ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10094961

ROBUST LOG-BASED ANOMALY DETECTION WITH HIERARCHICAL CONTRASTIVE LEARNING

Yuhui Zhao[†] *Ruichun Yang*[‡] *Ning Yang*[†*] *Tao Lin*[†*] *Qiuai Fu*[⋆] *Yuchi Ma*[⋆]

[†] School of Computer Science, Sichuan University
[‡] The Chinese University of Hong Kong, Shenzhen
[⋆] Huawei Cloud Computing Technologies CO., LTD.

ABSTRACT

Logs are widely employed in modern systems to record critical information and serve as an important source for anomaly detection, which has attracted increasing research interests. However, logs usually suffer from perturbations and it makes the existing log-based anomaly detection methods unstable. In this paper, we aim to solve this problem from the perspective of contrastive learning, by which the intrinsic and robust representations of logs are learned for anomaly detection. We propose two data augmentation methods to generate different views at different granularity for log data and design a deep hierarchical contrastive model for anomaly detection. In the contrastive semantic embedding module, we fine-tune a language model with a message-level contrastive loss. And in the contrastive anomaly detection module, we apply a sequence-level contrastive constraint to assist the detection model to learn robust embeddings for log sequences. Experiments on three datasets verify the effectiveness of our proposed method.

***Index Terms*—** Logs, Anomaly Detection, Contrastive Learning

1. INTRODUCTION

Modern software-intensive systems are becoming much more large-scale and complex, which makes anomaly detection an indispensable task to help maintain reliability and stability. These systems usually produce copious logs, and they are typically present in the form of semi-structured natural languages, where critical information describing current circumstances is wrote down. When a failure occurs, the operators can refer to the logs to track the suspicious events, analyze the underlying causality and find the possible root cause [1, 2].

Most of the existing approaches [3, 4, 5, 6] for log-based anomaly detection make an assumption explicitly or implicitly that logs are generated by the system following a predefined fixed paradigm [7]. That is, the log statements usually keep stable and the logs are regularly printed along with the execution flows. However, this assumption cannot always be satisfied due to the challenges from two aspects:

Perturbations in the log messages According to a previous empirical study [8, 7], the percentage of the ever-changed log statements is about $20\% \sim 45\%$. These changes can be induced by the updates of the log templates. For example, words can get inserted or appended in the description statements to make the logs more comprehensive, or some words may be replaced or removed with the business adjustments.

Perturbations in the log sequences As modern systems are becoming much more sophisticated, it often entails different modules or services collaborating at the same time. And the log sequences produced by different modules are interleaved in a common log file. The perturbation in the log sequence can be caused by the delay or advance of sub-tasks in some modules. The existing works which model the log sequence to discover the latent sequential patterns will flag an anomaly for such situation. However, it may bring about false alarms if these tasks are not related at all.

In this paper, we propose logContrast, a robust log-based anomaly detection model based on contrastive learning, aiming to learn the intrinsical representations of logs and log sequences for the anomaly detection task. Firstly, inspired by the contrastive framework [9], we propose two methods of data augmentations to generate multiple views of different granularity, i.e., log-level and sequence-level. Specifically, to handle the perturbations in log messages, we augment the individual logs using three types of operations. To address the perturbations in the log sequences, we also propose to augment the log sequences in three ways, random repetition, random deletion, and random shuffling. Then we design a contrastive deep neural network to model the log sequence characteristics to determine whether a log sequence is an anomaly. We evaluate logContrast on three public real-world log datasets. The results demonstrate its effectiveness and robustness in the anomaly detection task.

2. BACKGROUND AND PRELIMINARIES

Logs are generated by large-scale systems to record critical

[*] Ning Yang and Tao Lin share the corresponding authorship.

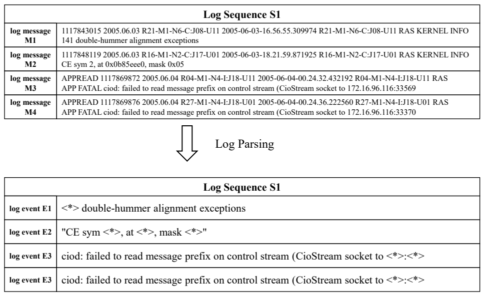
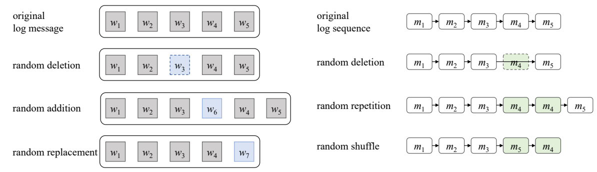


Fig. 1: Illustration of logs and log parsing



(a) Augmentation in log-level. (b) Augmentation in seq-level.

