A Study on Log Anomaly Detection using Deep Learning Techniques

Kamiya Pithode

***Abstract*—**Anomaly detection is critical in the administration of current large-scale networked systems. Logs created by security systems, servers, network tools, and different software applications are some methods of recording the operational behavior of the equipment or software. These logs are useful for obtaining useful data about system activity. Deep Learning (DL) has lately established the lead of performance in detecting intrusions, denial-of-service attacks, malware, hardware & software system failures. A research background is discussed about feature extraction, machine learning, and deep learning. It also discussed the challenges in log analysis such as unstructured data, instability, log burst, and availability of public datasets. Recurrent Neural Network (RNN) language standards reinforced with awareness to anomaly detection in system logs. The goal of the research is to make an overview of the current study on log anomaly detection utilizing Deep Neural Networks. The research is also providing a summary of log parsing algorithms, datasets utilized for log analysis, and different ideas offered for log anomaly recognition. However, there is comprehensive comparison of representative log-based anomaly detectors that apply DL.

Keywords:* *Log Anomaly, Feature Extraction, Deep Learning, Recurrent Neural Network, Log burst.

I. INTRODUCTION

Anomaly detecting abnormalities in system logs are becoming more important for big corporations due to the rising complexity of systems and applications. There are vulnerabilities and more bugs in the security mechanisms, which is which is manipulated to launch an attack by the attacker. Researchers are working to increase log data processing efficiency and to investigate the possibility of detecting risks in logs [1]. The traditional log anomaly detection method is no longer useful as attackers get more sophisticated. Data mining and machine learning have been the most often used techniques such as Support Vector Machine (SVM), Decision Tree (DT), and Principal Component Analysis (PCA) must be utilized for separating more key elements. Logs faithfully reflect the runtime status of a software system, which is important for the monitoring, administering, and troubleshooting of a system. Therefore, log-based anomaly detection has become a crucial tool to assure system dependability and service quality which seeks to identify anomalous activity in the system [2].

Large-scale incident management is highly reliant on anomaly detection, which attempts to discover unpredictable anomalous behavior in a system. Anomaly detection is assisted system developers (or operators) in swiftly identifying and correcting issues, reducing system downtime, and improving system performance. Systems are created logs regularly which is capture comprehensive runtime information during system operation. System anomaly detection is relying heavily on logs that are readily accessible. Anomaly detection in traditional standalone systems is based on the developer's domain knowledge and the usage of

keywords (e.g., "failure" or "exception") or regular expressions to locate irregularities. However, for large-scale systems, such anomaly detection is mostly based on human log review and is no longer adequate due to the following factors [3].

- System behavior is too complicated for a single developer to understand the parallel nature and large-scale of current systems. Several open-source systems (such as Spark and Hadoop) are built through dozens or even hundreds of people.
- It is estimated modern systems generate 50 gigabytes (120~200 million lines) of log data each hour. It is impossible to manually extract the most valuable data from the noise data for anomaly detection in such many logs with the usage of search and filter [4].
- Many fault-tolerance mechanisms are utilized in large-scale systems. Systems are repeated the same work several times, or they are even deleting speculative tasks before they are performed. In such a setting, the conventional method of extracting suspicious log messages in these systems using keyword search becomes inefficient, resulting in many false positives. This will need a significant quantity of human inspection effort.

Log-based anomaly detection has been intensively researched during the last few decades. These have utilized attention processes in conjunction with DL models to assign greater weight to a certain sequence of data. The absence of comprehensive analysis for research effort done on log datasets and DL algorithms utilized to the datasets used for log anomaly detection is stifling future development in this field. There are studies of the various investigated methods for log anomaly detection utilizing DL [5]. There are no test oracles to check that the underlying machine learning algorithms are successfully implemented, massive efforts are usually required to replicate the systems [6].

II. RESEARCH BACKGROUND

A. Feature Extraction

This information is mined from the log context to get the vector features of a certain log's data. DL techniques in Natural Language Processing (NLP) utilize both word embedding and TF-IDF as input characteristics. Word2vec and TF-IDF are both utilized in this study to create dense vectors from the log's words for feature extraction [7].

