,0.11,0.05,0.02]
```

```
[0.82,0.33,0.64,0.35,0.44,0.62,0.85,0.12]
[0.82,0.33,0.64,0.35,0.44,0.62,0.85,0.12]
[0.82,0.33,0.64,0.35,0.44,0.62,0.85,0.12]
⋮
[0.73,0.13,0.66,0.36,0.41,0.52,0.77,0.02]
```

Fig. 2. Logs processing flow.

Algorithm 1 Generate Low-Dimension Semantic Vector

```
Input: High-dim Semantic Vector, Hyperparameter d
Output: Low-dim Semantic Vector
Require: PCA
Semantic Vector←PCA(High-dim Semantic Vector)
Semantic Vector←Semantic Vector-Average(Semantic Vector)
Semantic Vector←PCA(Semantic Vector)
for v in Semantic Vector do
    Low-dim Semantic Vector←v-$\sum_{n=1}^{d}$(Semantic Vector*v)*Semantic Vector
return Low-dim Semantic Vector
```

vectors is still too heavy to edge devices. For example, a sample log sequence requires 0.46M of memory for processing. Therefore, dimensionality reduction is used to further reduce these semantic vectors for the lightweight log anomaly detection deployment.

Post Processing Algorithm (PPA) [15] is a dimensionality reduction technique that combines with PCA to further construct effective word embeddings. Compared to PCA, it has additional feature embedded from null space of PCA but offers a more concise feature representation. The algorithm is divided into two stages, the first stage reduces the dimension of semantic vectors by PCA, and the second stage processes the processed vectors by PPA to remove the mean vector and dominating directions from the PCA space. The number of removed eigen-directions is determined by the hyperparameter d, which is defaultly set to 7 by following the implementation in [15]. The pseudocode of PPA is presented in Algorithm 1.

3.3. Enhanced TCN

TCN is a one-dimensional fully convolutional network (1D-FCN) which has significant advantage on memory consumption and processing speed via parallel programming. The proposed TCN architecture is illustrated in Fig. 3. In this architecture, three key elements make it outperform other network architectures, e.g., Bi-LSTM and 1D-ResNet, to process the large-scale log anomaly detection task: (i) causal convolution operation; (ii) dilated convolution operation; and (iii) enhanced residual blocks. Causal convolution operation ensures that only the input data before current timestamp t is used to predict the output of the timestamp t. While dilated convolution operation improve feature representations extracted from various receptive fields. In our work, we set three dilated rates, i.e. $d = 1, 2, 4, 8$ respectively. The dilation convolution is calculated as:

$$F(s_t) = (S * F(d))(s_t) = \sum_{i=0}^{k-1} f(i) * s_{t-d*i} \tag{1}$$

Here, d denotes the dilation rate and k denotes the size of the convolution kernel. When $d = 1$, the dilation convolution degenerates to normal convolution.

In the proposed TCN, we enhance the residual blocks to achieve both high accuracy and computational efficiency. On one hand, to enrich low-level feature representations, multiple convolutional kernels are added into residual blocks as illustrated in Fig. 3. This ensures reliable performance of the accurate detection. On the other hand, pointwise-convolution in MobileNet [39] is adopted in our TCN architecture to produce a lightweight model with reduced parameter number. The pointwise-convolution operation is equivalent to a weighted fusion of multiple feature maps from its previous layer to generate a new feature map. As an example illustrated in Fig. 4, the three 1-dimensional feature maps are fused into a new feature map by a 1*3 kernel. Thus, the number of parameters by using our proposed TCN has been effectively reduced to deploy on edge devices.

After extracting feature maps, we use global averaging pooling (GAP) to further reduce parameters in fully connected layers. GAP was proposed in the literature [40] to replace fully connected layers to average the output feature maps of the last convolutional layer for compact and efficient feature representations. With the combination of multi-kernels pointwise convolution operation and GAP, the number of parameters in our model is extremely low, which can be effectively deployed on edge devices. In addition, the architecture avoids overfitting issue to improve the reliability of our model, which is demonstrated in our experimental results.

