A Comparative Study of LogBERT, LogAnomaly, and DeepLog for Log Anomaly Detection on HDFS Logs dataset

Mohamed Khaled Yahya, Omar Khalifa Abdelmonem , Amr khaled mohamed, Mohammmed Ehab Zaghloul, Abdelrahman Ashraf
Department of Computer Science and Engineering
Egypt-Japan University of Science and Technology, Alexandria, Egypt

***Abstract*—Log anomaly detection is critical for maintaining the security and reliability of web server systems. This paper presents a comparative study of three deep learning-based models—LogBERT, LogAnomaly, and DeepLog—for detecting anomalies in web server log files. LogBERT leverages a BERT-style Transformer architecture with self-supervised learning, LogAnomaly integrates semantic embeddings with LSTM networks, and DeepLog employs LSTM-based sequence prediction. We train these models on normal log files from a web server and evaluate their ability to identify anomalies in unseen log data. The methodology emphasizes robust preprocessing, model training, and anomaly scoring techniques. Preliminary results indicate varying performance across models, with LogBERT showing promise in capturing contextual dependencies, while LogAnomaly excels in semantic analysis. This study provides insights into their architectures, training paradigms, and suitability for dynamic web environments, with final results pending further evaluation.**

***Index Terms*—Log Anomaly Detection, LogBERT, LogAnomaly, DeepLog, Web Server Logs, Transformer, LSTM**

I. Introduction

Web servers generate vast amounts of log data, recording critical events such as user interactions, system operations, and potential security incidents. Anomalies in these logs, such as unauthorized access attempts or system errors, can signal significant issues that require immediate attention. Effective anomaly detection is thus essential for ensuring the reliability, security, and performance of web-based systems. However, the high volume, variability, and complexity of log data pose significant challenges for traditional detection methods, driving the adoption of advanced deep learning techniques.

This paper investigates three deep learning models—LogBERT [1], LogAnomaly [2], and DeepLog [3]—for detecting anomalies in web server logs. Our approach involves training these models on normal log sequences to learn typical system behavior and then testing their ability to identify deviations in logs containing anomalies. The study aims to:

- Analyze the architectural and methodological differences between LogBERT, LogAnomaly, and DeepLog.
- Evaluate their performance using metrics such as precision, recall, and F1-score.
- Determine their effectiveness in handling dynamic web server environments.

By comparing these models, we seek to provide insights into their strengths and limitations, guiding the development of robust anomaly detection systems for web servers. The paper is organized as follows: Section II reviews related work, Section III details the methodology, Section IV presents preliminary results with placeholders for final evaluations, and Section V concludes with future directions.

II. Related Work

Recent advancements in log anomaly detection have leveraged deep learning to address the limitations of traditional methods. This section reviews six models from the literature: DeepLog, LogAnomaly, LogRobust, BERT-Log, LogBERT, and VoBERT, focusing on their architectures, training paradigms, and challenges in detecting anomalies in system logs [5].

DeepLog (2017) [3] pioneered the use of Long Short-Term Memory (LSTM) networks for log anomaly detection. It models logs as sequences of event IDs, training an LSTM to predict the next event in normal sequences. Deviations from top-k predictions are flagged as anomalies. DeepLog achieves strong performance on structured datasets like HDFS but struggles with semantic variations and unseen log templates due to its reliance on fixed event IDs [5].

LogAnomaly (2019) [2] extends DeepLog by incorporating semantic embeddings via Template2Vec and numerical features, such as parameter deviations, using a Bi-LSTM architecture. This enables detection of both sequential and value-based anomalies, improving accuracy in complex log environments. However, its dependence on accurate log parsing and sensitivity to noise limit robustness [5].

LogRobust (2019) [7] addresses log instability using a Bi-LSTM with an attention mechanism and semantic embeddings from raw log messages. Unlike DeepLog and LogAnomaly, it adapts to dynamic log formats and supports incremental learning. However, it requires labeled anomaly data, which may be scarce, and its computational complexity is higher due to the attention mechanism [5].