Fig. 3: Illustration of two kinds of augmentation.

information of runtime, which includes but is not limited to timestamps, verbosity level, events, and parameters of execution details. As shown in Fig.1, a raw *log message* is made up of a series of tokens and can be seen as a piece of semi-structured text where certain information is arranged, following a predefined way. Then a *log sequence* is a series of log messages that belong to a same-time interval or are denoted by a common task identifier.

Log parsing aims to separate raw log messages into a constant part and a variable part [10, 11] in a meaningful manner. The constant part contains a group of keywords that represent the *template* of a log message, which is also called *log event*, and the variable part is comprised of log parameters recording some attribute information like IP address. The bottom part of Fig.1 shows an example of the parsing results of logs from the upper part. It can be seen that the parameters are removed and only the key tokens are retained. Besides, it can also be found that different log messages can have the same log events after log parsing, such as log messages M3 and M4. Traditional log-based anomaly detection methods such as PCA [12], Invariants Mining [13] and some deep methods like DeepLog [14] and logAnomaly [15] require log parsing to obtain log templates/log events as a preprocessing step. In recent decades, log parsing has been widely investigated and a lot of log parsing approaches are proposed from different aspects [11, 16], and people can refer to [10] for a comprehensive survey.

3. PROPOSED METHOD

3.1. Preprocess Module

The preprocessing procedure mainly consists of three steps.

Tokenization We tokenize each raw log message into a list of tokens with three steps inspired by [17]. First, a log message is split into candidate tokens with common delimiters like white space or punctuation. Then all the tokens containing number digits or special symbols are removed, as they usually are non-informative parameters and notations. Finally, we replace each capital letters in the remaining tokens with corresponding lowercase letters. We illustrate this procedure with an log message from a dataset BGL [18]. With tokenization, a raw log message ″1117842974 2005.06.03 R24-M0-N1-C:J13-U11 2005-06-03-16.56.14.254137 R24-M0-N1-C:J13-U11 RAS KERNEL INFO 162 double-hummer alignment exceptions″ is transformed into token list ″ras, kernel, info, double, hummer, aligment, exceptions″.

Log Augmentation in log-level To learn robust semantic embeddings for log messages, we apply contrastive learning on the vectorization process, which requires log data from different views. As shown in Fig.3 (a), three data augmentation techniques are introduced to generate different views of log messages, i.e., random deletion, random addition, and random replacement, where the items in the token list are changed randomly.

Log Augmentation in sequence-level We also generate different views for contrastive learning at the granularity of log sequence, as shown in Fig.3 (b). The first is random deletion of log messages from the original log sequences. The second is random repetition, for which we randomly select a log message and repeat it several times in its belonging sequence. The third is random shuffle, that is, we randomly select a segment of the log sequence and shuffle the log messages in it while the other log messages maintain stable.

3.2. Contrastive Semantic Embedding Module

3.2.1. Log Message Encoding

A log message can be considered as a sentence comprising a line of words, and we adopt the language model to extract the semantic embedding from it. As a powerful deep representation model, BERT has been pre-trained on a huge natural language corpus and has shown its great learning ability. In this paper, we follow the practice in [17] and employ the BERT base model [19] for log message encoding. We use the average of all token embeddings in a log message as the embedding of the log message. Given $i\text{-}th$ log message x_i, it is encoded using BERT as $\mathbf{x}_i$, which is $\mathbf{x}_i = \mathrm{BERT}(x_i)$.

3.2.2. Fine-tune BERT with Contrastive Learning

To make the semantic embedding more robust, we fine-tune the BERT model through a contrastive loss computed on the embeddings of log messages from different views. The em-