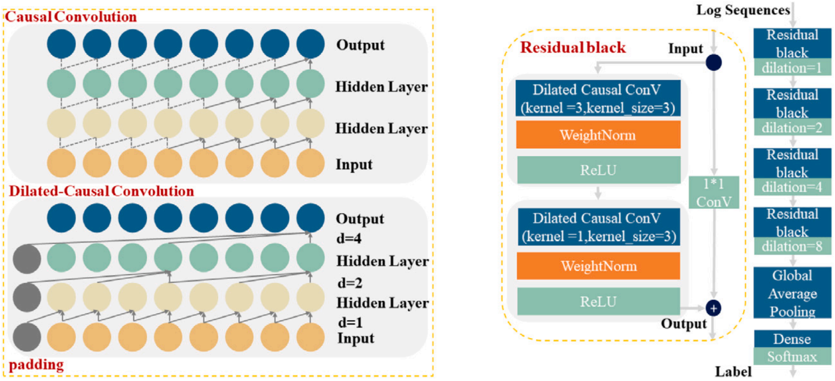


Fig. 3. The proposed TCN architecture has three key elements: causal convolutions, dilated convolutions, and enhanced residual blocks. A residual block of the right image contains the left two layers of dilated causal convolution.

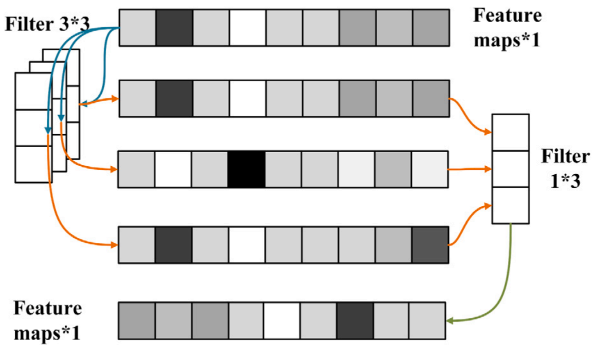


Fig. 4. Pointwise-convolution.

Table 1
Set up of log datasets.

Datasets	Methods	Train set		Test set	
		Normal	Abnormal	Normal	Abnormal
HDFS	Session window	6,000	6,000	490,000	10,000
BGL	Sliding window	3,000	3,000	20,000	3,000

4. Experiment

In this section, we demonstrate the superior performance of the proposed LightLog method in terms of both accuracy and efficiency by comparing with several state-of-the-art log anomaly detection methods. In Section 4.1, we present the experimental setup, including the datasets for the assessment, evaluation metrics, experimental configurations, and implementation details. In Section 4.2, we compare the LighLog with several state-of-the-art methods, including DeepLog [7], LogAnomaly [17] and RobustLog [16], w.r.t. the detection accuracy. In Section 4.3, the efficiency of LightLog is demonstrated to outperform benchmarking methods based on two metrics, FLOPs and Parameter number. Further, in Section 4.4, an ablation study is conducted to show how the proposed elements in the LightLog contribute to its overall performance.

4.1. Experiment setup

We conduct our experiments on the HDFS dataset [21] and BGL dataset [31]which are widely used for log anomaly detection.

The HDFS dataset consists of 11,175,629 logs generated on over 200 Amazon EC2 nodes. As shown in Table 1, these log messages form different log sequences based on session window, including 575,059 normal log sequences and 16,838 abnormal log sequences. We selected 6000 normal log sequences and 6000 abnormal log sequences for training, and randomly collected 500,000 log sequences among the remaining log sequences, including 490,000 normal sequences and 10,000 abnormal sequences for testing. The percentage of abnormal is 2.9%, which is in line with the percentage in the original HDFS dataset and consistent with a realistic scenario.

