RESEARCH ON LOG ANALYSIS TECHNIQUES TO DETECT NETWORK ATTACKS

Phong Pham Huu
Institute of Information and Communication Technology
LeQuyDon technical university
huuphonga3@gmail.com

Viet Hung Nguyen*
Institute of Information and Communication Technology
LeQuyDon technical university
hungnv@lqdtu.edu.vn

Ngoc Anh Tran
2nd Brigade Command 86 Ho Chi Minh Vietnam
anhtn69@gmail.com

Minh Hieu Nguyen
Institute of Information and Communication Technology
LeQuyDon technical university
hieunguyenmta@gmail.com

ABSTRACT

System logs play a vital role in upholding information security by capturing events to address potential risks. Numerous research initiatives have harnessed log data to create machine learning models geared towards spotting unusual activities within systems. In this pragmatic study, we introduce an innovative approach to detecting anomalies in log data, employing a three-step process encompassing preprocessing, advanced natural language processing (NLP) utilizing BERT, and a custom 1D-CNN classification model. During the preprocessing phase, we tokenize the data and eliminate non-essential elements, while BERT enriches log message representations. Our Sliding Window and Overlapping Mechanism ensures consistent input dimensions. The 1D-CNN model extracts temporal features for robust anomaly detection. Empirical findings on HFDS, BGL, Spirit, and Thunderbird datasets illustrate that our method outperforms prior approaches in identifying network attacks.

CCS CONCEPTS

• **Security and privacy** → Intrusion/anomaly detection and malware mitigation; Intrusion detection systems; • **Additional Keywords and Phrases: Logs, attack detection, 1D-CNN.**;

ACM Reference Format:
Phong Pham Huu, Viet Hung Nguyen*, Ngoc Anh Tran, and Minh Hieu Nguyen. 2023. RESEARCH ON LOG ANALYSIS TECHNIQUES TO DETECT NETWORK ATTACKS. In *The 12th International Symposium on Information and Communication Technology (SOICT 2023), December 07, 08, 2023, Ho Chi Minh, Vietnam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3628797.3628943

1 INTRODUCTION

Modern software systems generate massive volumes of log data, containing valuable information about system behavior and potential anomalies. Effectively analyzing these logs is crucial for maintaining system reliability and security. In this paper, we propose a novel method for anomaly detection in log data by leveraging a combination of preprocessing techniques, advanced natural language processing (NLP) models, and convolutional neural networks (CNNs).

The increasing complexity of software systems makes traditional rule-based anomaly detection methods less effective. Our approach addresses this challenge by introducing a three-stage process. First, we preprocess raw log messages, which involves tokenization and the removal of non-informative elements like numbers and punctuation. This initial step sets the foundation for capturing the essential content of the log messages.

To enhance the representation of log messages, we employ the Bidirectional Encoder Representations from Transformers (BERT) model, a cutting-edge NLP model. BERT's skill at understanding words in sentences helps us accurately catch the real meaning in log messages. This transformation creates special vectors that keep the important details needed for finding anomalies.

The core of our approach lies in the 1D-CNN classification model. This architecture is specifically designed for anomaly detection and operates on the organized log sequences generated by the previous stage. The architecture employs multiple convolutional layers, progressively extracting intricate temporal features. These features are then refined through fully connected layers, leading to a robust classification process. The architecture's use of Batch Normalization and Dropout layers enhances training convergence and prevents overfitting.

In summary, our method presents a comprehensive approach to anomaly detection in log data. By combining preprocessing, advanced NLP techniques, and a tailored CNN architecture, we aim to achieve accurate and efficient identification of anomalies within complex software systems. The subsequent sections detail each stage of our proposed method and present experimental validation, highlighting its effectiveness in anomaly detection.

2 RELATED WORK

Over the past few years, there has been a noticeable surge in the utilization of neural network models for log anomaly detection.

Du et al. [1] introduced DeepLog as a groundbreaking method that employs LSTM (Long Short-Term Memory) for spotting anomalies in logs. This was the first instance where LSTM was used for log anomaly detection. Moreover, DeepLog is innovative in its

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SOICT 2023, December 07, 08, 2023, Ho Chi Minh, Vietnam
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0891-6/23/12…$15.00
https://doi.org/10.1145/3628797.3628943

approach, as it anticipates anomalies using a forecasting method, a technique that has since been adopted in other research due to its versatility and efficacy.