1) TF-IDF: Term Frequency-Inverse Document Frequency (TF-IDF) is a popular feature extraction process. It is a method of determining the relative value of words within a body of work. This statistic shows how often a phrase occurs in the target document and throughout the corpus. The stronger a phrase's significance occurs in other texts and the more frequently to appears in the target material. The vectorizer class in the sci-kit-learn package is often used to compute the TF-IDF. In the process of sentiment analysis, raw data is transformed into vectors of real numbers. The certain formulas are as follows [8]:

$$TF = \frac{\text{the number of times it appears in the file}}{\text{the total number of words in the file}} \quad (1)$$

$$IDF = log \frac{\text{the total number of document in the corpus}}{\text{the number of document containing the word} + 1} \quad (2)$$

$$TF - IDF = TF \times IDF \quad (3)$$

2) Word Embedding: Word embedding is used to vector of real values to map each word to a representation that is similar to other words with the same meaning to model and learn aspects of a language. Neural networks are utilized to learn about values. Word2vec (GloVe, or Gensim) is a popular word embedding system that includes models like Continuous Bag-Of-Words (CBOW) and skip-gram. There are two ideas based on the possibility of words being near to one other [9]. Word2vec is offered two new models for transforming a word embedding to another. CBOW forecasts are provided the center word based on nearby phrases using the window dimension. In the second model, skip-gram, a core word is utilized as a reference point to anticipate the surrounding words. Word embedding has the primary benefit of keeping nearby in vector space various words used in a similar context. The vector difference between males and women is equivalent to that of “king” and “queen”. A queen is formed by multiplying "king" by a woman, then subtracting a man to get a queen. Word embeddings have proven useful in a variety of NLP applications, including anomaly identification, sentiment analysis, and parsing [10].

B. Machine Learning

Machine learning (ML) is the analysis of algorithm that can enhance mechanically across knowledge and with the usage of data. There are several machine learning algorithms such as support vector machine, decision tree and principal component analysis.

1) Support Vector Machine (SVM): SVM has been offered as a revolutionary intrusion detection tool. An SVM is utilized a nonlinear mapping to turn a true-valued input feature vector into a higher-dimensional feature space. SVMs are based on the structural risk reduction concept. Structural risk minimization strives to identify a hypothesis (h) with the lowest chance of mistake, while standard pattern recognition learning approaches are based on empirical risk reduction, which attempts to maximize the performance of the learning set. SVMs learn a wider variety of patterns and scale better because the classification complexity does not depend on the size of the feature space. SVMs dynamically update the training patterns which a new pattern in categorization emerges [11].

2) Decision Tree (DT): The decision tree is constructed from the class tuples. There are internal nodes, branches, and leaf nodes in a decision tree's structure. There is an internal node that gives the property to evaluate the branch to indicate the result of the test, and the leaf node represents the class name. The primary objective is predicting the productivity of a continuous characteristic but, decision trees are less effective for task estimation. It is easy to make errors in anticipating the classes when utilizing a decision tree method. Pruning algorithms are expensive, and building decision trees are costly as the splitting of nodes at each level [12].

3) Principal Component Analysis (PCA): PCA is developed to decrease the dimensions of a dataset by linearly converting the original space into a new space with fewer dimensions while characterizing the data's variability as much as possible. The PCA plots along the covariance matrix's eigenvalues have the highest values to determine the direction of the greatest data variation indicated by the eigenvectors. Eigenvalues are also utilized to determine the intrinsic dimension of data. If the magnitude of the x eigenvalues is bigger than the magnitude of the other eigenvalues, the number x is regarded as the data's true dimension. A linear approach can only extract linear correlations between variables, and it often overestimates the underlying dimension for data with complicated relationships [13].

C. Deep Learning

Deep Learning (DL) is subgroup of ML which is itself a subgroup of artificial intelligence (AI). There are various DL techniques such as long-short term memory, recurrent neural network, autoencoder, and bidirectional long-short term memory [14].

1) Long-Short Term Memory (LSTM): LSTM is an upgraded approach of classic RNN networks that tackles the problem of disappearing and bursting gradients by imposing a steady error flow. It is replaced by the memory block rather than the plain RNN unit in the LSTM algorithm. Input, output, forget, and a self-recurrent relationship along with a fixed weight of 1.0 are all included in a memory block. A memory cell is a location where values are stored and retrieved at various times. Long-term temporal territories are represented by estimating the current time step value using past information. Similarly, when a value becomes useless, the forget gate forgets or resets the previous data that exists in self-connection, and the output gate controls the output flow of a memory cell. In many Artificial Intelligence (AI)

applications, long-term temporal relationships are trained using LSTM [14].

