Graph neural networks based log anomaly detection and explanation

Li, Z.; Shi, J.; Leeuwen, M. van

Graph Neural Networks based Log Anomaly Detection and Explanation

Zhong Li, Jiayang Shi, Matthijs van Leeuwen
LIACS, Leiden University
Leiden, the Netherlands
{z.li,j.shi,m.van.leeuwen}@liacs.leidenuniv.nl

ABSTRACT

Event logs are widely used to record the status of high-tech systems, making log anomaly detection important for monitoring those systems. We propose a graph-based method for unsupervised log anomaly detection, dubbed *Logs2Graphs*, which first converts event logs into attributed, directed, and weighted graphs, and then leverages graph neural networks to perform graph-level anomaly detection. Specifically, we introduce OCDiGCN, a novel graph neural network model for detecting graph-level anomalies in a collection of attributed, directed, and weighted graphs. By coupling the graph representation and anomaly detection steps, OCDiGCN can learn a representation that is especially suited for anomaly detection, resulting in a high detection accuracy. For each detected anomaly, we provide a subset of nodes that are crucial in OCDiGCN's predictions, offering useful insights for root cause diagnosis. Experiments on five benchmark datasets show that *Logs2Graphs* matches or exceeds current top log anomaly detection methods on simple datasets and largely outperforms them on complex ones.

ACM Reference Format:
Zhong Li, Jiayang Shi, Matthijs van Leeuwen. 2024. Graph Neural Networks based Log Anomaly Detection and Explanation. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3639478.3643084

1 INTRODUCTION

Existing log anomaly detection methods can be roughly divided into three categories: quantitative-based, sequence-based, and graph-based methods. Specifically, quantitative-based methods, such as OCSVM [9] and PCA [12], utilise a log event count matrix to detect anomalies, and are therefore unable to capture semantic information of and sequential information between log events. Meanwhile, sequence-based methods, including DeepLog [2] and LogAnomaly [8], aim to detect anomalies by taking sequential (and sometimes semantic) information into account. They cannot consider the full structure among log events though. In contrast, graph-based methods, such as GLAD-PAW [11] and GLAD [5], convert logs to graphs and exploit semantic information as well as the structure among log events, exhibiting the following three advantages over the former two categories of methods [4]: 1) they are able to identify problems for which the structure among events is crucial, such as performance degradation; 2) they are capable of providing contextual log messages corresponding to the identified problems; and 3) they can provide the 'normal' operation process in the form of a graph, helping end-users find root-causes and take remedial actions. However, graph-based methods like GLAD transform log events into *undirected* graphs though, which may fail to capture important information on the order among log events. Moreover, most existing graph-based methods perform graph representation and anomaly detection separately, leading to suboptimal performance.

The growing use of anomaly detection algorithms in critical areas has led to a need for explanations of their decisions, as detailed by Li et al [7]. Precise anomaly explanations are vital for quick isolation and diagnosis, reducing the impact of anomalies through early intervention. Yet, most current log anomaly detection methods focus on accuracy while ignoring explanations.

To overcome these limitations, we propose *Logs2Graphs*, a graph-based unsupervised log anomaly detection approach for which we design a novel one-class graph neural network. Specifically, *Logs2Graphs* leverages the rich and expressive power of attributed, directed, and edge-weighted graphs to represent logs, followed by using graph neural networks to effectively detect graph-level anomalies, taking into account both semantic information of log events and structure information (including sequential information as a special case) among log events. Importantly, by decomposing the anomaly score of a graph into individual nodes and visualizing these nodes based on their contributions, we provide understandable explanations for identified anomalies.

2 APPROACH

Logs2Graphs consists of the following main steps.

Log Parsing & Log Grouping: The *log parsing* step extracts log events and log parameters from raw log messages, where we use Drain [3] for this task. Next, the *log grouping* step uses the log identifiers to divide the parsed log messages into log groups.