In the context of log anomaly detection, DeepLog [1] learns log patterns from the sequential relationships among log events, representing each log message with its corresponding event index. To enhance the understanding of log semantics, LogAnomaly, proposed by Meng et al. [2], introduces the concept of template2Vec. This approach captures word meanings within log templates by accounting for synonyms and antonyms. For instance, words like "down" and "up" are distinctly represented due to their opposing meanings. Similar to DeepLog, LogAnomaly utilizes LSTM-based predictive modeling to identify anomalies. This current study explores the potential performance enhancement of DeepLog by incorporating log semantics, following the approach of LogAnomaly.

Farzad et al. [3] introduced a significant advancement in log-based anomaly detection by incorporating an autoencoder in conjunction with an isolation forest. This novel method encompassed employing the autoencoder for feature extraction, with the isolation forest being responsible for anomaly detection using the features derived from the autoencoder. Through their research, the authors convincingly showcased that this amalgamation outperforms the straightforward application of the isolation forest to raw log data. This combined methodology holds promise in enhancing anomaly detection accuracy and robustness within log analysis systems.

LogRobust addresses a common challenge in log anomaly detection, as pointed out by Zhang et al. [4].They observed that many existing methods tend to fall short of achieving their anticipated performance in real-world scenarios. One primary reason for this discrepancy lies in the closed-world assumption that most methods operate under. To combat the issue of log instability, Zhang et al. introduced LogRobust. This approach seeks to extract semantic information from log events by utilizing readily available word vectors. Notably, LogRobust is among the pioneering efforts to consider the semantic aspect of logs, following in the footsteps of the work done by Meng et al. [5].Through the utilization of word vectors, LogRobust strives to augment the flexibility and efficiency of log anomaly detection techniques, particularly when confronted with evolving log data.

Lu et al. [6] conducted pioneering research into the applicability of CNNs for log-based anomaly detection. They introduce a cutting-edge method that leverages Convolutional Neural Networks (CNNs) to autonomously learn intricate event relationships from these logs, resulting in highly accurate anomaly detection. Their deep neural network architecture comprises logkey2vec embeddings, three 1D convolutional layers, a dropout layer, and max-pooling, achieving an impressive accuracy rate of up to 0.99. Notably, this CNN-based approach outperforms alternative methods such as Long Short-Term Memory (LSTM) and Multilayer Perceptron (MLP) when applied to Hadoop Distributed File System (HDFS) logs. This research underscores the potential of CNNs in log analysis for anomaly detection within the context of big data, offering valuable insights to the field.

NeuralLog, developed by Le et al.[7], offers a novel log anomaly detection approach that is highly effective and efficient on real-world data. A key innovation is avoiding explicit log parsing and instead directly encoding the raw log messages into semantic vector representations. This allows NeuralLog to preserve more information compared to methods that rely on potentially error-prone parsers. Specifically, NeuralLog transforms each log message into a vector that captures semantics within the message itself as well as relationships between different log events. These log vectors are fed into a Transformer-based classifier powered by multi-head self-attention [8]. This architecture is well-suited for modeling contextual patterns in log sequences. By operating directly on raw log data and leveraging Transformers' strength in capturing textual semantics, NeuralLog can accurately detect anomalies without log parsing. Evaluations demonstrate strong performance on real-world systems logs, highlighting the promise of this log encoding and Transformer-based approach. NeuralLog delivers an innovative, highly effective anomaly detection solution without relying on parser-induced information loss.

3 BACKGROUND

In this section, the background knowledge needed to build the proposed model will be presented in detail.

3.1 Pre-processing

Log data tends to be unstructured, requiring preprocessing before inputting into deep learning models. The common approach employs log parsers to extract distinct event types and parameter values from each log line. However, as discussed by Le et al., reliance on log parsers can undermine anomaly detection performance. One issue is that log formats evolve dynamically during software updates, introducing new events unrecognized by parsers. Another is that parsing inevitably loses semantic information, like original word ordering, that provides useful context. These parser-induced errors introduce inaccuracies and ambiguity, impairing anomaly detectors' ability to learn log patterns. The challenges highlight the need for more robust parsing methods to minimize information loss and confusion about log event semantics. Ideally, preprocessing should maximize retention of original message details in a machine-understandable representation. Advances here would significantly improve deep learning systems' leverage of log data characteristics for more effective anomaly detection and monitoring. Robust log parsing remains an open research problem. Overall, existing parsers are a major source of uncertainty, detracting from anomaly detection performance. Developing parsing techniques that better handle log evolution and preserve semantics is critical for fully unlocking the value of log data.

