CHAOS
Theory and Applications
in Applied Sciences and Engineering

e-ISSN: 2687-4539
RESEARCH ARTICLE
Vol.5 / No.3 / 2023 / pp. 188-197
https://doi.org/10.51537/chaos.1348302

Anomaly Detection in Cyber Security with Graph-Based LSTM in Log Analysis

Yusuf Alaca [*,1], **Yüksel Çelik** [α,2] **and Sanjay Goel** [β,3]

[*]Osmancık Omer Derindere Vocational School, Hitit University, Corum, Turkiye, [α]Department of Computer Engineering, Karabuk University, Karabuk,Turkiye, [β]University at Albany, State University of New York BA 310b, 1400 Washington Ave. Albany, NY 12222, USA.

ABSTRACT Intrusion detection systems utilize the analysis of log data to effectively detect anomalies. However, detecting anomalies quickly and effectively in large and heterogeneous log data can be challenging. To address this difficulty, this study proposes the GLSTM (Graph-based Long Short-Term Memory) framework, a graph-based deep learning model that analyzes log data to detect cyber-attacks rapidly and effectively. The framework involves standardizing the complex and diverse log data, training this data on an artificial intelligence model, and detecting anomalies. Initially, the complex and diverse log data is transformed into graph data using Node2Vec, enabling efficient and rapid analysis on the artificial intelligence model. Subsequently, these graph data are trained using LSTM (Long Short-Term Memory), Bi-LSTM, and GRU(Gated Recurrent Unit) deep learning algorithms. The proposed framework is tested using Hadoop's HDFS dataset, collected from different systems and heterogeneous sources, as well as the BGL and IMDB datasets. Experimental results on the selected datasets demonstrate high levels of success.

KEYWORDS
Anomaly detection
Graph
Node2Vec
Deep learning
Cyber security
HDFS

INTRODUCTION

Logging is the process of collecting numerical and textual data that captures the behavior of software, including events like low memory conditions or attempts to access files. The current focus of logging practices primarily revolves around the storage and organization of logs (Wang *et al.* 2019). Logging mechanisms consist of extensive datasets of log statements and their corresponding activation codes, which are implemented either by developers or specific software platforms.

In large internet networks, analyzing event and system-based logs using a combination of multiple systems, software, and hardware is crucial. Since log records are collected from devices and software responsible for system security, they often contain traces of attacks carried out by malicious actors during or after an attack. Therefore, it is essential to analyze log records and detect anomalies resulting from these traces in order to identify cyber-attacks (Elbasani and Kim 2021).

Manuscript received: 23 August 2023,
Revised: 5 October 2023,
Accepted: 18 October 2023.

[1]yusufalaca@hitit.edu.tr (**Corresponding author**).
[2]yuksel.celik@karabuk.edu.tr
[3]goel@albany.edu

Numerous techniques have been developed to address the challenges associated with log analysis and anomaly detection. These techniques include frequent pattern mining, heuristics, clustering, evolutionary algorithms, and deep learning (He *et al.* 2020). However, it has been observed that these techniques are not as effective and efficient in detecting log anomalies as our proposed method.

To overcome these challenges, comprehensive log data collected from various devices and software in different structures needs to be converted into a standardized format for analysis. Additionally, it is necessary to analyze standardized data effectively and efficiently in order to detect attacks (Li and Li 2020).

In this study, we propose a framework that converts diverse log data into graphs and detects anomalies using deep learning methods. Our framework utilizes the node2vec algorithm, which is a semi-supervised and heuristic approach, to convert different log data into graphs. By leveraging node2vec, we can scale feature learning and select adjacent nodes through a random walk approach between nodes. This algorithm offers flexibility due to its adjustable parameters (Grover and Leskovec 2016).

For the deep learning component of our framework, we employ the LSTM algorithm, which is a type of recurrent neural network (RNN). Unlike traditional RNNs, LSTM networks address issues such as gradient weakening or gradient bursting that can occur in redundant neural networks (Hochreiter and Schmidhuber 1997).

LSTM networks also utilize feedback connections instead of solely relying on feed-forward connections. In this study, the data is first transformed using the node2vec algorithm and then inputted into the LSTM algorithm. Our experiments have demonstrated that this approach achieves high accuracy in detecting anomalies.

When examining the literature, four different methods have been employed in studies on log anomaly detection. These methods are as follows: 1. Stencil removal, 2. Document management, 3. System monitoring, and 4. Learning-based anomaly detection. Several studies have focused on template extraction within these methods.

The template extraction method aims to extract word frequencies from log files, identify abnormal words, and detect anomalies. To be considered a template, terms must surpass a certain threshold. IPLoM, for instance, is a study that employs this method, recursively splitting log records assuming equal line lengths (Makanju *et al.* 2009).

Another study utilizing the template extraction technique applies deep learning to sequentially stored log records, using natural language processing to detect anomalies. Anomalies are detected when the sequential order is disrupted or when log records deviate from the expected flow. Meaningful words are extracted from log records and organized into templates. These templates are then converted into vectors, a method referred to as template2vector (Meng *et al.* 2019a). LSTM, a deep learning algorithm, is utilized in this study, with HDFS and BGL datasets employed for testing purposes(Alaca and Çelik 2023).

Document management, particularly using the Word2Vec method, is also prominent in log anomaly detection. Word groups are created, dividing words into different categories such as sentences and paragraphs based on the dataset size (Church 2017). In another study employing this method, natural language processing techniques are applied to detect anomalies in Thunderbird logs and system log records. Word2Vec and TF-IDF feature extraction algorithms are used, and the LSTM deep learning algorithm is employed for classification analysis (Wang *et al.* 2018).

System monitoring is another method used for anomaly detection. Log records from various systems can be monitored, and an exemplary tool in this context is Google's Dapper tool. This tool has demonstrated high success in complex, large-scale distributed systems (Sigelman *et al.* 2010).