The BGL data contains of 4,747,963 logs generated by the BlueGene/L supercomputer system, of which 48,460 logs were labelled as malfunctions. Unlike HDFS logs, BGL logs do not record the block ID generated by each log event. therefore, we use a sliding window [41] to divide the log sequence. For more flexible control of detection in real-time, in our experiments, the unit of the sliding window is the number of logs. We set the size of the sliding window to 300 and the step size to 100, and label the log sequence as abnormal if there are any faulty logs in the sequence. As shown in Table 1, a total of 41,457 normal sequences and 6,007 abnormal sequences were obtained by this method. We select 3000 normal samples and 3000 abnormal samples as the training set, and the remaining 3000 abnormal sequences and 20,000 normal sequences as the test set to maintain the ratio of normal samples to abnormal samples in the real data.

Log anomaly detection is a process of binary classification of sequences and the results are classified into the following four types:

- TP: Abnormal sequence samples are correctly detected as abnormal sequence samples
- FN: Abnormal sequence samples are incorrectly detected as normal sequence samples
- TN: Normal sequence samples are correctly detected as normal sequence samples
- FP: Normal sequence samples are misclassified as abnormal sequence samples

We use Precision, Recall and F1-measure as detection effectiveness metrics. Precision is the proportion of all log sequences identified as anomalous that are correctly identified as anomalous. Recall is the proportion of all log sequences that are correctly identified as anomalous, and F1-measure is the summed average of precision and recall. The metrics are defined as follows:

$$Precision = TP/(TP + FP) \tag{2}$$

$$Recall = TP/(TP + FN) \tag{3}$$

$$F1 - measure = (2 * Precision * Recall)/(Precision + Recall) \tag{4}$$

In addition, we used the number of parameters, Floating points of operations (FLOPs), model size and detection time to measure the performance of the method in terms of operational performance. The number of parameters therein is related to the size of the kernels and

Table 2
Number of FPs and FNs on HDFS log.

Methods	FN	FP
DeepLog	451	733
LogAnomaly	372	386
RobustLog	839	**338**
LightLog	**22**	595

the number of input and output channels, and can measure the usage of computational resources such as memory during model training and detection. FLOPs are the number of multiplication and addition operations in the model and are used to measure the computational complexity of the model. The model size provides a more intuitive sense of the number of parameters and the computational resources used in the detection phase, and measures the portability of the approach to edge computing devices. The detection time gives an indication of the method's ability to handle large scale logs.

To complete the testing of the method, the training experiments were performed on a CPU Intel Xeon E5-2620, GPU NVIDA GTX1080ti, 64G of RAM, 11G of video memory, CuDNN 7.6.5, CUDA 11.0, Tensorflow 1.13.1, and Keras 2.2.4. In addition, we chose the Nvidia jetson TX2 module as the edge computing environment in our test experiments. The module has a combination of NVIDIA Denver and Arm® Cortex®-A57 MPCore processors, NVIDIA Pascal™ GPUs, 8G of RAM, Ubuntu 16.04, CuDNN 7.4.5, CUDA 9.0, Tensorflow 1.8.0, and Keras 2.1.6.

Since the training task is carried out as binary classification, the loss was calculated based on binary cross entropy and the LightLog was trained by using the Adam optimizer. We use an initial learning rate of 0.001 and learning rate decay rate of 0.98 per 20 epochs. In addition, we utilized Softmax as the activation function for the output layer and set the training epoch to 100.

4.2. Performance comparison on accuracy

Three state-of-the-art log anomaly detection methods, including DeepLog, LogAnomaly, and RobustLog, are selected as benchmark to evaluate the performance of LightLog. All of these methods are RNN based algorithms with a two-layer stacking structure. Table 1 shows the number of samples misclassified for each method on the HDFS test set. The LightLog method with semantic vector dimensionality reduction achieves the best performance in terms of false negative rate. LogAnomaly was higher than the LightLog method in detecting normal samples, but showed more false detections in classifying abnormal logs. Although the RobustLog method has the least number of false positives, it misses many abnormous logs. Since LightLog has a different sliding window unit than the other three methods on the BGL dataset, we present the experimental results on a more intuitive evaluation metric.