3.2 WordPiece Tokenization

Tokenization, a foundational technique in natural language processing (NLP), breaks text into smaller units called tokens, which can be words or subword units. It's a vital initial step in NLP tasks, like text analysis and machine learning, facilitating efficient data processing and enabling machines to understand and work with human-readable text. Tokenization also standardizes text, aiding in the analysis of large text volumes while maintaining linguistic and syntactic context.

Wordpiece is a subword tokenization technique commonly associated with models like BERT (Bidirectional Encoder Representations from Transformers). It works by breaking down words into

smaller subword units called "wordpieces." Unlike fixed-size subword units like characters or syllables, wordpieces are dynamic and can represent both whole words and parts of words. This flexibility allows models to effectively handle morphologically rich languages and rare or out-of-vocabulary words by encoding them as combinations of subword units. Wordpiece tokenization has become a crucial component in modern NLP models, improving their ability to capture fine-grained linguistic information and adapt to various languages and text types.

WordPiece is selected for its adeptness at managing OOV words and curtailing vocabulary size. Compared to alternative tokenization methods, WordPiece emerges as a more proficient strategy.

3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a groundbreaking neural network model developed by Google in 2018 for natural language processing. Based on the Transformer architecture, BERT leverages bidirectional training of encoder stacks to learn contextual word representations. This differs from previous models like ELMo and GPT that only used unidirectional context. BERT is pre-trained on enormous unlabeled corpora using two novel unsupervised tasks: masked language modeling and next sentence prediction. This pre-training enables BERT to learn general language representations that can effectively transfer to downstream NLP tasks through fine-tuning. One key capability is generating semantic vector representations for arbitrary sentences or documents. BERT splits input text into subword tokens, passes them through the pretrained encoders, and outputs corresponding contextual vector embeddings. These rich BERT embeddings capture syntactic and semantic characteristics, as demonstrated by their effectiveness when used to initialize models for tasks like question answering and sentiment analysis. The contextual representations produced by BERT have become the standard in many NLP applications. (Figure 1)

BERT's strength lies in its encoder stack of transformer architecture, allowing it to learn language patterns for specific tasks, thus empowering models trained using supervised learning. While the BASE model establishes a performance baseline, the LARGE model achieves remarkable results. With 110M and 340M parameters respectively, BERTBASE and BERTLARGE continue to spearhead advancements in language processing and understanding.

3.4 1D-CNN

In the context of log-based anomaly detection, the selection of the appropriate deep learning architecture plays a pivotal role in the overall effectiveness of the method. When considering different options, such as Long Short-Term Memory (LSTM) and Transformer architectures, we opted for the 1D Convolutional Neural Network (1D-CNN) as the core component of our approach due to a combination of factors that make it uniquely suited for this task.

One of the standout features of 1D-CNNs is their proficiency in efficiently processing sequential data. Logs, comprising sequences of events, are inherently sequential in nature, and anomalies often manifest as local irregularities in these sequences. 1D-CNNs excel in capturing these local patterns, making them particularly adept at identifying anomalies in log data based on the immediate context.

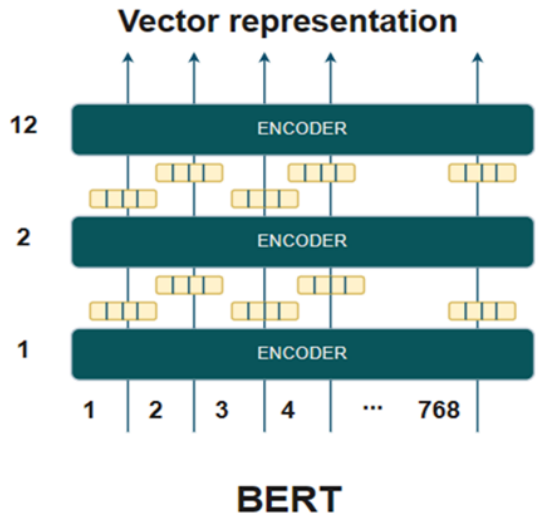

Figure 1: Vector generation architecture

1D-CNNs offer the advantage of parallel processing, enabling them to handle multiple subsequences concurrently. This parallelism significantly accelerates the training process, an essential factor for dealing with the large-scale log data typically encountered in real-world applications.