In the literature, numerous studies are based on learning approaches, utilizing various machine learning techniques. DeepLog is a prominent study for log anomaly detection. The proposed approach consists of two main parts: defining the log key and establishing a workflow that includes anomaly parameters. The anomaly parameters are converted into vectors based on the log key, and the LSTM algorithm from artificial neural networks is employed to detect anomalies corresponding to the log key. The algorithm also incorporates manual feedback to improve accuracy (Du *et al.* 2017).

In another study, the CNN algorithm, a deep learning technique, is employed for anomaly detection from log records (Lu *et al.* 2018). This study identifies keywords in log records and detects anomalies based on these keywords. The identified keywords are digitized, normalized, and transformed into a 29x128 vector. This method is referred to as logkey2vector.

Deep learning is further explored in a study where different models are developed using datasets such as BGL (BlueGene/L), Thunderbird, Openstack, and IMDB (Internet Movie Database). The IMDB dataset is used to demonstrate the generalizability of the proposed model for other text classification problems. Positive and negative labeled data are fed into two distinct Autoencoders to enhance understanding of the original data, and the output is used as input for deep learning algorithms such as LSTM, BLSTM, and GRU (Farzad and Gulliver 2019).

An alternative approach to log anomaly detection aims to detect subsets of the original data space by making multiple passes over the entire dataset using frequent pattern mining. This approach involves three steps: summarizing the data by traversing the dataset, generating cluster candidates through another pass, and selecting suitable clusters from the candidates (Vaarandi 2003).

Graph structures have been employed in multiple studies for anomaly detection from log records. In one study, authentication logs are analyzed using graph structures to prevent unauthorized access to the operating system. A graph clustering method is developed specifically for forensic computing (Studiawan *et al.* 2017).

Another study utilizing graph structures detects anomalies from log data using time series and kill chain mechanisms. This advanced method creates attack profiles and simulates daily attacks on computer networks (Schindler 2017).

Graph structures have also been utilized in a study aiming to detect software errors in cloud computing (Yan *et al.* 2015). This method converts the complex relationships between log records into a graph and assigns importance scores to each log. The log anomaly detection method developed through this approach effectively identifies software errors.

In conclusion, various methods have been explored in the literature for log anomaly detection. These methods include stencil removal, document management, system monitoring, and learning-based approaches. Template extraction, deep learning algorithms like LSTM and CNN, as well as graph structures, have been utilized to detect anomalies in log records. Each method has its strengths and limitations, and further research is needed to enhance the accuracy and efficiency of log anomaly detection techniques.

It is crucial to continue advancing the field of log anomaly detection as it plays a vital role in ensuring the security and reliability of systems and networks. By detecting anomalies and potential cyberattacks, these techniques contribute to early threat identification and mitigation. Future research should focus on refining existing methods, exploring new algorithms, and leveraging emerging technologies to improve the effectiveness and scalability of log anomaly detection systems.

The aim of this study is to transform raw log data into meaningful and analyzable information that can effectively identify log anomalies. We achieve this by combining the node2vec and LSTM algorithms and applying them to the Hadoop HDFS dataset collected from multiple sources.

MATERIALS AND METHODS

The architecture of this study is based on the use of two algorithms together. First, the data converted to templates was vectorized using the Node2Vec algorithm to analyze it in deep learning algorithms. Then, this vectorial data was given as input to the LSTM algorithm, and models were created for anomaly detection; thus, anomaly detection was performed.

There are three types of anomalies in anomaly detection from log data. The first of these anomalies is the point anomaly. A point anomaly is data that deviates significantly from the mean or normal distribution of the remaining data (Gogoi *et al.* 2011). The second is the contextual anomaly. Contextual abnormality is an abnormal behavior confined to a specific context and standard

in other contexts (Ahmed *et al.* 2016). The third is the collective anomaly. Unlike contextual and point anomalies, aggregate anomalies appear in the data as abnormal values. Aggregate anomalies are the abnormal behavior of a collection of data samples relative to the entire dataset (Li *et al.* 2015).

Log anomaly detection identifies abnormal system patterns in log data that do not conform to expected behavior. This section discusses our work based on the algorithms adopted here. The outline of our study is shown in Fig. 1. First, raw log data were taken from different log groups and made meaningful by removing unnecessary and noisy data. Templates were created from this log data and given input to the Node2Vec algorithm to generate the feature vector. Model training was done with the LSTM algorithm, and anomaly detection was made with these trained models.

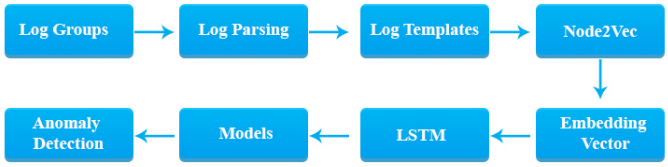

Figure 1 Flow chart of the proposed model algorithm.

Log parsing

Analysis of log data takes numerical and categorical data as input. This requires cleaning, sorting, and normalizing the raw log data. Log records consist of two main parts. Head part and text part. The head part usually consists of several features such as timestamps, hostnames, and severity of events. The developers manually predefine text message input. This can vary significantly between systems, even within one. These messages also consist of two parts: fixed messages and variable messages.

Each raw log data consists of two parts. One of them is the timestamp, and the other is the log complement part. The timestamp records the time of each log occurrence. Timestamps in different formats can be easily extracted from raw log data during log parsing, as they are regular expressions. A log identifier is a token that identifies multiple processes or message exchanges of the system(Du *et al.* 2017).

Log data X_1,X_2,X_3,X_4....X_n let be created. These log data are T_1,T_2,T_3,T_4....T_n corresponds to log templates. T_K is log parsing method, date(t), time(z), pid(p), type(r), component(b), content(i), templateid(j), template(l), and anomaly(k) performs the separation process.

$$t, z, p, r, b, i, j, l, k = T_K(\mathrm{X}) \tag{1}$$

As a result of the log parsing method;

$$k = \begin{Bmatrix} 0 & Normal \\ 1 & Abnormal \end{Bmatrix} \tag{2}$$

After creating the log templates in Eq.(1), they are transferred to the Node2Vec algorithm. With the embedding vector resulting from this algorithm, training and test data are created from the labeled abnormal data in Eq.(2).