Furthermore, we illustrate the performance of the LightLog in terms of precision, recall and F1-measure in Fig. 5. On HDFS test set, LightLog achieves the best performance. According to the F1-measure, the LightLog has 97.0% which is higher than the other three methods on 94.16%, 96.21% and 93.96% respectively. Although LightLog has a lower precision of 94.37% compared to the RobustLog method on 96.64% and the LogAnomaly on 96.1% . Both RobustLog and LogAnomaly have much higher false detection rates on abnormal log sequences. Thus, the LightLog has a reliable detection performance although the model is much lighter compared to the benchmarking models. On the BGL test set, LightLog achieves the highest accuracy rate, with a detection rate of 99.18% for abnormal samples. DeepLog performs slightly better than LightLog in terms of precision, but only achieves 91.18% in the detection of abnormal samples, which is far behind LightLog in overall.

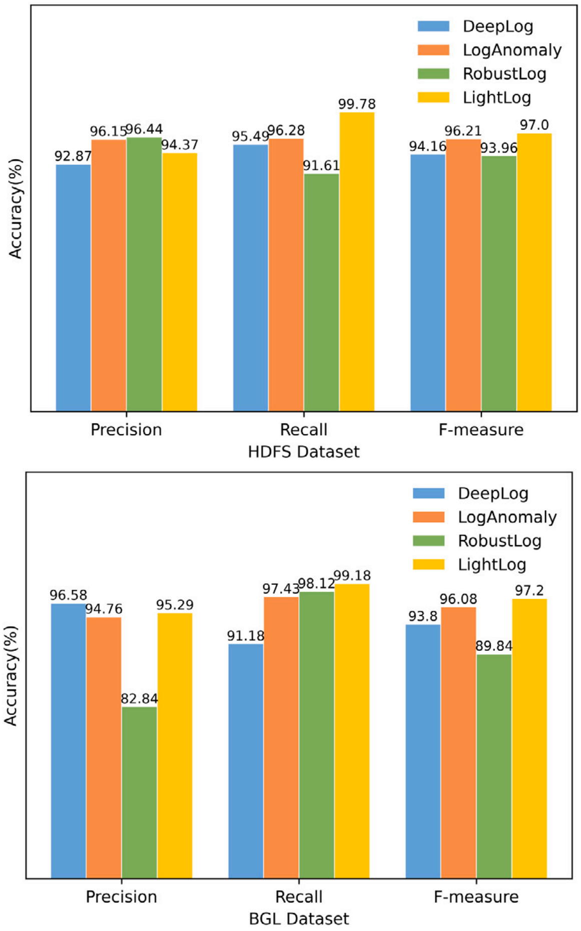


Fig. 5. Comparison with different methods.

Table 3
FLOPs, params and model size in different methods.

Methods	FLOPs	Params	Model size (KB)	Test time (min)	
				HDFS	BGL
DeepLog	53,248	52,252	618	224	25.7
LogAnomaly	106,496	104,476	1,239	860	96.3
RobustLog	354,432	352,385	4,138	13.5	1.7
LightLog	**1,408**	**544**	**139**	**4.3**	**0.45**

4.3. Performance in computation efficiency

Clearly, LightLog has a definite advantage in a number of evaluation criteria, such as floating point operations, number of parameters and model size. As shown in Table 2, its floating point operations are only 1% of the RobustLog method. In terms of number of parameters, LightLog only defines 544 parameters, which is close to 0.19% of the number of parameters in the RobustLog method. Both of these advantages are a good indication that the method in the paper requires much less computing resource and memory than the other three methods. In addition, the model trained by the LightLog method is only 139 KB. This model can be effectively ported to all types of terminal equipment with limited computing resources, including IoT edge computing devices.

As shown in the framework of the approach in Fig. 1, the model needs to cope with large-scale logs on edge computing devices. Difficult to determine factors, such as the number of calls to external storage, also affect detection performance. Therefore, the detection time of logs is also a very important measure. We perform 100 detections on different test sets. As shown in Table 2, the DeepLog, LogAnomaly and

Table 4
Comparison of TCN networks with different structures.