2) Recurrent Neural Network (RNN): RNNs are used for time-series data modeling at the beginning of the experimentation. There is a cyclic loop added to the Feed-Forward Network (FFN). The information is sent from one-time step to the next in this cyclic cycle. RNNs are learned temporal patterns, and values at the current time-step are computed using prior and present states. RNNs have long excelled in machine translation, language modeling, and voice recognition to name a few long-standing AI positions [14].

3) Autoencoder: An autoencoder is a neural network technology that does not need supervision. It effectively compresses and encodes the information, encoded information, and recreates the productivity which is near to the initial effort. Autoencoders are utilized for feature extraction in an equivalent way to PCA. However, it is learned non-linear revolution using a non-linear stimulation role, which is useful for learning non-linear transformation [15].

4) Bi-LSTM: Bidirectional Long-Short Term Memory LSTM (Bi-LSTM) is an extension that separates a standard LSTM's buried layer of neurons into two opposed orientations, i.e., backward, and forward. Bi-LSTM has collected log sequence knowledge through both input sources [15].

III. CHALLENGES IN LOG ANALYSIS

System logs are difficult to analyze to their unstructured nature because they include a huge quantity of duplicate information. It must convert logs into structured data before doing log analysis. Previously, log keywords are extracted and deleted redundant data. The diverse character of logs causes several processing issues [16].

A. Unstructured data

Unstructured or semi-structured logs are the most common, and they vary depending on the operating system, software version, and Original Equipment Manufacturer (OEM) of the device. Log files are unstructured data and lack a defined formal structure or syntax. A significant problem is the centralized collecting and processing of all this data [41].

B. Instability

It is recognized the existence of the log instability issue. Several factors have been cited, including the growth of logging statements via source code update and the handling of making noise in log data. There is no predefined collection of logs or numerous logs necessary for some given action, and there is no clear rule for how they should be generated [17].

C. Log burst

The number of log data is growing at an exponential rate as devices become more safe software and sophisticated. All logs must be gathered centrally for storage, correlation, and later analysis to compound the difficulty of the big data problem. The log burst is generated as the creation process complicates log analysis and data extraction from raw log data even more [41].

D. Availability of public dataset

The logs are unstructured much of the time, and their substances are insecure; so, they cannot be made public due to security issues. It is unlikely publicly available datasets will be made available for research purposes [41].

IV. LOG ANOMALY DATASETS

A NetEngine40E series router from a firm is high-end network equipment. It is utilized log data generated by the router in the actual network for one month for the experiment. The total number of logs utilized is 18,727,517, with a dataset volume of 1.09 GB [18]. The Autoencoder output for the Openstack dataset had over 180,000 positive log messages and 180,000 negative log messages. Six thousand sixty-eight messages are utilized for training, one thousand one hundred sixty-seven messages for validation, and the remaining 147,733 logs for testing [19].

As part of an OpenStack experiment (version Mitaka), a single control node, a single network node, and eight compute nodes are set up on CloudLab. Approximately 7% of the unusual collected of 1,335,318 log entries. A script is always running carried out automated activities such as creating/deleting/stopping and suspending/resuming Virtual Machines (VMs) [20]. Amazon EC2 log messages total 11,175,629 messages and are stored in Hadoop Distributed File System (HDFS). Each block activity, including allocation, writing, replication, and deletion, is logged in HDFS using a unique block ID. Log operations are more naturally captured by session windows since each unique block ID can be utilized to break the logs into a collection of log sequences as specified. Following that, 575,061 event count vectors are extracted from these log sequences. Outliers were identified in a total of 16,838 samples [21].

In the Blue Gene/L (BGL) data, the BGL supercomputer system at Lawrence Livermore National Laboratory (LLNL) collected 4,747,963 log messages. There is no unique identifier for each task execution in BGL logs, unlike HDFS data. It is divided logs into log sequences using fixed or sliding windows before extracting the appropriate event count vectors. It is getting a different number of windows depending on the size of the windows. In the BGL data, 348,460 log messages are classified as failures, and every log structure that contains any failure records is labeled as an anomaly [22]. Table 1 shows the summary of log anomaly datasets.