As seen in Table 1, the first part is seen as a timestamp, the other part as a log complement. Thus, some of the log data contains numerical data, and the other part contains verbal data. Each word in the log data can be used as a log keyword or parameter. Log parameters usually consist of IP addresses, MAC information, or user information. Log anomaly detection is generally detecting that the log data is not abnormal. The presence of "INFO" in the log data does not mean that the log data is normal. It is unknown if parsing log data for this is abnormal or not. The purpose of log parsing is to extract meaningful data from raw log data. Thus, using these data, analyzes are made, and models are created.

To automatically analyze the logs, it is necessary to convert them into appropriate formats that can fit textual and machine learning algorithms. To analyze the log data, its unique parts must be determined. As shown in Fig. 2, unique templates were produced by labeling the parts with different similarity ratios in the log records.

081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating

Date	081109
Time	203615
Level	INFO
Component	dfs.DataNode$PacketResponder
Event Template	PacketResponder 1 for block <*> terminating
Parameters	blk_38865049064139660

Figure 2 Log parsing steps for each log row.

Logs are preprocessed during log parsing. The values in the timestamp in Table 2 are also separated as date, time, and PID. Since each log template is different from the other, each template is labeled as TemplateID. Component and content parts were also subjected to separation under a different column.

Architecture of the Proposed Model Algorithm

It is challenging to detect anomalies in log analysis. Because log data consists of both numerical and categorical data. To be able to analyze these data, certain preprocesses are required. Different preprocessing techniques are applied to each study dataset mentioned in Section 2. Thus, the feature is extracted from the data set and made into a vector. Later, this vectorial data set was analyzed with deep learning algorithms, and anomaly detection was made.

In this study, the GLSTM algorithm is proposed. This algorithm consists of two stages. In the first stage, the data was transferred to a graph after converting the data into templates without making attributes from the data set. For this study, the Node2Vec algorithm, one of the graph algorithms, was used. Because it is the most effective algorithm for obtaining a vectorial data set for analyzing data. Experimental tests have proven that this algorithm is suitable and adequate for this study. The second stage is the analysis and classification process. At this stage, LSTM, one of the deep learning algorithms, was used. This algorithm is an iterative deep learning algorithm. It is one of the most preferred algorithms for detecting anomalies in log analysis. As a result of the experimental tests, a high success rate was obtained using Node2Vec and LSTM in anomaly detection.

The structure of the GLSTM architecture is shown in Fig. 3. When the structure is examined, log data from multiple heterogeneous sources is taken, and templates are created. Since the graph

Table 1 Raw log data structure.

Raw Log Data
081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
081109 204005 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.73.220:50010 is added to blk_7128370237687728475 size 67108864
081109 214529 2747 WARN dfs.DataNode$DataXceiver: 10.251.123.132:50010: Got exception while serving blk_3763728533434719668 to /10.251.38.214:
081109 220032 3137 WARN dfs.DataNode$DataXceiver: 10.250.14.196:50010: Got exception while serving blk_-305633040016166849 to /10.251.38.53:

Table 2 Assigning log preprocessing parameters.

NoID	1,2,3,. . . ..
Date	081109 , 081110, ...
Time	203615, 203807, 204005
PID	148, 222, 35, 308, 329, ...
Level	INFO, WARN
Component	dfs.DataNode$PacketResponder, dfs.FSNamesystem, dfs.DataNode$DataXceiver

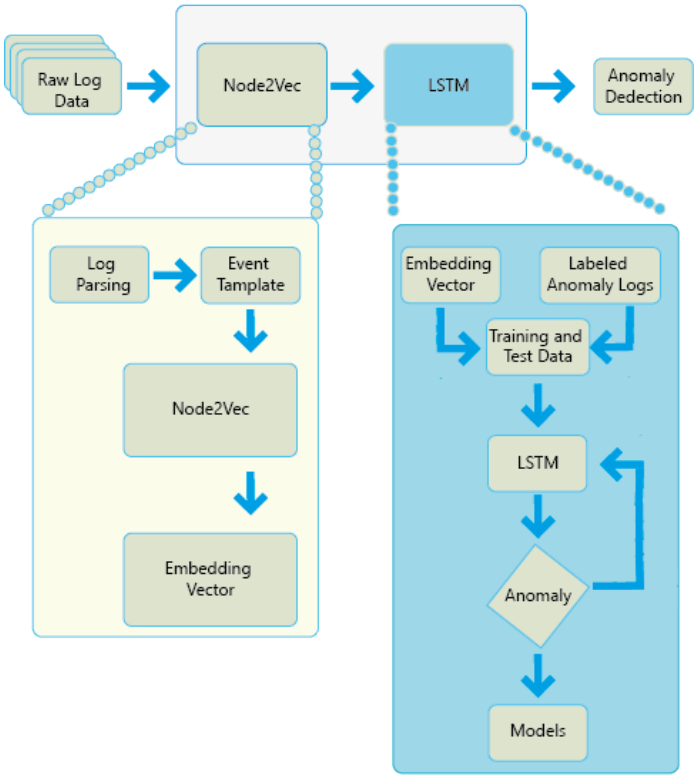

Figure 3 Proposed Model Architecture structure.

algorithm accepts the data set as numerical, the categorical part of the data set of these templates was digitized. Digitization was done by two methods used in the literature. One of them is Label Encoding, and the other is One Hot Encoding. The graph structure was created by giving the digitized data set to the Node2Vec algorithm as an edge and a node. A vectorial result was obtained from this graph structure. By providing this result as an input to the LSTM algorithm, log anomaly detection was made.

Exporting data to graph

The Node2Vec algorithm, one of the graph algorithms, has been developed as an alternative to the word2vec algorithm, a natural language processing algorithm. Although it was designed with natural language processing in mind, this algorithm has been used in more than one area. The approach of this algorithm uses probability to maximize the neighborhood of each node in the network in a d-dimensional feature space. A random walk approach is used to obtain the network neighborhood of the nodes.