One-dimensional convolutional neural networks (1D CNNs) are a specialized neural network architecture designed for processing sequential data such as time series or text sequences [9]. Unlike 2D CNNs commonly used for computer vision tasks, 1D CNNs leverage convolutional layers that convolve 1D filters across the input sequence data. This allows them to learn local patterns and temporal dependencies within the sequence while preserving the ordering of data points. After convolution, additional operations like non-linear activations and 1D pooling extract hierarchical features from the sequence. Overall, 1D CNNs' capability to capture temporal relationships and extract meaningful patterns from ordered sequence data makes them well-suited for supervised learning problems involving time series or sequence prediction tasks. Their optimization for ordered, time-based data and ability to model sequences while extracting layered feature representations drive strong performance on diverse sequence modeling and time series analysis problems.

4 PROPOSED METHOD

The proposed anomaly detection method has three main stages: preprocessing, log message encoding, and 1D CNN classification. In preprocessing, non-essential symbols are removed to normalize the log messages. Next, each cleaned message is encoded into a semantic vector representation using BERT, retaining crucial lexical and semantic information. The log sequence then goes through a sliding window with overlap to create fixed-length sequences for CNN input, capturing local temporal patterns. Finally, a 1D CNN model performs anomaly detection on the log sequence representations, learning predictive patterns and detecting deviations. By avoiding message parsing, leveraging BERT's semantic encoding,

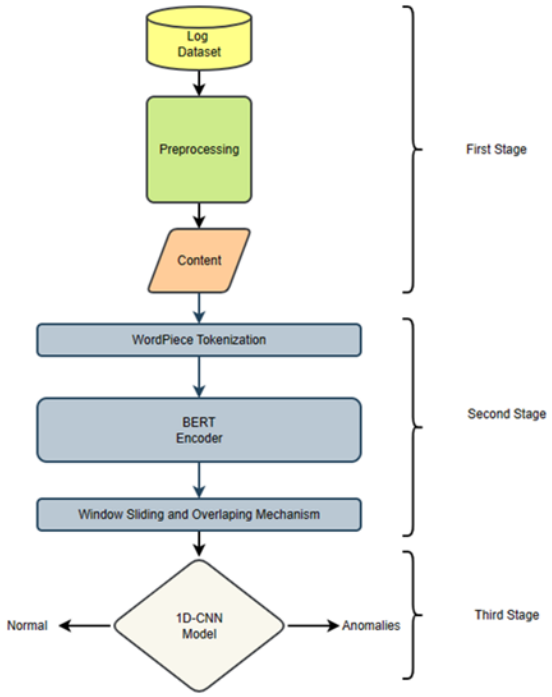

Figure 2: Workflow Approach

and using 1D convolutions to model log sequence dynamics, this workflow aims to improve anomaly detection performance without losing vital log data details. The combination of preprocessing, robust encoding, windowing and 1D CNN classification provides an integrated approach to detect anomalous log events. The complete workflow is shown in Figure 2 below.

4.1 Preprocessing

Similar to various software components, logs experience a continuous process of evolution. This evolution arises from the frequent updates and modifications that developers make to the source code, including the logging statements, resulting in changes to the log data. Numerous conventional methods, such as IM, PCA, and SVM, traditionally involved the transformation of log messages into log events through log parsing. However, this approach often led to semantic misunderstanding. As a result, existing anomaly detection techniques that rely on log events have struggled to deliver satisfactory results, primarily due to the inherent flaws in log parsing methods. Hence, our approach, instead of relying on log parsing for anomaly detection based on log events, directly utilizes raw logs. This approach preserves all the information, even when log messages undergo changes during the development and maintenance phases.

The first step in building our model is preprocessing the log data. This involves tokenizing the log messages into individual words using common delimiters like whitespace, colons, and commas. The log message gets split into a collection of word tokens. Next, all letters are converted to lowercase, and non-character tokens are removed from the word set. Non-characters include operators, punctuation, and numbers, which usually represent variables in the log rather than informative keywords. These tokens lack value for analysis and are excluded. The goal of this preprocessing is to break the log message into focused words, standardize them, and filter out non-essential variable components. This leaves a clean collection of informative keywords as input for subsequent embedding and modeling. [7] For instance, the input log message: '- 1117838570 2005.06.03 R02-M1-N0-C:J12-U11 2005-06-03-15.42.50.363779 R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction cache parity error corrected' is preprocessed into: 'ras kernel info instruction cache parity error corrected'.