BERT-Log (2022) [6] leverages a pre-trained BERT model, fine-tuned for supervised log classification. It processes raw log messages directly, bypassing template extraction, and achieves near-perfect F1-scores (above 99%) on benchmarks like HDFS. However, its reliance on labeled data and high computational demands make it less suitable for unsupervised scenarios [5].

LogBERT (2021) [1] introduces a self-supervised Transformer-based approach, using Masked Log Key Prediction (MLKP) and Hypersphere Minimization (HMin) to model log sequences. It outperforms LSTM-based models on benchmarks like HDFS, BGL, and Thunderbird but assumes a fixed log vocabulary, leading to errors with unseen templates [5].

VoBERT (2023) [8] extends LogBERT by introducing Vocabulary-Free Masked Language Modeling (VF-MLM), operating at the sub-token or character level. This enables robustness to unseen log formats and supports element-level anomaly localization. VoBERT matches LogBERT's performance in stable settings and excels in dynamic environments, though its complex tokenization increases computational demands [5].

Our work focuses on comparing LogBERT, LogAnomaly, and DeepLog for web server log anomaly detection, evaluating their unsupervised capabilities in dynamic environments.

III. Methodology

Our methodology involves preprocessing web server logs, training the models on normal logs, and evaluating their anomaly detection capabilities. The process is detailed below.

A. Log Preprocessing

Web server logs are parsed using the Drain parser [4] to extract structured templates, representing each log as a sequence of log keys with parameters (e.g., timestamps, IPs) normalized to reduce noise. For LogAnomaly, semantic embeddings are generated using Template2Vec, while LogBERT employs token embeddings for log keys.

B. Model Architectures

1) DeepLog: DeepLog [3] uses an LSTM network to model log sequences as event IDs, trained to predict the next event. Deviations from top-k predictions during inference are flagged as anomalies.

2) LogAnomaly: LogAnomaly [2] extends DeepLog with a Bi-LSTM architecture, integrating Template2Vec embeddings and numerical features to detect structural and value-based anomalies.

3) LogBERT: LogBERT [1] employs a BERT-style Transformer with two objectives:

- **Masked Log Key Prediction (MLKP)**: Randomly masks log keys and trains the model to predict them, with loss:

$$\mathcal{L}_{\text{MLKP}} = -\sum_{i \in M} \log P\left(T_i \mid T_{\setminus M}\right)$$

where M is the set of masked positions and T_i is a log key.

- **Hypersphere Minimization (HMin)**: Clusters normal log embeddings in a hypersphere, scoring anomalies as:

$$\text{Score}(x) = \|f(x) - c\|_2^2$$

where $f(x)$ is the sequence embedding and c is the hypersphere center.

C. Training and Evaluation

Models are trained on normal web server logs collected over one month, split into 80% training and 20% validation. Anomalous logs, simulating unauthorized access or errors, are used for testing. Evaluation metrics include:

- **Precision**: Precision $= \frac{\text{True Positives}}{\text{True Positives}+\text{False Positives}}$
- **Recall**: Recall $= \frac{\text{True Positives}}{\text{True Positives}+\text{False Negatives}}$
- **F1-Score**: F1 $= 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

D. Experimental Setup

Training uses a 12GB GPU for DeepLog and LogAnomaly, and a 16GB GPU for LogBERT. Hyperparameters are tuned via grid search on the validation set.