Algorithms	HDFS		BGL		FLOPs	Params	Model size (KB)
	FN	FP	FN	FP			
TCN	347	824	196	651	5,132	2,464	160
TCN (PCA-PPA)	325	712	172	376	1,772	784	142
TCN (with 3-kernels)	31	656	35	155	5,388	2,474	160
TCN (with 3-kernels and pw-conv)	47	**471**	**21**	212	4,996	2,338	160
Proposed TCN	**22**	595	25	**147**	**1,408**	**544**	**139**

RobustLog methods built with RNNs have higher detection times than the LightLog method. On HDFS dataset, the average detection time of LightLog is 4.3 min, much lower than the 860 min of LogAnomaly. It is clear that the LightLog approach is able to process large-scale logs in real-time on edge computing devices.

4.4. Ablation study on TCN

In this ablation study, we discuss how each designed component in the LightLog model contributes to the overall performance. As shown in Table 4, the Mapping built from the dimensionality-reduced semantic vectors using PCA-PPA is slightly higher in detection accuracy than the Mapping built from the original semantic vectors extracted by using Word2Vec. At the same time, the use of the proposed feature provides a more compact representation at the deployment stage in terms of FLOPs and number of parameters.

Regarding to the improvement of the proposed TCN model, it is found that both the multiple-kernel and the GAP provide a significant margin on computational resource consumption, with the number of parameters being only 1/4 of the original TCN. Therefore, by comparing the performance of different components in the LightLog, it is proven that both the feature representation, i.e., PCA-PPA and the improved TCN model, make contributions to reduce the LightLog model size when achieving impressive detection accuracy. The LightLog, therefore, can be easily deployed on edge computing devices with limited computational resources.

5. Conclusion

In this paper, a log anomaly detection method based on temporal convolutional networks is designed. The method uses the PCA-PPA algorithm to reduce the dimensionality of the semantic vector formed by the log template, and improves the TCN by embedding multi-core point-by-point convolution and global average pooling to build the lightweight detection model. Experimental results show that our approach achieves significant advantages in terms of arithmetic power, memory and high availability, while ensuring detection accuracy. In the field of IoT and edge computing, our method is of high value to enhance their security and reliability.

One limitation of the present study is that the model was trained to achieve the best overall performance, which is the standard way to evaluate model performance. While in application level, the false negative rate may become more important when manual checks are involved. In our future work, we will make investigations on the tolerant level of false alarm so that a better trade-off can be achieved.

CRediT authorship contribution statement

Zumin Wang: Project administration, Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Funding acquisition. **Jiyu Tian:** Conceptualization, Methodology, Formal analysis, Software, Validation, Data curation, Visualization, Investigation, Writing – original draft, Resources, Writing – review & editing. **Hui Fang:** Methodology, Formal analysis, Writing – original draft, Writing – review & editing. **Liming Chen:** Formal analysis, Writing – original draft, Writing – review & editing. **Jing Qin:** Formal analysis, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Youth Fund Project of the National Natural Science Foundation of China under grant 62002038.

References

[1] I. Butun, P. Österberg, H. Song, Security of the internet of things: Vulnerabilities, attacks, and countermeasures, IEEE Commun. Surv. Tutor. 22 (1) (2019) 616–644.

[2] A. Mudgerikar, P. Sharma, E. Bertino, E-Spion - A system-level intrusion detection system for IoT devices, in: AsiaCCS, 2019, pp. 493–500.

[3] D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, R. Li, LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial IoT, IEEE Trans. Ind. Inf. 16 (8) (2020) 5244–5253.

[4] X. Xie, Z. Jin, J. Wang, L. Yang, Y. Lu, T. Li, Confidence guided anomaly detection model for anti-concept drift in dynamic logs, J. Netw. Comput. Appl. 162 (2020) 102659.

[5] J. Wang, Y. Tang, S. He, C. Zhao, K.P. Sharma, O. Alfarraj, A. Tolba, LogEvent2vec: LogEvent-to-vector based anomaly detection for large-scale logs in internet of things, SENSORS 20 (9) (2020).

[6] A. Borghesi, A. Bartolini, M. Lombardi, M. Milano, L. Benini, Anomaly detection using autoencoders in high performance computing systems, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol.33, 2019, pp. 9428–9433.