4.2 Log Message Representation and Transformation

The process of encoding and transforming the raw log messages involves several key steps. First, the textual content of each log is tokenized into words and symbols for processing. Next, the BERT representation model is used to encode these tokenizations into semantic vector embeddings that capture the contextual meaning within each message. The log sequence vector representations then go through a sliding window with overlaps to convert the variable-length embeddings into fixed-size log samples. This windowing organizes the sequence into consistent, uniform sequences while preserving local temporal ordering. Together, the thoughtful tokenization, BERT semantic encoding, and sliding window conversion transform the raw log text into structured vector representations that retain crucial semantics and temporal characteristics. This facilitates effective downstream anomaly detection by representing the logs in a machine-understandable format amenable to further modeling.

4.2.1 Tokenization. Our work adopts the WordPiece tokenization method [10], [11], a widely utilized approach in recent language modeling studies [12], [13], [14]. This technique effectively handles out-of-vocabulary words by iteratively expanding the vocabulary to encompass commonly occurring subwords. The use of WordPiece reduces the vocabulary size compared to other tokenization methods, mitigating the issue of OOV words.

4.2.2 Log Message Representation. After preprocessing, each log message consists of a collection of words and subwords from tokenization. Traditional embedding methods like Word2Vec [15] capture absolute word meanings, but ignore context. To address this limitation, we leverage BERT [16], a deep pre-trained language model, to derive contextualized semantic representations.

This decision to incorporate BERT is grounded in the need for a contextual understanding of log content. Log messages often contain words with multiple meanings, and their interpretation hinges on the surrounding context. BERT's pre-trained model, with its deep comprehension of linguistic nuances and semantics, is well-suited to this task. By using BERT for feature extraction, our research aims to enhance the accuracy and effectiveness of log analysis and anomaly detection. Thus, the inclusion of BERT is a pivotal component of our method, significantly contributing to its success in log message analysis.

The tokenized words and subwords are fed into the base BERT model [17], consist of 12 transformer encoder and 768 hidden units each, which outputs a fixed-length vector for each element. We average these individual word and subword vectors to produce a single vector summarizing the full log message. This condensed representation encapsulates the semantic meaning of the log event .

BERT handles out-of-vocabulary words by breaking them into subword pieces and embedding them based on their components. Additionally, BERT's positional embeddings and self-attention allow it to incorporate contextual cues about word significance within the message. Overall, BERT provides a robust semantic encoding of log messages, handling OOV terms and using self-attention to account for word context. Averaging the constituent BERT vectors yields a rich representation summarizing the log event for further analysis.

4.3 Sliding Window and Overlapping Mechanism

The Sliding Window and Overlapping Mechanism constitute integral components for converting vector representations of dimensions (768,) into organized log sequences with consistent dimensions of (20,768). This process is designed to enable uniform input dimensions for subsequent processing phases.

We achieve this through a Sliding Window technique with a window size of (20,768), systematically traversing preprocessed log messages with a step size of 1 message. This window captures subsets of the vector representation, segmenting it into smaller fragments that adhere to the specified dimensions.

Further enhancing this approach is the Overlapping Mechanism, which allows windows to overlap. As one window progresses, the subsequent one shares content with the previous, preserving information during transitions. The outcome is structured log sequences capturing detailed vector information while maintaining uniformity.

In summary, this methodology transforms continuous vector representations into an organized format, paving the way for subsequent analysis using advanced techniques such as convolutional neural networks (CNNs).

4.4 1D-CNN Classification Model

In this section, we present our proposed 1D-CNN architecture tailored to the task of anomaly detection. Figure 3 visually outlines the architecture's design, highlighting its key components and their interactions.

The architecture commences with three successive Convolutional Layers, each augmented with a Batch Normalization layer. These layers collaboratively extract intricate features from the input data. The initial convolution layer employs 64 filters, each with a kernel size of 3, operating on sequences of dimension (20, 768). This input structure aligns with the representation of our data, facilitating the extraction of essential temporal features.