The classical search algorithms in the graph are shown in Fig. 4. One of these algorithms is Breadth Priority Sampling (BFS), and the other is Depth Priority Sampling (DFS). It seems that BFS can detect close quarters, whereas DFS can detect distant neighborhoods. With its flexible structure, Node2Vec uses these two approaches together. Probability was used to find neighborhoods by taking a random walk. Semi-supervised operation in unweighted and undirected networks achieved better results than classical search approaches BFS and DFS(Grover and Leskovec 2016).

The structure of the Node2Vec algorithm differs from other algorithms. This algorithm takes four parameters. These are p, q, random walk, and walk length parameters. It is an algorithm that works as a semi-control as optimum results are obtained by

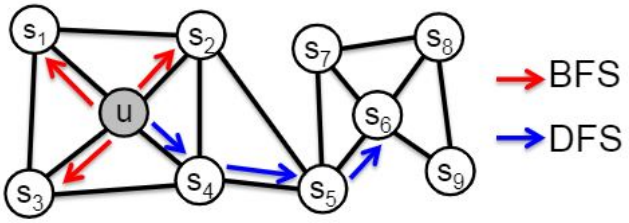

Figure 4 Classical search graph algorithms(Grover and Leskovec 2016)

changing these parameters. Of these parameters, p is the return parameter. It reduces the probability of sampling the previously visited node. The other parameter q is the input-output parameter. With this parameter, previously unvisited nodes are visited. If $q > 1$, the random walk is performed around the more visited node. In this respect, it is similar to the BFS algorithm. If $q < 1$, the random walk visits previously unvisited nodes. In this respect, it is similar to the DFS algorithm.

To get vector data from the Node2Vec algorithm, it needs to be exported to a Graf. In this study, StellarGraph was used because machine learning and deep learning structures are easy to use (CSIRO's Data61 2018). The main reasons for using this graphic structure are; 1. It can be used for visualization and various machine learning, 2. Ability to extract features from nodes and edges, 3. Applicable in big data, 4. Classification of nodes, It can perform many operations, such as easy and applicable. Multiple studies have been conducted on deep learning and machine learning using this graph structure(Rong 2014; Demeester *et al.* 2016; Kipf and Welling 2016).

The following procedure was followed for transferring the data to Graph. Graphs are made up of edges and nodes. Nodes, on the other hand, need to go from a specific source to the destination. TemplateID, which is different for each template, was chosen as the source, and anomaly or normal column was chosen for the target. The remaining columns are used for nodes. In Fig. 5, the edges and nodes of the data transferred to the Graph are shown.

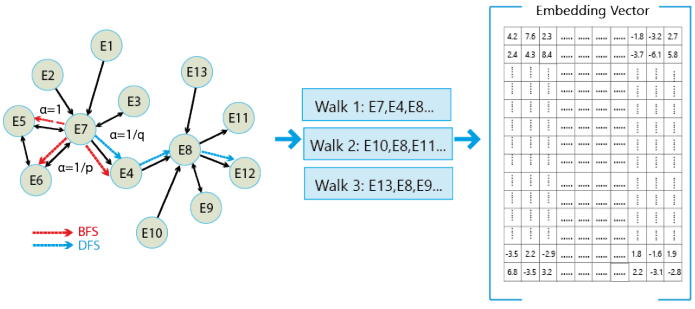

Figure 5 Data exported to graph.

Anomaly Detection

LSTMs are members of repetitive RNNs. RNNs are self-repetitive models, taking sequential data one item at a time. Compared to Markov models, although state-space sets increase, they give better results in the long run due to dependency(Specht 1990; Werbos 1988). LSTMs were developed to eliminate the disadvantages of RNNs. LSTMs work recursively like RNNs, the difference being that they run on different cells with their hidden display.

Fig. 6 shows the use of the LSTM algorithm in this study. Input data is used HDFS verse which Hadoop collects from multiple sources. These data were digitized with 1-hot encoding and label encoding methods. Then, these data are given to the Node2Vec algorithm as an input parameter, and an embedding vector is created as an output. Anomalous log data labeled with this embedding vector is provided as input parameters to the LSTM algorithm. Thus, models that detect abnormal values are created.

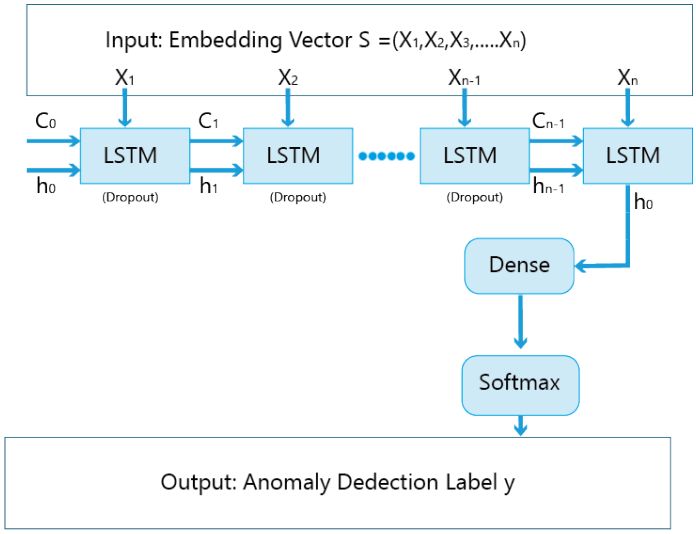

Figure 6 Anomaly detection in the proposed model.

RESULTS AND DISCUSSION

This study proposes a model to analyze log records to detect cyberattacks and to detect the anomalies created by the traces left by the attackers in the log records. HDFS dataset was used to test this model. The HDFS log dataset consists of 11,175,629 logs collected from more than 200 Amazon heterogeneous sources. HDFS log data records operations such as allocation, duplication, and deletion in a specific block using block_id. This dataset comprises 575,061 log blocks and has been labeled 16,838 abnormal by Hadoop's experts. Table 3 gives information about the HDFS data set.