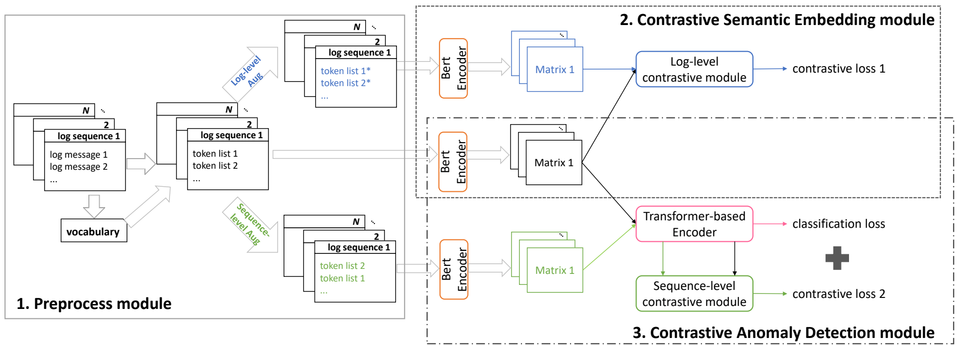


Fig. 2: The framework of the proposed logContrast.

beddings of the related log messages are pulled closer and the embeddings of the unrelated log messages are pushed much further. Given a set of M log message $x_1^v, ..., x_i^v, .., x_M^v$, we denote its corresponds embeddings extracted by BERT as $\mathbf{x}_1^v, ..., \mathbf{x}_i^v, .., \mathbf{x}_M^v$, where $v \in \{or, rd, ra, rr\}$ represents the views in log-level and $\{or, rd, ra, rr\}$ represents different log augmentations of **or**iginal log message, **r**andom **d**eletion, **r**andom **a**ddition and **r**andom **r**eplacement respectively. Then we define the total contrastive loss for all given log message x_i^v between view v_a and v_b as follow:

$$\mathcal{L}_{v_a,v_b} = -\mathop{\mathbb{E}}_{i}\left[\log \frac{\exp\left(\frac{\theta(\mathbf{x}_i^{v_a}, \mathbf{x}_i^{v_b})}{\tau}\right)}{\exp\left(\frac{\theta(\mathbf{x}_i^{v_a}, \mathbf{x}_i^{v_b})}{\tau}\right) + \sum_{j=1}^{\mathcal{N}_1} \exp\left(\frac{\theta(\mathbf{x}_i^{v_a}, \mathbf{x}_j^{v_b})}{\tau}\right)}\right], \tag{1}$$

where $v_a, v_b \in \{or, rd, ra, rr\}$. Function $\theta(*, *)$ computes the similarity between two vectors and τ is the temperature parameter. And for each log message, we sample $\mathcal{N}_1$ negative samples from the contrasting view. Then we fix the view *or* as the core view and compute the contrastive loss between it and the others, which leads to the total contrastive loss for the contrastive semantic embedding module defined as follows:

$$\mathcal{L}_{cl} = \sum_{v_b \in \{rd, ra, rr\}} \mathcal{L}_{or, v_b}. \tag{2}$$

3.3. Contrastive Anomaly Detection Module

3.3.1. Log Sequence Encoding

We employ the classic transformer [20] to encode a sequence of log messages. Given a sequence of log message $S_i = \{x_1, ..., x_{|S_i|}\}$, it is first input in the fine-tuned BERT encoder to get the semantic embeddings $\{\mathbf{x}_1, ..., \mathbf{x}_{|S_i|}\}$. To capture the context information of a log message in the sequence, we also add positional embeddings $\{\mathbf{p}_1, ..., \mathbf{p}_{|S_i|}\}$ before putting them into the transformer layers. The procedure can be defined as $\mathbf{s}_i = \mathrm{Transformer}(\mathbf{x}_1 + \mathbf{p}_1, ..., \mathbf{x}_{|S_i|} + \mathbf{p}_{|S_i|})$.

Anomaly detection can be regarded as a classification problem with two target classes, anomaly and normal. With log sequence embeddings $\{\mathbf{s_i}\}$, a softmax layer is appended to compute probability $p_{i,f}$ of each log sequence S_i on each class f. We denote the set of log sequences with labels as $\mathcal{S}_L$ and use the cross-entropy loss to construct the classification objective as follows:

$$\mathcal{L}_d = -\sum_{S_i \in \mathcal{S}_L} \sum_{f \in F} Y_{i,f} \ln p_{i,f}, \tag{3}$$

where $Y_{i,f} = 1$ denotes that log sequencce S_i actually belongs to class f, and otherwise $Y_{i,f} = 0$.