Subsequently, the second convolution layer, consisting of 128 filters, and the third convolution layer, featuring 256 filters, continue to extract progressively complex features from the input sequences.

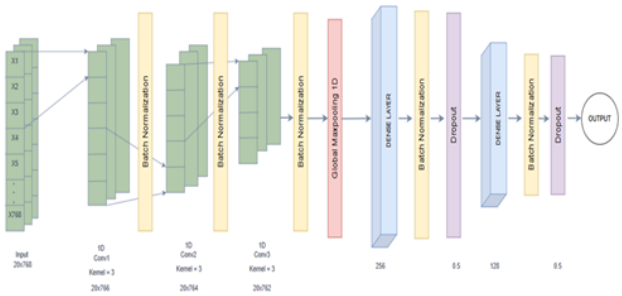

Figure 3: 1D-CNN Architecture

Each convolutional layer is accompanied by a Batch Normalization layer to stabilize and accelerate training convergence.

We employ a Global Max Pooling layer to encapsulate the most salient features from the segmented data. This layer selects the most relevant information across the temporal dimension, effectively summarizing the learned features.

The Fully Connected (Dense) layers that ensue serve to refine the extracted features for robust classification. With 256 units and a 'relu' activation function, the first dense layer strengthens the model's feature abstraction capability. A Dropout layer with a rate of 0.5 is strategically inserted to mitigate overfitting. The subsequent dense layer, equipped with 128 units and 'relu' activation, further enhances the feature discrimination process.

The final layer of the architecture is an Output Layer with a 'sigmoid' activation function. Given the binary classification nature of our task, this activation facilitates clear decision boundaries. The model's compilation entails using the Adam optimizer and binary cross-entropy loss, with accuracy as the evaluation metric.

Our 1D-CNN architecture dynamically learns to uncover intricate patterns within the structured log message representations, enabling accurate anomaly detection. Batch Normalization is incorporated to expedite training convergence and bolster overall performance. Dropout layers are strategically introduced to enhance model generalization by discouraging over-reliance on specific inputs. All the detail information is shown in Table 1.

Experimental validation of the proposed architecture is presented in the subsequent section, elucidating its effectiveness in anomaly detection through comprehensive evaluation.

5 EXPERIMENT

5.1 Datasets

This research involves a comprehensive assessment of our method using four publicly available datasets [18] - HDFS, Blue Gene/L, Thunderbird, and Spirit. These datasets contain distinct log message collections that enable a thorough evaluation of the approach. One widely used log dataset is HDFS [19], [20], which contains over 11 million log messages from a Hadoop Distributed File System deployed on Amazon EC2. Each session in this dataset is identified by a block ID and categorized as normal or abnormal. The BGL dataset [21], [22] is another log benchmark containing over 4.7

Table 1: Detail 1D-CNN Model

Layer	Maps	Size	Kernel	Activation Function
Input	1	20x768	-	-
Conv 1	64	20x766	3	ReLU
BN	64	20x766	-	-
Conv 2	128	20x764	3	ReLU
BN	128	20x764	-	-
Conv 3	256	20x762	3	ReLU
BN	256	20x762	-	-
Global Max Pooling		762	-	ReLU
Dense 1	256	256	-	ReLU
BN	256	256		
Dropout	256	256		
Dense 2	128	128		
BN	128	128		
Dropout	128	128		
Output	1	-	-	Sigmoid

Table 2: Training Ratio

Dataset	Train Ratio	
	Anomalies:Normal	
	Our ratio	Actual ratio
HDFS	1:33 (300:1000)	1:33 (13435:446613)
BGL	1:5 (10000:50000)	1:5 (345092:1725460)
Thunderbird	1:5 (10000:50000)	1:5 (28432:142160)
Spirit	1:5 (10000:50000)	1:5 (1631496:4755160)

Table 3: Testing Ratio

Dataset	Testing Ratio	
	Our ratio	Actual ratio
HDFS	1:33 (300:10000)	1:33 (3403:111610)
BGL	1:15 (2000:30000)	1:15 (60161:889412)
Thunderbird	1:55 (900:50000)	1:2189 (913:1999067)
Spirit	1:180 (200:36000)	1:180 (8804:1587845)

million log messages. This data originates from the Blue Gene/L supercomputer at Lawrence Livermore National Labs. Additionally, our evaluation includes the Thunderbird and Spirit datasets [21], sourced from Sandia National Labs' real-world supercomputers, where each log message has been manually classified as anomalous or non-anomalous, offering valuable reference information. For a comprehensive overview of the dataset specifics, please refer to Table 2 and Table 3 for detailed information.