The BGL dataset consists of a comprehensive collection of 4,747,963 logs, meticulously labeled as either anomalous or normal. Among these logs, 348,460 instances have been classified as anomalous. The BGL dataset was obtained from the Blue Gene/L supercomputer, a highly sophisticated computing system employed at Lawrence Livermore National Laboratory (LLNL). With its extensive infrastructure consisting of 128K processors, the Blue Gene/L supercomputer has played a crucial role in generating the BGL dataset for research and academic endeavors (Guo *et al.* 2021).

The IMDb dataset is a collection of film reviews. It comprises 50,000 reviews written by users on the IMDb website for various movies. The dataset includes reviews that have been labeled as positive or negative. The reviews are rated using a rating scale ranging from 1 to 10. Ratings between 1 and 4 are labeled as negative, while ratings between 7 and 10 are labeled as positive. Ratings of 5 and 6 are not included in the dataset. Each film has a maximum of 30 reviews. The IMDb dataset consists of 25,000 positive and 25,000 negative reviews (Tripathi *et al.* 2020).

Table 3 Detail of the datasets.

Dataset	Time	Log Line	Anomaly
HDFS	38,7 hours	11,175,629	16,838(block)
BGL	7 months	4,447,963	348,460(logs)
IMDB	-	50,000	25,000(negative)

Research Questions

Logging collects numerical and textual data of software behavior, such as low memory conditions or attempts to access a file. Log anomaly in modern software engineering is still challenging for three main reasons.

The main reasons for this are;

• Great effort is required for large volumes of logs and thus manual regular expression generation,

• The complexity of the software and, therefore, the variety of event templates,

• Frequency of software updates and hence frequent updating of logging statements.

In this study, templates were created for each row of log records, and these large-volume logs were made regularly by reducing their size using templates. Then, the embedding vector was created by establishing a relationship between these templates with the Node2Vec algorithm. The model was trained with LSTM, and anomaly detection was performed in the newly created log template.

Evaluation Metrics

A confusion matrix was used for the performance evaluation of experimental studies conducted to classify HDFS, BGL and IMDB datasets. In these experimental studies, the data was randomly partitioned into training, validation, and test sets, with 70% of the dataset allocated for training and 15% each for testing and validation. This data splitting strategy was employed to ensure reliable outcomes in the experiments. Performance metrics such as accuracy, sensitivity, specificity, precision, and F-score of the model were calculated using the confusion matrix. The calculation of these metrics is given in Eq. 3, 4, 5 and 6 mathematically.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$F1 = \frac{2 * TP}{2 * TP + FP + FN} \tag{6}$$

In the experimental studies, in the first stage, large volumes of log data were transformed into templates to reduce their size and make them regular. The Node2Vec algorithm was used to establish a relationship between these templates and to train the model with the deep learning algorithm. Then, model training was performed with the LSTM algorithm for anomaly detection. In the model created with this dataset, the LSTM input layer consists of 128, the hidden layer 64, and the output layer consists of 1 neuron to obtain the normal or abnormal result. As a result of the experimental study, an accuracy rate of 97.01% was obtained with the proposed model.

The performance results obtained with the proposed model are given in Table 4. The results vary depending on the datasets. In the tests conducted on the HDFS dataset, the LSTM method achieved an accuracy rate of 97.01%. The Bi-LSTM method followed closely with an accuracy rate of 96.98%. The GRU method demonstrated the highest performance with an accuracy rate of 98.15%. In the tests conducted on the BGL dataset, the LSTM method had an accuracy rate of 81.56%. The Bi-LSTM method performed slightly better with an accuracy rate of 84.21%. The GRU method exhibited the best performance with an accuracy rate of 86.44%. In the tests conducted on the IMDB dataset, the LSTM method achieved an accuracy rate of 97.40%, while the Bi-LSTM method outperformed with an accuracy rate of 98.54%. The GRU method showed the highest performance with an accuracy rate of 98.89%. Based on these experimental results, it can be observed that the GRU, Bi-LSTM, and LSTM methods perform differently on different datasets.

The Fig. 7 illustrates the progression of accuracy rates during the training and validation processes. The training accuracy curve represents the accuracy rate achieved on the training dataset, while the validation accuracy curve reflects the accuracy rate on the validation dataset. At the beginning of the graph, both the training and validation accuracy rates are low. However, as the training process progresses, the accuracy rates increase and eventually converge. This indicates that the model performs well on the training dataset and also provides good results on the validation dataset. Analyzing this graph is important to evaluate the model's performance during the training process and demonstrate the absence of issues such as overfitting or underfitting.

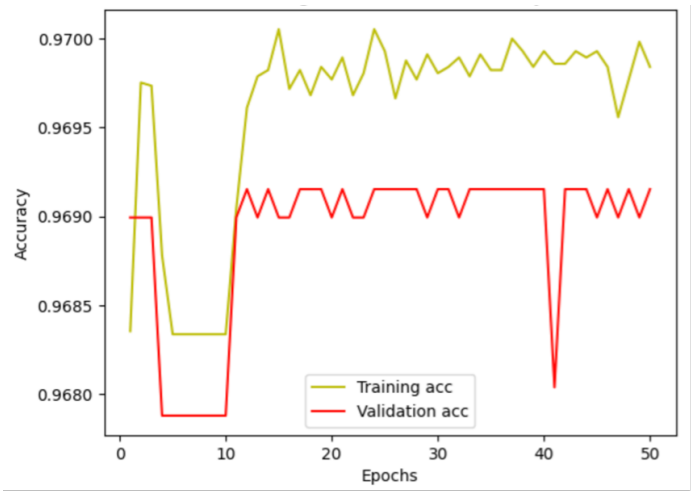

Figure 7 Training and Validation Accuracy Performance Curves.

Table 4 Performance results of the proposed model.