Table 1. Summary Of Log Anomaly Datasets

S. No.	Dataset Name	Log Source	No of Message	Log Source Type	References
1.	Router Logs	NetEngine40E Router	18,727,517	Router	[18]
2.	Openstack	OpenStack Software	137,074	Distributed System	[19]
3.	OpenStack	CloudLab	1, 335, 318	Distributed System	[20]
4.	HDFS	Hadoop Distributed File System	11,175,629	Distributed System	[21]
5.	BGL	Blue Gene/L supercomputer	4,747,963	Supercomputer	[22]

V. REVIEW ON EXISTING METHODOLOGY

This section goes through log processing, feature extraction, and anomaly detection in detail. It is providing a brief overview of log parsing and explores a few of the most popular log parsers in use today. Three feature extraction techniques are utilized to analyze log issues to produce feature vectors. It is developed picked six example anomaly detection techniques based on the feature vectors which utilize supervised methods and the other three utilize unsupervised approaches. Hadoop and SILK are utilized to put the planned approach for detecting anomalies to the test. It is examining a few real-world situations and draws some general conclusions about demonstrating the technique. Hadoop is an open-source version of Google's well-known Map-Reduce architecture and Distributed File System (DFS). This system can distribute large-scale, data-intensive, stage-based parallel programmers [23].

Log Parsing: A text log message is considered in two ways: as a whole and in chunks. The fixed section text in messages remains constant, while the changeable parts include values that vary depending on the parameter. This is insight inspires the creation of an event concept and a log message template. Preprocessed structured data must be processed in line with the requirements of the machine learning or deep learning model. These methods are utilized to analyze textual data. There are ways to log parsing that are heuristic-centered (e.g., iPLoM [24], SLCT [25]) as well as clustering-centered (e.g., LKE [26], LogSig [27]).

Cluster-based log parsers start by determining the distances between logs and then use clustering techniques to group the logs into various clusters. Finally, each cluster creates a template for an event. These are counted heuristic-based strategies, the number of events of every phrase on every log position [28]. The next stage is to find and create phrases that are often used as candidates for the event. It is previously created and tested four log parsers. This work parses raw logs into log events using an open-source log parsing framework that it has made publicly accessible [29].

It is feasible to improve accuracy by adding domain information to log data, but the techniques are not general. Log parsers such as SPELL [30], FT-Tree [31], and Drain [32] are utilized by M Du et al. [33], Weibin Meng et al. [34], and Xu Zhang et al. [35] to generate log issue patterns and log structures. There are many alternatives based on the problem description. Depending on the model technique utilized statistical, deep learning, and machine learning are all feasible strategies for detecting abnormalities. The anomaly is decoded by looking at their semantics in addition to looking at the logs' chronological sequence.

VI. LITERATURE SURVEY

Chen, Zhuangbin, et al. [36] stated that the logs have long been seen as an essential resource for ensuring the dependability and continuation of many software systems to particularly large-scale distributed systems. These are meticulously captured runtime information to aid in system troubleshooting and behavior analysis. The number of logs generated by current software systems has surpassed all previous records. Standard human inspection methods and even conventional machine learning-based approaches have become impractical for log-based anomaly detection, which acts as a catalyst for the rapid development of deep learning-based solutions. There is no comprehensive comparison of conventional log-based anomaly detectors that utilize neural networks. The re-implementation procedure is very time-consuming, and prejudice to easily develop. The objective of this research is to provide a complete assessment and evaluation of five commonly used neural networks utilized by six cutting-edge methods to better comprehend the characteristics of different anomaly detectors. Several techniques are tested using around 16 million log messages and 0.4 million anomaly occurrences on the two publicly available datasets. It is anticipated that research will provide the groundwork for future academic and industrial research in this area.

Li, Xiaoyun, et al. [37] studied that have been done on log-based anomaly detection, and it works effectively when

dealing with stable log data. Current efforts are falling short of addressing the following issues: log formats are always developing in actively created and maintained software systems. Latent causes are not detectable by standard monitoring techniques are based on performance issues. SwissLog, a deep learning-based anomaly detection method has been suggested to detect a broad spectrum of problems. SwissLog focuses on the log sequence order and logs time interval changes that occur to address these concerns. Furthermore, a unified attention based BiLSTM model is trained to detect anomalies utilizing the semantic embedding and temporal embedding approaches. SwissLog's capacity to manage a broad variety of mistakes and endure changes in log data has been shown via testing on real-world and synthetic datasets.