In our study, we conducted an experiment under specific resource constraints, particularly relating to RAM and memory availability. Despite these limitations, we were able to perform a comprehensive evaluation using variations of the available datasets.

For instance, in the HDFS dataset, due to resource constraints, we focused on the HDFS training dataset. This dataset maintains a ratio of anomalies to normal instances at 1:5, specifically 1,000 anomalies to 5,000 normal instances. This ratio mirrors the distribution of the actual training dataset. Similarly, for the HDFS test dataset, we maintained the anomalies-to-normal ratio at 1:33, with 300 anomalies to 10,000 normal instances. This preservation of ratios ensures that our evaluation aligns closely with the real-world dataset distribution.

The BGL dataset was similarly evaluated with the constraints in mind. The ratio for the BGL training dataset was maintained at 1:5, with 10,000 anomalies to 50,000 normal instances. For the BGL test dataset, the anomalies-to-normal ratio was set at 1:15, featuring 2,000 anomalies to 30,000 normal instances.

Moving to the Thunderbird dataset, we retained a consistent approach. The ratio for the Thunderbird training dataset was preserved at 1:5, presenting 10,000 anomalies to 50,000 normal instances. The Thunderbird test dataset followed a ratio of 1:55, with 900 anomalies to 50,000 normal instances.

The Spirit dataset was also part of our comprehensive evaluation. The Spirit training dataset held a ratio of 1:5, having 10,000 anomalies to 50,000 normal instances. For the Spirit test dataset, the ratio was maintained at 1:180, featuring 200 anomalies to 36,000 normal instances.

For detailed insight into these ratios and the specific dataset characteristics, please refer to the table provided below. This tailored evaluation strategy enables us to effectively assess the performance and applicability of our approach across diverse datasets while accommodating resource constraints.

By maintaining consistent ratios and carefully selecting subsets of datasets, our evaluation method ensures that the results remain representative of actual log data distribution despite the imposed resource limitations.

5.2 Metrics

To evaluate the anomaly detection model, we utilize three key performance metrics: Precision, Recall, and F1-Score. These provide a comprehensive assessment of the model's ability to accurately identify anomalous logs.

Precision: Precision measures the percentage of log sequences the model predicts as anomalous that truly are anomalous. It indicates how precise the model is in classifying abnormal logs.

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

Recall: Recall, also called sensitivity, measures the percentage of actual anomalous logs the model correctly detects. It quantifies the model's ability to identify all the anomalous logs.

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

F1-Score: F1-Score balances Precision and Recall by taking their harmonic mean. This provides a holistic evaluation of the model across both metrics.

$F1 = \frac{2 \times precision \times recall}{precision+recall}$

5.3 Results

The experiments were conducted on Google Colab using Python3 and the TensorFlow library. Google Colab is equipped with 12.7 GB of RAM and 225 GB of memory. We are conducting a comparative analysis of our approach with the NeuralLog method, which detects

Table 4: Comparison Results

Dataset		LR	SVM	IM	LogRobust	Log2Vec	NeuralLog	1D-CNN
	P	0.99	0.99	1.00	0.98	0.94	0.96	0.98
HDFS	R	0.92	0.94	0.88	1.00	0.94	1.00	0.99
	F1	0.96	0.96	0.94	0.99	0.94	0.98	0.99
	P	0.13	0.97	0.13	0.62	0.80	0.98	0.96
BGL	R	0.93	0.30	0.30	0.96	0.98	0.98	0.99
	F1	0.23	0.46	0.18	0.75	0.88	0.98	0.97
	P	0.46	0.34	-	0.61	0.74	0.93	0.99
Thunder-	R	0.91	0.91	-	0.78	0.94	1.00	0.95
bird	F1	0.61	0.50	-	0.68	0.84	0.96	0.97
	P	0.89	0.88	-	0.97	0.91	0.98	0.99
Spirit	R	0.96	1.00	-	0.94	0.96	0.96	0.97
	F1	0.92	0.93	-	0.95	0.95	0.97	0.98

anomalies using a Transformer-based classification model and other log anomaly detection methods. All methods use the same training and testing dataset in Table II for fair comparison.