Datasets	Algorithm	Accuracy	Precision	Recall	F1_score	Average train time (second)
HDFS	LSTM	97.01	97.23	96.06	84.25	6.60
	Bi-LSTM	96.98	97.40	97.18	86.89	
	GRU	98.15	98.10	98.55	88.42	
BGL	LSTM	81.56	81.76	88.31	81.54	5.46
	Bi-LSTM	84.21	86.89	85.18	85.79	
	GRU	86.44	87.34	88.29	91.52	
IMDB	LSTM	97.40	95.99	98.18	95.89	7.21
	Bi-LSTM	98.54	97.44	97.39	96.22	
	GRU	98.89	97.49	98.28	97.36	

Fig. 8 illustrates the changes in training loss and validation loss during the training process. The training loss curve represents the loss on the training dataset, while the validation loss curve reflects the loss on the validation dataset. The graph shows that the training loss decreases over time, indicating that the model is learning and improving its performance. Initially, the validation loss also decreases, but at a certain point, it starts to increase again. This situation indicates that the model is not overfitting to the training data.

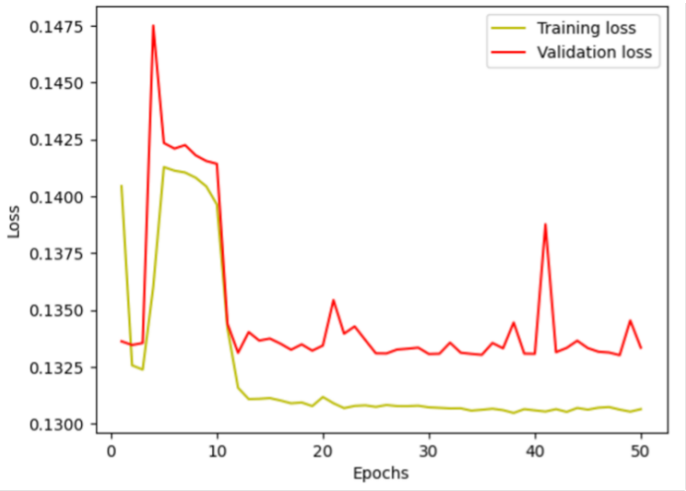

Figure 8 Training and Validation Loss Performance Curves.

The Confusion Matrix is given in Fig. 9, which shows the success status due to the tests performed in this study. This graph calculates the efficiency of the actual and predicted values. The important thing is that the estimated values obtained after training our model were compared with the actual values, and their accuracy was determined. This graph shows how many anomalies the actual anomaly detected after the model was trained. Thus, this graph shows that our model has achieved high success.

Two useful tools, AUC curves, are used to measure the outcome of experiments performed accurately. These curves are used to

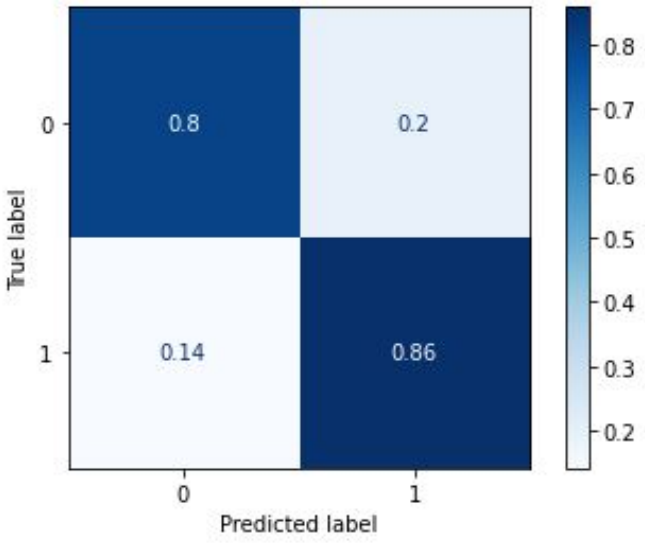

Figure 9 The confusion matrix of the experimental results of the proposed model.

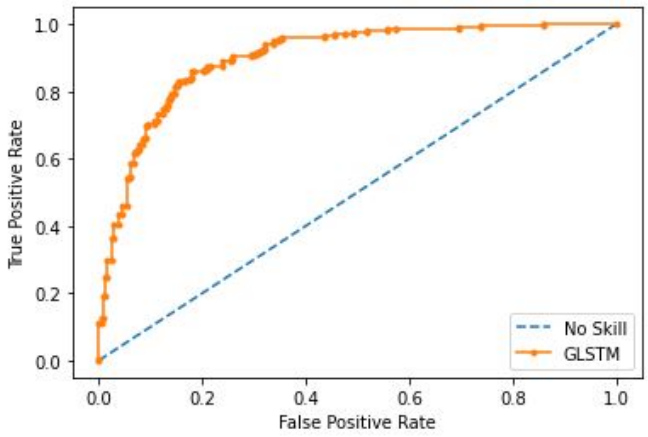

Figure 10 The graph of the AUC Curve.

eliminate two different errors. One of them is HPs. This error gives results as if there is an event when there is no event. The other is FN. This error also produces erroneous results because it does not detect the event when there is an event. Due to these two errors, the results of the experiments are not clearly understood. To avoid this, AUC curves are used.

$$TruePositiveRate = \frac{TP}{TP + FN} \tag{7}$$

$$TrueNegativeRate = \frac{TN}{FN + TN} \tag{8}$$

Two important ratios are calculated in the AUC curve. One of them is the True Positive Ratio shown in Eq. 7. The other is the True Negative Ratio shown in Eq. 8. Fig. 10 shows the graph of the AUC Curve. Smaller values on the graph's x-axis indicate lower false positives and higher true negatives. The graph's y-axis also shows larger values, i.e., higher true positives and lower false negatives. This shows that a good model shows a value higher than 0.5; the part is shown with dashed lines in the graph, that is, the threshold value. This shows that the model gives good results.