Brown, Andy et al. [38] stated that DL has recently performed very well in computer systems maintenance chores such as identifying intrusions, denial-of-service attacks, hardware and software faults, and viruses. Model interpretability is critical for administrators and analysts to have trust in automated machine learning model analysis. DL techniques have been chastised for their lack of transparency as a black box oracle. It is conducted this work to "bridge the gap" among the spectacular achievements of DP patterns and the requirement for understandable model introspection. RNN language models have been upgraded for anomaly detection in system logs as a way of reaching this goal. Many approaches are utilized to logging sources or some computer system. It can do paradigm introspection and analysis devoid of surrendering innovative performance by including attention variations into RNN language models. These are verifying the model's performance and emphasize interpretability. An intrusion detection assignment is utilizing the Los Alamos National Laboratory (LANL) cyber safety dataset with the model describing a region under the receiver operator typical curve of up to 0.99.

Du, Min, et al. [39] studied that the identification of anomalies is a crucial step in creating a safe and reliable system. For the most part, system logs are used for troubleshooting system errors and root cause analysis by recording system states and noteworthy occurrences at different crucial moments. Almost all computer systems have access to this kind of log data. The different system logs are a fantastic source of information for online monitoring and anomaly detection since log data is a significant and useful resource for understanding system-level and implementation concerns. DeepLog can detect anomalies in log patterns when they diverge from the model based on log data during regular operation and learn new log patterns from the log data automatically. It is illustrating how the DeepLog model is modified live in an iterative, incremental method to make it more flexible and sensitive to changing log patterns. DeepLog is identified uses the underlying system log to develop processes, enabling users to analyze anomalies. Large log data sets indicated that DeepLog outperforms competing log-based anomaly detection systems created on conventional data mining processes.

He, Shilin, et al. [40] stated that modern large-scale distributed systems need a high level of anomaly detection. It is common to practice examining logs to spot anomalies in system performance. Manual log inspection by developers (or operators) exploitation keyword search and rule matching has long been the norm. Manual inspection is no longer feasible due to the ever-increasing complexity and scope of contemporary systems. Many automated log analysis approaches have been presented to minimize the amount of time it takes to identify anomalies. In the absence of research and comparison of multiple anomaly detection methods, developers are unclear which one to choose. Anomaly detection techniques, on the other hand, need a large amount of time and effort. A comprehensive examination and assessment of six cutting-edge log-based anomaly detection systems are offered, including three supervised and three unsupervised approaches and an open-source toolkit for simple re-use. The evaluation findings and conclusions, to the best of knowledge, impact the application of these approaches and serve as a foundation for future progress. Table 2 shows the summary of related work.

Table 2. Summary of Related Work

S. no	Author's	Year	Techniques	Outcome
1.	Chen, Zhuangbin, et al. [36]	2021	Deep Learning	It will provide the groundwork for future academic and industrial research in this area.
2.	Li, Xiaoyun, et al. [37]	2020	SwissLog	SwissLog's capacity to manage a broad variety of mistakes and endure changes in log data has been shown via testing on real-world and synthetic datasets.
3.	Brown, Andy et al. [38]	2018	RNN	An intrusion detection task using the LANL cyber safety dataset with the model reporting a region under the

				receiver operator typical curve of up to 0.99.
4.	Du, Min, et al. [39]	2017	Deep Neural Network	Large log data sets indicated that DeepLog outperforms competing log-based anomaly detection systems created on conventional data mining processes.
5.	He, Shilin, et al. [40]	2016	Machine Learning	It is evaluation findings and conclusions, to the best of knowledge, are impact the application of these approaches and serve as a foundation for future progress.

VII. COMPARATIVE ANALYSIS

This section of the paper contains the comparative analysis in which various data mining techniques are applied for Log anomaly detection using DL. It is the most popular algorithm for anomaly detection. There are used many datasets such as drain, BGL, SwissLog, HDFS, and OpenStack. SwissLog has highest 097% precision. Second, HDFS has 0.96% precision. Table 3 shows the Comparison based on Accuracy.