3.3.2. Sequence-level Constraint

To handle the perturbations in the log sequences, we also implement a contrastive constraint at the sequence level. Given a series of log sequence $S_1^u, ..., S_i^u, .., S_N^u$, where $u \in \{OS, RD, RR, RS\}$ corresponding to the sequence-level view of original log sequence, random deletion, random repetition and random shuffle respectively, their embedding by transformer are $\mathbf{s}_1^u, ..., \mathbf{s}_i^u, .., \mathbf{s}_N^u$. Similarly, we define the loss of the sequence-level contrastive loss as follows:

$$\mathcal{L}_{u_s,u_t} = -\mathop{\mathbb{E}}_{i}\left[\log \frac{\exp\left(\frac{\theta(\mathbf{s}_i^{u_s}, \mathbf{s}_i^{u_t})}{\tau}\right)}{\exp\left(\frac{\theta(\mathbf{s}_i^{u_s}, \mathbf{s}_i^{u_t})}{\tau}\right) + \sum_{j=1}^{\mathcal{N}_2} \exp\left(\frac{\theta(\mathbf{s}_i^{u_s}, \mathbf{s}_j^{u_t})}{\tau}\right)}\right], \tag{4}$$

$$\mathcal{L}_{cs} = \sum_{u_t \in \{RD, RR, RS\}} \mathcal{L}_{OS, u_t}, \tag{5}$$

where $\mathcal{N}_2$ is the number of negative samples. And the final total loss for the contrastive anomaly detection module is $\mathcal{L} = \mathcal{L}_d + \mathcal{L}_{cs}$.

4. EXPERIMENTS

4.1. Datasets and Baselines

We choose three representative log datasets HDFS, BGL, and Thunderbird for evaluation from LogHub [10], as shown in

Table 2: Performance comparison anomaly detection on original datasets.

Methods	HDFS			BGL			Thunderbird		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
PCA	0.9710	0.6280	0.7630	0.9412	0.1538	0.2645	0.9880	0.3200	0.4835
IM	0.8950	1.0000	0.9440	0.4177	0.6279	0.5017	0.9992	0.6316	0.7739
DeepLog	0.9650	0.9040	0.9450	0.3488	0.8333	0.4918	0.7512	1.0000	0.6015
LogRobust	0.9340	0.9950	0.9640	0.7250	0.9886	0.8365	0.5170	0.9984	0.6812
Logsy	0.8600	1.0000	0.9250	0.6797	0.9886	0.8056	0.5273	0.9862	0.6872
LogContrast	0.9798	1.0000	**0.9898**	0.8482	0.8714	**0.8596**	0.9971	0.9992	**0.9981**

Table 3: Performance comparison anomaly detection on noise datasets.

Methods	HDFS			BGL			Thunderbird		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
PCA	0.9586	0.6063	0.7426	0.4310	0.0519	0.0926	1.0000	0.0056	0.0111
IM	0.0146	1.0000	0.0288	0.4129	0.5560	0.4739	0.9992	0.6316	0.7739
DeepLog	0.0078	0.3829	0.0153	0.1034	1.0000	0.1874	0.9825	1.0000	0.9912
LogRobust	0.2764	0.1418	0.1874	0.8951	0.3001	0.4503	0.9945	0.9756	0.9850
Logsy	0.0340	0.1936	0.0578	0.1055	1.0000	0.1909	0.9835	1.0000	0.9917
LogContrast	0.9125	0.8670	**0.8892**	0.7934	0.8087	**0.8010**	0.9985	1.0000	**0.9993**

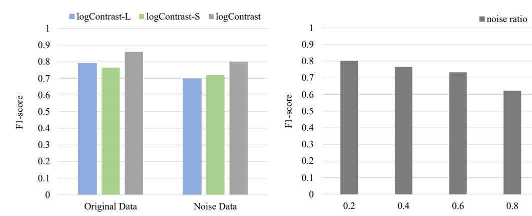
Dataset	Description	#Messages
HDFS	Hadoop distributed file system log	11,175,629
BGL	Blue Gene/L supercomputer log	4,747,963
Thunderbird	Thunderbird supercomputer log	5,000,000

Table 1: The details of datasets.

Table 1. For HDFS, we use the identifiers to group log messages into log sequences, and for BGL and Thunderbird, we slice them based on certain window sizes. To verify the robustness, we also construct synthetic datasets following the practice in [7] and randomly add some noise into the original datasets. The noise ratio is set as 20% for general comparison, and we also investigate the influence of different noise ratio settings. The baseline methods include two classic traditional methods, PCA [12], IM [13], and three deep learning methods, DeepLog [14], LogRobust [7] , Logsy [21].