[7] M. Du, F. Li, G. Zheng, V. Srikumar, DeepLog: Anomaly detection and diagnosis from system logs through deep learning, in: CCS, 2017, pp. 1285–1298.

[8] S. Raza, L. Wallgren, T. Voigt, SVELTE: Real-time intrusion detection in the internet of things, Ad Hoc Netw. (2013) 2661–2674.

[9] A. Libri, A. Bartolini, L. Benini, pAElla: Edge AI-based real-time malware detection in data centers, IEEE Internet Things J. 7 (10) (2020) 9589–9599.

[10] X. Wang, Y. Zhao, H. Xiao, X. Chi, X. Wang, Multi-node system abnormal log flow mode detection method, J. Softw. (2020) 3295–3308.

[11] P. He, J. Zhu, S. He, J. Li, R.M. Lyu, Towards automated log parsing for large-scale log data analysis, IEEE Trans. Dependable Secure Comput. (2018) 931–944.

[12] A. Oprea, Z. Li, T.-F. Yen, H.S. Chin, A.S. Alrwais, Detection of early-stage enterprise infection by mining large-scale log data, in: DSN '15 Proceedings of the 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2015, pp. 45–56.

[13] D. Breitenbacher, I. Homoliak, L.Y. Aung, O.N. Tippenhauer, Y. Elovici, HADES-IoT - A practical host-based anomaly detection system for IoT devices, in: AsiaCCS, 2019, pp. 479–484.

[14] W. Ling, C. Dyer, W.A. Black, I. Trancoso, Two/Too simple adaptations of Word2Vec for syntax problems, in: HLT-NAACL, 2015, pp. 1299–1304.

[15] V. Raunak, Effective dimensionality reduction for word embeddings, in: RepL4NLP@ACL, 2019.

[16] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, D. Zhang, Robust log-based anomaly detection on unstable log data, in: ESEC/SIGSOFT FSE, 2019, pp. 807–817.

[17] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, R. Zhou, LogAnomaly - Unsupervised detection of sequential and quantitative anomalies in unstructured logs, in: IJCAI, 2019, pp. 4739–4745.

[18] W. Meng, Y. Liu, Y. Huang, S. Zhang, F. Zaiter, B. Chen, D. Pei, A semantic-aware representation framework for online log analysis, in: 2020 29th International Conference on Computer Communications and Networks ICCCN, 2020, pp. 1–7.

[19] X. Li, P. Chen, L. Jing, Z. He, G. Yu, Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults, in: 2020 IEEE 31st International Symposium on Software Reliability Engineering, ISSRE, IEEE, 2020, pp. 92–103.

[20] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, Z. Luan, HitAnomaly: Hierarchical transformers for anomaly detection in system log, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 2064–2076.

[21] W. Xu, L. Huang, A. Fox, D. Patterson, M. Jordan, Largescale system problem detection by mining console logs, in: Proceedings of SOSP'09, 2009.

[22] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, J. Li, Mining invariants from console logs for system problem detection, in: USENIX Annual Technical Conference, 2010, pp. 24–24.

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

[24] A. Borghesi, A. Libri, L. Benini, A. Bartolini, Online anomaly detection in HPC systems, in: 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems AICAS, 2019, pp. 229–233.

[25] V.D.A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, W.A. Senior, K. Kavukcuoglu, WaveNet: A generative model for raw audio, in: SSW, 2016.

[26] S. Bai, Z.J. Kolter, V. Koltun, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, in: ArXiv: Learning, 2018.

[27] J. Yan, L. Mu, L. Wang, R. Ranjan, Y.A. Zomaya, Temporal convolutional networks for the advance prediction of ENSO, Sci. Rep. (2020) 8055.

[28] Y. Chen, Y. Kang, Y. Chen, Z. Wang, Probabilistic forecasting with temporal convolutional neural network, Neurocomputing (2020) 491–501.

[29] Y. He, J. Zhao, Temporal convolutional networks for anomaly detection in time series, J. Phys. Conf. Ser. 1213 (2019) 042050.