Table 3 Comparison based on Accuracy

Datasets Type	Precision	Recall	F1-Score
Drain [37]	0.95	0.96	0.96
BGL [36]	0.93	0.98	0.96
SwissLog [37]	0.97	1.00	0.99
HDFS [36]	0.96	0.92	0.94
OpenStack [42]	0.77	0.99	0.87

VIII. CONCLUSION AND FUTURE SCOPE

This study conducted that DL used for log evaluation is a fast-emerging discipline for extracting information from unstructured textual log data. It has covered numerous research methods in deep learning-based anomaly detection, as well as their application in a variety of fields. Logs are utilized to detect abnormalities in today's large-scale distributed systems. It is difficult to detect abnormalities solely on human log assessment because log volumes have expanded in recent years. Recent years have seen a rise in the use of automated log analysis and anomaly detection systems. There is no systematic study and comparison of existing approaches, developers are still unaware of the most up-to-date methods for anomaly detection, and frequently have to re-design a new anomaly detection technique. Logs have been utilized extensively in the maintenance of software systems for a long time. Modern software systems' log-based anomaly detection approaches are being overwhelmed by statistics and classical machine learning algorithms due to their unparalleled volume.

Many attempts have been made to construct DL-based anomaly detectors to explore more intelligent solutions. Therefore, site reliability engineers must have a thorough grasp of properly deploying DL techniques. Word2vec and TF-IDF, as well as LSTM deep learning algorithms, are integrated with NLP methods, such as Word2vec, to give an effective and accurate way for anomaly identification. It is expected that in the future, the deficiency of log analysis for research effort done on log datasets and DL algorithms utilized to the datasets utilized for log anomaly detection is stifling future development in this field.

REFERENCES

[1]. TAN, Tun-Zi, Xing-Chen ZHANG, Sui-Xiang GAO, Wen-Guo YANG, Yue-Zhong SONG, and Cheng-Yong LIN. "HWLog Analysis: A Tool for Routers' Syslog Anomaly Detection and Root Causes Diagnosis." In Artificial Intelligence Science and Technology: Proceedings of the 2016 International Conference (AIST2016), pp. 799-806. 2017.

[2]. Lin, Qingwei, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. "Log clustering-based problem identification for online service systems." In 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), pp. 102-111. IEEE, 2016.

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

[4]. Mi, Haibo, Huaimin Wang, Yangfan Zhou, Michael Rung-Tsong Lyu, and Hua Cai. "Toward fine-grained, unsupervised, scalable performance diagnosis for

production cloud computing systems." IEEE Transactions on Parallel and Distributed Systems 24, no. 6 (2013): 1245-1255.

[5]. Chalapathy, Raghavendra, and Sanjay Chawla. "Deep learning for anomaly detection: A survey." arXiv preprint arXiv:1901.03407 (2019).

[6]. He, Pinjia, Jieming Zhu, Shilin He, Jian Li, and Michael R. Lyu. "An evaluation study on log parsing and its use in log mining." In 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN), pp. 654-661. IEEE, 2016.

[7]. Wang, Mengying, Lele Xu, and Lili Guo. "Anomaly detection of system logs based on natural language processing and deep learning." In 2018 4th International Conference on Frontiers of Signal Processing (ICFSP), pp. 140-144. IEEE, 2018.

[8]. Zhang, Ke, Jianwu Xu, Martin Renqiang Min, Guofei Jiang, Konstantinos Pelechrinis, and Hui Zhang. "Automated IT system failure prediction: A deep learning approach." In 2016 IEEE International Conference on Big Data (Big Data), pp. 1291-1300. IEEE, 2016.

[9]. Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[10]. Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[11]. Adebowale, Ajayi, S. A. Idowu, and A. Amarachi. "Comparative study of selected data mining algorithms used for intrusion detection." International Journal of Soft Computing and Engineering (IJSCE) 3, no. 3 (2013): 237-241.

[12]. Gera, Mansi, and Shivani Goel. "Data mining-techniques, methods and algorithms: A review on tools and their validity." International Journal of Computer Applications 113, no. 18 (2015).

[13]. Jo, Han-Shin, Chanshin Park, Eunhyoung Lee, Haing Kun Choi, and Jaedon Park. "Path loss prediction based on machine learning techniques: Principal component analysis, artificial neural network, and Gaussian process." Sensors 20, no. 7 (2020): 1927.

[14]. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning. nature 521 (7553), 436-444." Google Scholar Google Scholar CrossRef CrossRef (2015).

[15]. Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." science 313, no. 5786 (2006): 504-507.

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

[17]. Zhang, Xu, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie 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, pp. 807-817. 2019.

[18]. Wang, Xiaojuan, Defu Wang, Yong Zhang, Lei Jin, and Mei Song. "Unsupervised learning for log data analysis based on behavior and attribute features." In Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, pp. 510-518. 2019.