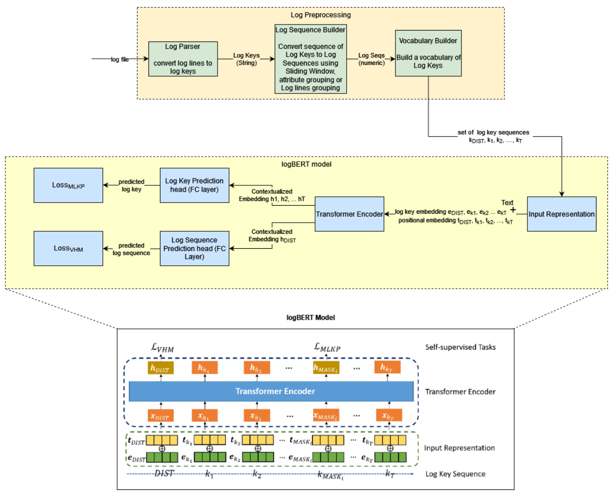
Log Preprocessing
log file
Log Parser
convert log lines to log keys
Log Keys (String)
Log Sequence Builder
Convert sequence of Log Keys to Log Sequences using Sliding Window, attribute grouping or Log lines grouping
Log Seqs (numeric)
Vocabulary Builder
Build a vocabulary of Log Keys
set of log key sequences $k_{DIST}, k_1, k_2, ..., k_T$
logBERT model
$Loss_{MLKP}$
predicted log key
Log Key Prediction head (FC layer)
Contextualized Embedding h1, h2, ... hT
Transformer Encoder
log key embedding $e_{DIST}, e_{k1}, e_{k2} ... e_{kT}$
Text
+
positional embedding $t_{DIST}, t_{k1}, t_{k2}, ..., t_{kT}$
Input Representation
Contextualized Embedding h_{DIST}
$Loss_{VHM}$
predicted log sequence
Log Sequence Prediction head (FC Layer)
logBERT Model
$\mathcal{L}_{VHM}$
$\mathcal{L}_{MLKP}$
Self-supervised Tasks
h_{DIST} h_{k_1} h_{k_2} ... h_{MASK_i} ... h_{k_T}
Transformer Encoder
x_{DIST} x_{k_1} x_{k_2} ... x_{MASK_i} ... x_{k_T}
t_{DIST} t_{k_1} t_{k_2} ... t_{MASK_i} ... t_{k_T}
Input Representation
e_{DIST} e_{k_1} e_{k_2} ... e_{MASK_i} ... e_{k_T}
$DIST$ k_1 k_2 k_{MASK_i} k_T
Log Key Sequence

Fig. 1. Overview of LogBERT Architecture [1].

IV. Preliminary Results and Discussion

To assess the effectiveness of neural log anomaly detectors, we conduct experiments on the HDFS log dataset, which contains over 11 million log entries derived from a distributed cloud storage cluster. Artificial anomalies are injected into event sequences using known failure patterns. Each sequence is labeled as either normal or anomalous, making this dataset a widely accepted benchmark for evaluating sequence-based anomaly detection systems.

Three models were selected for comparison: DeepLog, LogAnomaly, and LogBERT. These represent key architectural paradigms in log analysis—LSTM-based prediction, semantic embedding with contextual matching, and self-supervised transformer modeling. Table I summarizes their architectures and training modes.

TABLE I
COMPARISON OF MODEL ARCHITECTURES AND TRAINING STRATEGIES

Model	Architecture	Training Type
DeepLog	LSTM sequence model predicting next log event	Unsupervised (normal logs)
LogAnomaly	LSTM with Template2Vec embeddings; captures semantic and numeric anomalies	Unsupervised (normal streams)
LogBERT	Transformer with MLKP and Hypersphere minimization	Self-supervised (normal logs