[30] Y. Cheng, Y. Xu, H. Zhong, Y. Liu, HS-TCN - A semi-supervised hierarchical stacking temporal convolutional network for anomaly detection in IoT, in: IPCCC, 2019, pp. 1–7.

[31] A. Oliner, J. Stearley, What supercomputers say: A study of five system logs, in: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks DSN'07, 2007, pp. 575–584.

[32] W. Meng, Y. Liu, Y. Huang, S. Zhang, F. Zaiter, B. Chen, D. Pei, A semantic-aware representation framework for online log analysis, in: 2020 29th International Conference on Computer Communications and Networks ICCCN, 2020, pp. 1–7.

[33] P. He, J. Zhu, S. He, J. Li, R.M. Lyu, An evaluation study on log parsing and its use in log mining, DSN (2016) 654–661.

[34] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, T. Mikolov, FastText.zip: Compressing text classification models, in: ArXiv: Computation and Language, 2017.

[35] T. Mikolov, K. Chen, G.S. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: ICLR, 2013.

[36] J. Pennington, R. Socher, D.C. Manning, Glove: Global vectors for word representation, in: EMNLP, 2014, pp. 1532–1543.

[37] G. Martin, word2vec, node2vec, graph2vec, X2vec: Towards a theory of vector embeddings of structured data, in: SIGMOD/PODS '20: International Conference on Management of Data Portland OR USA June, 2020, 2020, pp. 1–16.

[38] S. Qaiser, R. Ali, Text mining: Use of TF-IDF to examine the relevance of words to documents, Int. J. Comput. Appl. (2018) 25–29.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[40] M. Lin, Q. Chen, S. Yan, Network in network, in: ICLR, 2014.

[41] S. He, J. Zhu, P. He, M.R. Lyu, Experience report: System log analysis for anomaly detection, in: ISSRE, 2016, pp.207–218.

Zumin Wang is a professor at Dalian University since 2014. His research interests include Sensors and Sensor Application, Wireless Sensor Networks, and Smart City. He received his MS degree in Mechanical Manufacturing and Automation in 2004 from North University of China, and Ph.D. degree in Physical Electronics in 2007 from the Chinese Academy of Sciences. He was a visiting scholar at the University of Washington from Nov. 2016 to Nov. 2017. He is a distinguished member of CCF.

Jiyu Tian is pursuing a master's degree in cyberspace security at Dalian University, Dalian, China. His research interests include web attack and defence, anomaly detection, deep learning, and industrial control systems.

Hui Fang received the B.S. degree from the University of Science and Technology, Beijing, China, in 2000 and the Ph.D. degree from the University of Bradford, U.K., in 2006. He is currently with the Computer Science Department, Loughborough University (LU). Before joined LU, he has carried out research in several world-leading universities, such as University of Oxford and Swansea University. His research interests include computer vision, image/video processing, pattern recognition, machine learning, data mining, scientific visualization, visual analytics, and artificial intelligence. Recently, he was awarded several grants as PI and co-PI, including Innovate UK funded "An agent-based modelling solution for reliable decision making in crisis and market turmoil in consumer retail", EPSRC funded "RAMP VIS: Making Visual Analytics an Integral Part of the Technological Infrastructure for Combating COVID-19", and NIHR funded "Computer vision to automatically monitor urine output". During his career, he has published more than 60 journal and conference papers.

Liming Luke Chen is Professor of Data Analytics in the School of Computing, Ulster University, UK. He received his BEng and MEng degrees at Beijing Institute of Technology, China, and DPhil on Computer Science at De Montfort University, UK. His current research interests include pervasive computing, data analytics, artificial intelligence, user-centred intelligent systems and their applications in smart healthcare and cyber security. He has published over 250 papers in the aforementioned areas. Liming is an IET Fellow and a Senior Member of IEEE.

Jing Qin received the Ph.D. degree in the School of Computer Science and Technology, Dalian University of Technology, Dalian, China. She is an associate professor in the School of Software Engineering, Dalian University. Her research interests include multimedia information retrieval, signal processing and machine learning.