The comparison among the performance of our proposed method (1D-CNN model) and others across four public datasets reveals insightful observations, as shown in Table 4.

1D-CNN consistently outperforms other methods, including Logistic Regression (LR), Support Vector Machine (SVM), Invariant Mining (IM), LogRobust, and Log2Vec across multiple datasets. It exhibits superior performance in terms of precision, recall, and F1-score. This can be attributed to its ability to capture complex data patterns and dependencies, making it particularly effective for tasks where intricate relationships exist within the data. Notably, 1D-CNN achieves notably high F1-scores, indicating a strong balance between precision and recall, which is crucial for tasks where both false positives and false negatives need to be minimized.

When comparing 1D-CNN to NeuralLog, it becomes evident that both methods deliver strong performance, though with slight variations depending on the specific dataset. 1D-CNN maintains its competitive edge in terms of precision, showcasing its ability to provide highly accurate results. Moreover, 1D-CNN consistently achieves a balanced F1-score, indicating its proficiency in minimizing both false positives and false negatives. This makes it an excellent choice for tasks where precision and overall performance are crucial. On the other hand, NeuralLog excels primarily in recall, demonstrating its capability to effectively identify a higher proportion of true positives. While NeuralLog's recall performance is commendable, 1D-CNN offers a well-rounded approach with impressive precision, which is essential in scenarios where false alarms need to be minimized. Therefore, the selection between these two methods should be guided by the specific objectives and trade-offs that align with the application's requirements.

6 CONCLUSION

This study presented a new approach for network attack detection using 1D convolutional neural networks (CNNs). Extensive evaluations across multiple public datasets reveal 1D-CNNs consistently outperform other methods like logistic regression, SVM, and NeuralLog with Transformers. The 1D-CNN model demonstrates superior precision, recall, and F1-scores owing to its ability to capture intricate patterns and dependencies in log data.

Experiments on four benchmark datasets verify this 1D-CNN approach is highly effective and efficient for log anomaly detection. It consistently exceeds alternative machine learning techniques. Going forward, an intriguing enhancement could be incorporating self-supervised learning on the log sequence vector representations. This could potentially uncover latent data patterns to further improve anomaly detection performance.

In summary, this work highlights 1D-CNNs as a robust log anomaly detection method via thorough comparative assessment. The results cement 1D-CNNs' versatility for modeling complex sequential data. Integrating self-supervised learning represents a promising direction for even stronger anomaly detection capabilities.

REFERENCES

[1] M. Du, F. Li, G. Zheng, and V. 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. ACM, 2017, pp. 1285–1298.

[2] 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.. In IJCAI, Vol. 7. 4739–4745.

[3] Amir Farzad and T Aaron Gulliver. 2020. Unsupervised log message anomaly detection. ICT Express 6, 3 (2020), 229–237.

[4] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, *et al.* 2019. 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. 807–817.

[5] 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.. In IJCAI, Vol. 7. 4739–4745.

[6] Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. 2018. Detecting anomaly in big data system logs using convolutional neural network. In 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech). IEEE, 151–158.

[7] V.H. L. a. H. Zhang, "Log-based Anomaly Detection Without Log Parsing," arXiv: 2108.01955v3, 2021.O'

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," arXiv preprint arXiv:1706.03762,

2017.
[9] A. J. W. K. Meliboev Azizjon, "1D CNN based network intrusion detection with normalization on imbalanced data," 2020
[10] M. Schuster and K. Nakajima, "Japanese and korean voice search," in 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2012, pp. 5149–5152.
[11] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv preprint arXiv: 1609.08144, 2016.
[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv: 1810.04805, 2018.
[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, 2019.
[14] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pretraining text encoders as discriminators rather than generators," arXiv preprint arXiv:2003. 10555, 2020.
[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in International conference on machine learning. PMLR, 2014, pp. 1188–1196.
[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv: 1810.04805, 2018.
[17] (2021) Bert pretrained models. [Online]. Available: https://github.com/google-research/bert
[18] (2021) Loghub. [Online]. Available: https://github.com/logpai/loghub
[19] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv preprint arXiv: 1609.08144, 2016.
[20] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, 2009, pp. 117–132.
[21] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07). IEEE, 2007, pp. 575– 584.
[22] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: a large collection of system log datasets towards automated log analytics," arXiv preprint arXiv:2008.06448, 2020.