Graph Construction: First,we utilise nodes to represent different log events. Second, starting from the first line of log messages in chronological order, we add a directed edge from log event L_i to L_j and set its edge-weight to 1 if the next event after L_i is L_j. If the corresponding edge already exists, we increase its edge-weight by 1. Moreover, we augment the nodes with the semantic embeddings of the log events as attributes. In this manner, we obtain an attributed, directed, and edge-weighted graph.

Graph Representation Learning & Anomaly Detection We propose One-Class Digraph Inception Convolutional Networks, abbreviated as OCDiGCN, a novel method for end-to-end graph-level anomaly detection. Particularly, we train a one-class classifier

ICSE-Companion '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0502-1/24/04.
https://doi.org/10.1145/3639478.3643084

Table 1: AP and RC denote Average Precision and ROC AUC, respectively (bold: top 1, underline: top 2).

Method	HDFS		Hadoop		BGL		Spirit		Thunderbird	
	AP	RC	AP	RC	AP	RC	AP	RC	AP	RC
PCA	0.91±0.03	**1.0**±0.00	0.84±0.00	0.52±0.00	0.73±0.01	0.82±0.00	0.31±0.00	0.19±0.00	0.11±0.00	0.34±0.01
OCSVM	0.18±0.01	0.88±0.01	0.83±0.00	0.45±0.00	0.47±0.00	0.47±0.01	0.34±0.00	0.29±0.00	0.12±0.00	0.45±0.01
IForest	0.73±0.04	0.97±0.01	0.85±0.01	0.55±0.01	0.79±0.01	0.83±0.01	0.32±0.03	0.23±0.02	0.11±0.01	0.24±0.10
HBOS	0.74±0.04	0.99±0.00	0.84±0.00	0.50±0.00	0.84±0.02	0.87±0.03	0.35±0.00	0.22±0.00	0.15±0.01	0.29±0.05
DeepLog	**0.92**±0.07	0.97±0.04	**0.96**±0.00	0.47±0.00	0.89±0.00	0.72±0.00	0.99±0.00	0.97±0.00	0.91±0.01	0.96±0.00
LogAnomaly	0.89±0.09	0.95±0.05	**0.96**±0.00	0.47±0.00	0.89±0.00	0.72±0.00	0.99±0.00	0.97±0.00	0.90±0.01	0.96±0.00
AutoEncoder	0.71±0.03	0.84±0.01	**0.96**±0.00	0.52±0.00	0.91±0.01	0.79±0.02	0.96±0.00	0.92±0.01	0.44±0.02	0.46±0.05
GLAM	0.78±0.08	0.89±0.04	0.95±0.00	**0.61**±0.00	0.94±0.02	0.90±0.03	0.93±0.00	0.91±0.00	0.75±0.02	0.85±0.01
Logs2Graphs	0.87±0.04	0.91±0.02	0.95±0.00	0.59±0.00	**0.96**±0.01	**0.93**±0.01	**1.0**±0.00	**1.0**±0.00	**0.99**±0.00	**1.0**±0.00

by optimising the following One-Class Deep SVDD objective:

$$\min_{\mathcal{H}} \frac{1}{M}\sum_{m=1}^{M}\|\text{DiGCN}(\mathcal{G}_\text{m};\mathcal{H})-\mathbf{o}\|_2^2+\frac{\lambda}{2}\sum_{l=1}^{L}\|\mathbf{H}^{(l)}\|_F^2, \quad (1)$$

where $\mathbf{H}^{(l)}$ represents the trainable parameters of DiGCN [10] at the l-th layer, $\mathcal{H}$ denotes $\{\mathbf{H}^{(1)},...,\mathbf{H}^{(L)}\}$, and $\mathbf{o}$ is the center of the hypersphere in the learned representation space. After training the model on a set of non-anomalous graphs, the anomaly score of a test graph is defined as

$$score(\mathcal{G}_m)=\|\text{DiGCN}(\mathcal{G}_\text{m};\mathcal{H})-\mathbf{o}\|_2. \quad (2)$$

Anomaly Explanation: The importance score of node v_j (in the penultimate layer) in a graph $\mathcal{G}_m$ is defined as

$$\frac{|score(\mathcal{G}_m)-score(\mathcal{G}_m\setminus\{\mathbf{Z}_j\})|}{score(\mathcal{G}_m)} \quad (3)$$

where $score(\mathcal{G}_m)$ is defined in Equation 2 and $score(\mathcal{G}_m\setminus\{\mathbf{Z}_j\})$ is the anomaly score by removing the embedding vector of node v_j when applying the Readout function to obtain the graph-level representation. Next, we extend the LRP algorithm [1] to obtain a minor set of important nodes in the input layer.