Another graph that measures the model accurately is the Precision – Accuracy graph. These curves are also called Sensitive Recall Curves. The precision shows how well the positive part of the model predicts, as shown in Eq. 8. The accuracy is shown in Eq. 3. This allows for a more accurate estimation of true positives. Fig. 11 shows the Precision vs. Accuracy graph. The integral of the area under the curve shows how accurately and accurately the model works.

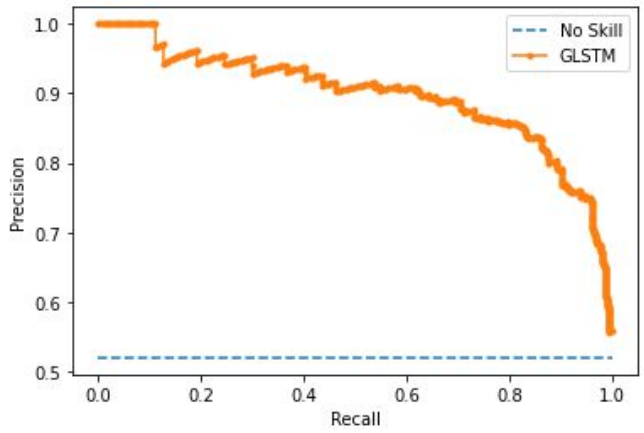

Figure 11 Precision-Accuracy Plot of the proposed model.

Competing Models

This study was carried out according to the method used, the dataset used, and the comparisons' accuracy with the previous run. Considering these criteria, comparisons for this use with other businesses are given in Table 5. LogAnomaly (Rodriguez *et al.* 1999) uses the same dataset as our proposed study and the same deep learning methods in the developed method. With LogAnomaly, primarily synonyms and antonyms were detected in the log data with Word2Vec. One sample for each log information is incorrect. These templates were then transferred to a vector and analyzed with LSTM. LogAnomaly faces significant challenges as log records consist of numeric and textual structures. Since the method we propose converts both numeric and textual data into graphs, it turned out to be in the size of such dimensions, and in fact, a better result was obtained than LogAnomay. DeepLog (Du *et al.* 2017) generates a key for each log information with a natural language processing method, and a vector result is obtained with the corresponding key. Anomaly data were made using this vectored LSTM. This method has difficulties analyzing numerical parts of log data at a certain level. Since the method we proposed analyzes using each feature of the whole data set, it achieved a more successful result despite using the same dataset and using this method. To reduce log anomaly, Bi-LSTM and PCA retentions (Meng *et al.* 2019b) were used with a dataset similar to our proposed work. With this work, the dataset was first separated and then made into templates. Then, it was converted into vectorial form with digitization and normalization processes. Although the examples we suggested used the same dataset, the model we did not particularly recommend was more successful than the results obtained with PCA. As a result, our proposed method has obtained more successful results than many previously applied models and evaluates that it can be used more effectively in daily anomalies.

In this study, graph structure was used instead of the NLP technique used in many studies. Node2vec from the graph algorithm is used. This algorithm was developed as an alternative to word2vec algorithms. In this study, it has been shown by tests that it achieves a better result than other algorithms in terms of decomposing logs and feature extraction.

To train deep learning network models and achieve high success in log anomaly detection, it should be brought to the level to be given as input to the model, especially after the log parsing process. In this study, the node2vec output vector was given to LSTM as input data, and 97.01% success was achieved. A better result was obtained than the methods using the NLP technique.

CONCLUSION

This study aims to make a large number of logs obtained from different sources in complex networks and uniformly contain different features and detect anomalies from them. The fact that the logs consist of huge and different data makes detecting fast and effective anomalies very difficult. For this reason, to process these different log data effectively and quickly, in this study, logs in different structures were turned into a template and then converted into a graph structure to obtain the relationships between these templates. Node2Vec, a graph algorithm, was used for graph transformation. The embedding vector of the log templates is obtained from this transformation. The obtained data containing these vector anomaly tags are divided into 70% training and 30% test data for the deep learning algorithm. These data were trained and tested using the LSTM algorithm, one of the deep learning methods. As a result of the tests, our Graf-based LSTM model, which we recommend, has achieved successful results with an accuracy of 97.01%.

Intrusion detection systems utilize the analysis of log data to effectively detect anomalies. However, detecting anomalies quickly and effectively in large and heterogeneous log data can be challenging. To address this difficulty, this study proposes the GLSTM (Graph-based Long Short-Term Memory) framework, a graph-based deep learning model that analyzes log data to detect cyber-attacks rapidly and effectively. The framework involves standardizing the complex and diverse log data, training this data on an artificial intelligence model, and detecting anomalies. Initially, the complex and diverse log data is transformed into graph data using Node2Vec, enabling efficient and rapid analysis on the artificial intelligence model. Subsequently, these graph data are

Table 5 Comparison of the proposed model with other payments.

Authors	Method	Datasets	Acc(%)
2019, Weibin Meng et al.(Rodriguez *et al.* 1999)	LSTM,Word2Vec	BGL,HDFS	96.00
2017,Min Du et al. (Du *et al.* 2017)	LSTM,tamplate2Vec	BGL,HDFS	92.00
2022, Zhang Yue et al.(Meng *et al.* 2019b)	Bi-LSTM,PCA	HDFS	95.60
2023, Proposed Method	LSTM,BGL,IMDB,Node2Vec	HDFS	97.01

trained using LSTM (Long Short-Term Memory), Bi-LSTM, and GRU (Gated Recurrent Unit) deep learning algorithms. The proposed framework is tested using Hadoop's HDFS dataset, collected from different systems and heterogeneous sources, as well as the BGL and IMDB datasets. Experimental results on the selected datasets demonstrate high levels of success.