[19]. Farzad, Amir, and T. Aaron Gulliver. "Log message anomaly detection and classification using auto-b/lstm and auto-GRU." arXiv preprint arXiv:1911.08744 (2019).

[20]. Ricci, Robert, Eric Eide, and CloudLab Team. "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications."; login: the magazine of USENIX & SAGE 39, no. 6 (2014): 36-38.

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

[22]. Oliner, Adam, 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), pp. 575-584. IEEE, 2007.

[23]. Fu, Qiang, Jian-Guang Lou, Yi Wang, and Jiang Li. "Execution anomaly detection in distributed systems through unstructured log analysis." In 2009 ninth IEEE international conference on data mining, pp. 149-158. IEEE, 2009.

[24]. Makanju, Adetokunbo AO, A. Nur Zincir-Heywood, and Evangelos E. Milios. "Clustering event logs using iterative partitioning." In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1255-1264. 2009.

[25]. Vaarandi, Risto. "A data clustering algorithm for mining patterns from event logs." In Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No. 03EX764), pp. 119-126. Ieee, 2003.

[26]. Fu, Qiang, Jian-Guang Lou, Yi Wang, and Jiang Li. "Execution anomaly detection in distributed systems through unstructured log analysis." In 2009 ninth IEEE international conference on data mining, pp. 149-158. IEEE, 2009.

[27]. Tang, Liang, Tao Li, and Chang-Shing Perng. "LogSig: Generating system events from raw textual logs." In Proceedings of the 20th ACM international conference on Information and knowledge management, pp. 785-794. 2011.

[28]. He, Pinjia, Jieming Zhu, Shilin He, Jian Li, and Michael R. Lyu. "An evaluation study on log parsing and its use in log mining." In 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN), pp. 654-661. IEEE, 2016.

[29]. He, Pinjia, Jieming Zhu, Shilin He, Jian Li, and Michael R. Lyu. "An evaluation study on log parsing and its use in log mining." In 2016 46th annual IEEE/IFIP international conference on dependable systems and networks (DSN), pp. 654-661. IEEE, 2016.

[30]. Yen, Ting-Fang, Alina Oprea, Kaan Onarlioglu, Todd Leetham, William Robertson, Ari Juels, and Engin Kirda. "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks." In Proceedings of the 29th Annual Computer Security Applications Conference, pp. 199-208. 2013.

[31]. Du, Min, and Feifei Li. "Spell: Streaming parsing of system event logs." In 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 859-864. IEEE, 2016.

[32]. Zhang, Shenglin, Weibin Meng, Jiahao Bu, Sen Yang, Ying Liu, Dan Pei, Jun Xu, et al. "Syslog processing for switch failure diagnosis and prediction in datacenter networks." In 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), pp. 1-10. IEEE, 2017.

[33]. Du, Min, Feifei Li, Guineng Zheng, and Vivek Srikumar. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298. 2017.

[34]. Meng, Weibin, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen et al. "LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs." In IJCAI, vol. 19, no. 7, pp. 4739-4745. 2019.

[35]. Zhang, Xu, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie 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, pp. 807-817. 2019.

[36]. Chen, Zhuangbin, Jinyang Liu, Wenwei Gu, Yuxin Su, and Michael R. Lyu. "Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection." arXiv preprint arXiv:2107.05908 (2021).

[37]. Li, Xiaoyun, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba 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), pp. 92-103. IEEE, 2020.

[38]. Brown, Andy, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. "Recurrent neural network attention mechanisms for interpretable system log anomaly detection." In Proceedings of the First Workshop on Machine Learning for Computing Systems, pp. 1-8. 2018.

[39]. Du, Min, Feifei Li, Guineng Zheng, and Vivek Srikumar. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285-1298. 2017.

[40]. He, Shilin, Jieming Zhu, Pinjia He, and Michael R. Lyu. "Experience report: System log analysis for anomaly detection." In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 207-218. IEEE, 2016.

[41]. Yadav, Rakesh Bahadur, P. Santosh Kumar, and Sunita Vikrant Dhavale. "A Survey on Log Anomaly Detection using Deep Learning." In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pp. 1215-1220. IEEE, 2020.

[42]. Du, M., Li, F., Zheng, G. and Srikumar, V., 2017, October. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (pp. 1285-1298).