3 RESULTS

Anomaly Detection: As shown in Table 1, in terms of ROC AUC, *Logs2Graphs* achieves the best performance against its competitors on three out of five datasets. Particularly, *Logs2Graphs* outperforms the closet competitor on BGL with 9.6% and delivers remarkable results (i.e., an ROC AUC larger than 0.99) on Spirit and Thunderbird. Similar observations can be made for Average Precision.

Anomaly Explanation: Figure 1 provides an example of log anomaly explanation with the HDFS dataset.

In summary, we introduced *Logs2Graphs* for unsupervised log anomaly detection. Extensive results on five benchmark datasets reveal that our method is at least comparable to and often outperforms other log anomaly detection methods. All code and datasets are provided on GitHub, and technical details can be found in [6].[1]

[1]**Zhong** and **Matthijs**: this research is part of the project Digital Twin with project number P18-03 of the research programme TTW Perspective, which is (partly) financed by the Dutch Research Council (NWO). **Jiayang**: the European Union H2020-MSCA-ITN-2020 under grant agreement no. 956172.

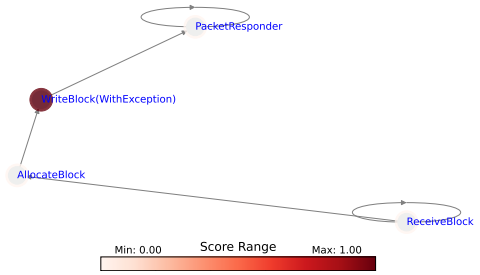

Figure 1: Example of anomaly explanation with HDFS.

REFERENCES

[1] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one* 10, 7 (2015), e0130140.

[2] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. *SIGSAC'17*, 1285–1298.

[3] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *ICWS'17*. IEEE, 33–40.

[4] Tong Jia, Pengfei Chen, Lin Yang, Ying Li, Fanjing Meng, and Jingmin Xu. 2017. An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services. In *2017 IEEE international conference on web services (ICWS)*. IEEE, 25–32.

[5] Yufei Li, Yanchi Liu, Haoyu Wang, Zhengzhang Chen, Wei Cheng, Yuncong Chen, Wenchao Yu, Haifeng Chen, and Cong Liu. 2023. GLAD: Content-aware Dynamic Graphs For Log Anomaly Detection. *arXiv preprint arXiv:2309.05953* (2023).

[6] Zhong Li, Jiayang Shi, and Matthijs van Leeuwen. 2023. Graph Neural Network based Log Anomaly Detection and Explanation. *arXiv preprint arXiv:2307.00527* (2023).

[7] Zhong Li, Yuxuan Zhu, and Matthijs van Leeuwen. 2023. A Survey on Explainable Anomaly Detection. *ACM Trans. Knowl. Discov. Data* (jul 2023). https://doi.org/10.1145/3609333

[8] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, et al. 2019. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. *IJCAI* 19, 7, 4739–4745.

[9] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural computation* 13, 7 (2001), 1443–1471.

[10] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. 2020. Digraph inception convolutional networks. *Advances in neural information processing systems* 33 (2020), 17907–17918.

[11] Yi Wan, Yilin Liu, Dong Wang, and Yujin Wen. 2021. Glad-paw: Graph-based log anomaly detection by position aware weighted graph attention network. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 66–77.

[12] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I Jordan. 2009. Detecting large-scale system problems by mining console logs. *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 117–132.