4.2. Overall Performance

Table 2 gives the anomaly detection results on the original non-noise datasets. Note that for all baselines on original HDFS datasets, we directly report the results from a comprehensive survey [22] without rerunning. As we can see, proposed logContrast constantly gives the best performance on all three datasets. It demonstrates that even for non-noise circumstance, applying the contrastive constraints can benefit the log-based anomaly detection task. Table 3 shows the results on the synthetic data where noise are added randomly. Mostly methods suffer from a drastic drop when confronting perturbations, however, the proposed logContrast still holds a pretty good performance. That is because compared with other method, logContrast can learn more intrinsic embeddings for logs and be more robust to perturbations.



(a) Ablation Study. (b) Noise Ratio Analysis.

Fig. 4: Results of Ablation Study and Parameter Analysis.

4.3. Ablation Study and Parameter Analysis

We denote the variant with only log-level contrastive learning as logContrast-L and the variant with only sequence-level as logContrast-S, and apply them on the BGL respectively. As shown in Fig.4 (a) , both modules play important role in robust anomaly detection. We also experiment on the noise BGL dataset with different noise ratio, the results of which are shown in Fig.4 (b). With the increase in noise ratio, the performance shows a trend of decline.

5. CONCLUSION

In this paper we propose logContrast for robust log-based anomaly detection. To handle the perturbations in log data, we design different data augmentation methods for logs and learn robust representations with a designed deep hierarchical contrastive model. Experiments on multiple datasets exhibit the effectiveness and robustness of the logContrast.

6. ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China under grant 61972270.

7. REFERENCES

[1] Jieming Zhu, Pinjia He, et al., “Learning to log: Helping developers make informed logging decisions,” in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, vol. 1, pp. 415–425.

[2] Shilin He, Jieming Zhu, et al., “Experience report: System log analysis for anomaly detection,” in *2016 IEEE 27th international symposium on software reliability engineering (ISSRE)*, 2016, pp. 207–218.

[3] Qingwei Lin, Hongyu Zhang, et al., “Log clustering based problem identification for online service systems,” in *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*, 2016, pp. 102–111.

[4] Jakub Breier and Jana Branišová, “Anomaly detection from log files using data mining techniques,” in *Information Science and Applications*, pp. 449–457. 2015.

[5] Mostafa Farshchi, Jean-Guy Schneider, et al., “Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis,” in *2015 IEEE 26th international symposium on software reliability engineering (ISSRE)*, 2015, pp. 24–34.

[6] Nengwen Zhao, Honglin Wang, et al., “An empirical investigation of practical log anomaly detection for online service systems,” in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1404–1415.

[7] Xu Zhang, Yong Xu, et al., “Robust log-based anomaly detection on unstable log data,” in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 807–817.

[8] Suhas Kabinna, Cor-Paul Bezemer, et al., “Examining the stability of logging statements,” *Empirical Software Engineering*, vol. 23, no. 1, pp. 290–333, 2018.

[9] Ting Chen, Simon Kornblith, et al., “A simple framework for contrastive learning of visual representations,” in *International conference on machine learning*, 2020, pp. 1597–1607.

[10] Jieming Zhu, Shilin He, et al., “Tools and benchmarks for automated log parsing,” in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 2019, pp. 121–130.

[11] Pinjia He, Jieming Zhu, et al., “Drain: An online log parsing approach with fixed depth tree,” in *2017 IEEE international conference on web services (ICWS)*, 2017, pp. 33–40.

[12] Wei Xu, Ling Huang, et al., “Detecting large-scale system problems by mining console logs,” in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 117–132.

[13] Jian-Guang Lou, Qiang Fu, et al., “Mining invariants from console logs for system problem detection,” in *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[14] Min Du, Feifei Li, et al., “Deeplog: Anomaly detection and diagnosis from system logs through deep learning,” in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[15] Weibin Meng, Ying Liu, et al., “Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs.,” in *IJCAI*, 2019, pp. 4739–4745.

[16] Min Du and Feifei Li, “Spell: Streaming parsing of system event logs,” in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 859–864.

[17] Van-Hoang Le and Hongyu Zhang, “Log-based anomaly detection without log parsing,” in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 492–504.

[18] Adam Oliner and Jon 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.

[19] Jacob Devlin, Ming-Wei Chang, et al., “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[20] Ashish Vaswani, Noam Shazeer, et al., “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30.

[21] Sasho Nedelkoski, Jasmin Bogatinovski, et al., “Self-attentive classification-based anomaly detection in unstructured logs,” in *2020 IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 1196–1201.

[22] Zhuangbin Chen, Jinyang Liu, et al., “Experience report: deep learning-based system log analysis for anomaly detection,” *arXiv preprint arXiv:2107.05908*, 2021.