Limitations of this study should be considered:

Data Diversity: Although this study was tested with Hadoop's HDFS dataset, its ability to generalize to datasets with greater diversity from various networks and systems may be limited. Specific anomalies based on different data types or sources could pose challenges. Data Size: Working with large datasets can be limited by computational resources and memory requirements. This study may provide limited insights into handling larger datasets. Feature Engineering: Data transformations and representation may pose challenges in feature engineering. Ensuring that data is accurately and meaningfully represented may not always be guaranteed. Training Data: The success of this study may be dependent on the specific datasets used and the training data. Results may vary with different datasets or data splitting strategies. Model Selection: This study employed specific deep learning algorithms like LSTM, Bi-LSTM, and GRU. The impact of these algorithms on model performance should be taken into account. Exploration of other deep learning methods may be warranted. Real-World Applications: The extent to which the study's results can be applied to real-world applications, generalize to specific network structures or systems, may require further investigation. These limitations should be considered for a better understanding of the study's findings and the real-world applicability of the model.

Availability of data and material

Not applicable.

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical standard

The authors have no relevant financial or non-financial interests to disclose.

LITERATURE CITED

Ahmed, M., A. N. Mahmood, and M. R. Islam, 2016 A survey of anomaly detection techniques in financial domain. Future Generation Computer Systems **55**: 278–288.

Alaca, Y. and Y. Çelik, 2023 Cyber attack detection with qr code images using lightweight deep learning models. Computers & Security **126**: 103065.

Church, K. W., 2017 Word2Vec. Natural Language Engineering **23**: 155–162.

CSIRO's Data61, 2018 StellarGraph Machine Learning Library.

Demeester, T., T. Rocktäschel, and S. Riedel, 2016 Lifted rule injection for relation embeddings. EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings pp. 1389–1399.

Du, M., F. Li, G. Zheng, and V. Srikumar, 2017 DeepLog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the ACM Conference on Computer and Communications Security pp. 1285–1298.

Elbasani, E. and J. D. Kim, 2021 LLAD: Life-Log Anomaly Detection Based on Recurrent Neural Network LSTM. Journal of Healthcare Engineering **2021**.

Farzad, A. and T. A. Gulliver, 2019 Log Message Anomaly Detection and Classification Using Auto-B/LSTM and Auto-GRU pp. 1–28.

Gogoi, P., D. K. Bhattacharyya, B. Borah, and J. K. Kalita, 2011 A survey of outlier detection methods in network anomaly identification. The Computer Journal **54**: 570–588.

Grover, A. and J. Leskovec, 2016 node2vec: Scalable Feature Learning for Networks .

Guo, H., S. Yuan, and X. Wu, 2021 Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE.

He, S., P. He, Z. Chen, T. Yang, Y. Su, *et al.*, 2020 A Survey on Automated Log Analysis for Reliability Engineering. arXiv preprint arXiv:2009.07237 .

Hochreiter, S. and J. Schmidhuber, 1997 Long Short-Term Memory. Neural Computation **9**: 1735–1780.

Kipf, T. N. and M. Welling, 2016 SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS. Technical report.

Li, H. and Y. Li, 2020 LogSpy: System Log Anomaly Detection for Distributed Systems. Proceedings - 2020 International Conference on Artificial Intelligence and Computer Engineering, ICAICE 2020 pp. 347–352.

Li, Y., Y. Zheng, H. Zhang, and L. Chen, 2015 Traffic prediction in a bike-sharing system. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 1–10.

Lu, S., X. Wei, Y. Li, and L. Wang, 2018 Detecting anomaly in big data system logs using convolutional neural network. Proceedings - IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, IEEE 16th International Conference on Pervasive Intelligence and Computing, IEEE 4th International Conference on Big Data Intelligence and Computing and IEEE 3 pp. 159–165.

Makanju, A. A. O., A. N. Zincir-Heywood, and E. E. Milios, 2009

Clustering event logs using iterative partitioning. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1255–1264.
Meng, W., Y. Liu, Y. Zhu, S. Zhang, D. Pei, *et al.*, 2019a Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. IJCAI International Joint Conference on Artificial Intelligence **2019-Augus**: 4739–4745.
Meng, W., Y. Liu, Y. Zhu, S. Zhang, D. Pei, *et al.*, 2019b Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. IJCAI International Joint Conference on Artificial Intelligence **2019-Augus**: 4739–4745.
Rodriguez, P., J. Wiles, and J. L. Elman, 1999 A recurrent neural network that learns to count. Connection Science **11**: 5–40.
Rong, X., 2014 word2vec Parameter Learning Explained pp. 1–21.
Schindler, T., 2017 Anomaly Detection in Log Data using Graph Databases and Machine Learning to Defend Advanced Persistent Threats. Technical report.
Sigelman, B. H., L. Andr, M. Burrows, P. Stephenson, M. Plakal, *et al.*, 2010 Dapper , a Large-Scale Distributed Systems Tracing Infrastructure. Google Research p. 14.
Specht, D. F., 1990 Probabilistic neural networks. Neural networks **3**: 109–118.
Studiawan, H., C. Payne, and F. Sohel, 2017 Graph clustering and anomaly detection of access control log for forensic purposes. Digital Investigation **21**: 76–87.
Tripathi, S., R. Mehrotra, V. Bansal, and S. Upadhyay, 2020 Analyzing sentiment using imdb dataset. In *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 30–33, IEEE.
Vaarandi, R., 2003 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.
Wang, M., L. Xu, and L. Guo, 2018 Anomaly detection of system logs based on natural language processing and deep learning. 2018 4th International Conference on Frontiers of Signal Processing, ICFSP 2018 pp. 140–144.
Wang, X., D. Wang, Y. Zhang, L. Jin, and M. Song, 2019 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.
Werbos, P. J., 1988 Generalization of backpropagation with application to a recurrent gas market model. Neural networks **1**: 339–356.
Yan, X., W. Zhou, Y. Gao, Z. Zhang, J. Han, *et al.*, 2015 PADM: Page rank-based anomaly detection method of log sequences by graph computing. Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom **2015-Febru**: 700–703.

How to cite this article: Alaca, Y., Celik, Y., and Goel, S. Anomaly Detection in Cyber Security with Graph-Based LSTM in Log Analysis. *Chaos Theory and Applications*, 5(3), 188-197, 2023.