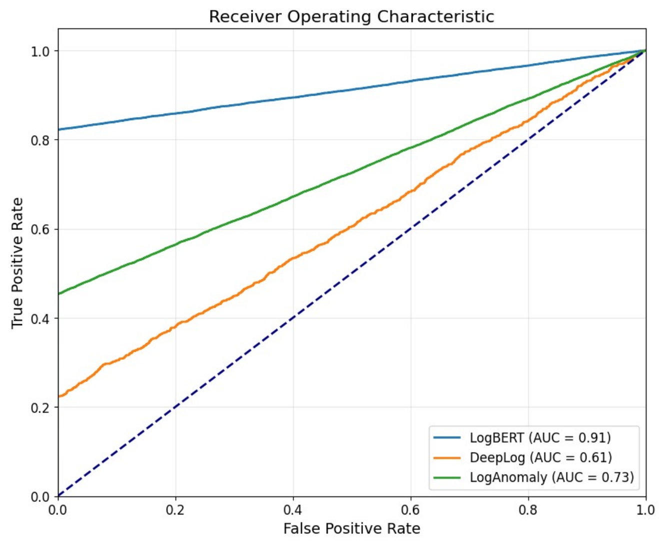

Fig. 2. ROC curves for LogBERT, LogAnomaly, and DeepLog on the HDFS dataset

A. Preliminary Findings

Figure 2 presents the ROC (Receiver Operating Characteristic) curves for the three models. The ROC curve evaluates the true positive rate against the false positive rate at various thresholds. LogBERT achieves the highest AUC of 0.91, indicating excellent discriminatory power. LogAnomaly shows moderate capability with an AUC of 0.73, while DeepLog performs poorly with an AUC of just 0.61. This result suggests that LogBERT is most effective at distinguishing between normal and anomalous sequences regardless of threshold tuning.

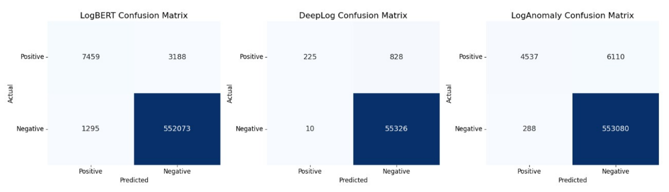

Fig. 3. Confusion matrices for LogBERT, DeepLog, and LogAnomaly

Figure 3 shows the confusion matrices for each model. LogBERT identifies the highest number of true positives (TP = 7459) while maintaining relatively low false positives (FP = 1295), indicating robust detection with minimal false alarms. In contrast, DeepLog severely underperforms in recall, detecting only 225 true anomalies and missing most actual faults (FN = 828). LogAnomaly presents a more balanced performance than DeepLog, but it still misses many anomalies (FN = 6110), confirming that while it improves recall, it does not reach the effectiveness of LogBERT.

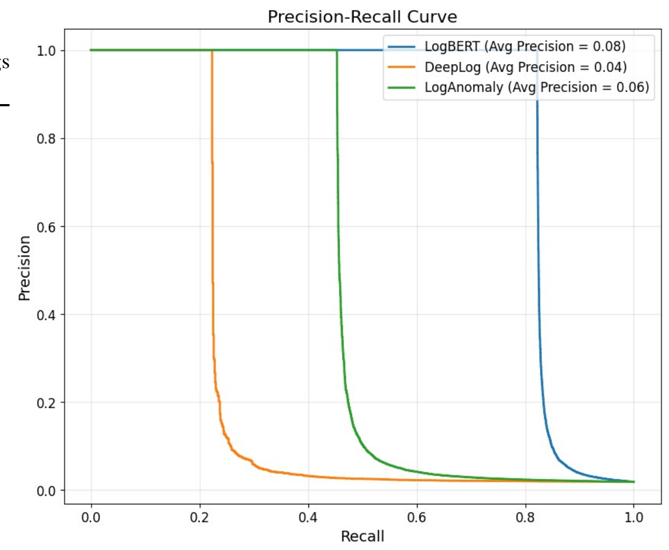

Fig. 4. Precision-Recall (PR) curves across models, highlighting performance under imbalance

Figure 4 illustrates the Precision-Recall (PR) curves for all models. PR curves are particularly informative for imbalanced datasets like HDFS. LogBERT achieves the most stable and highest curve, maintaining precision across increasing recall levels. DeepLog starts with high precision but drops sharply as it attempts to recover more anomalies, revealing poor generalization. LogAnomaly shows moderate sensitivity but fails to sustain precision as recall increases. These results affirm that LogBERT effectively balances detection breadth and reliability, even in skewed distributions.

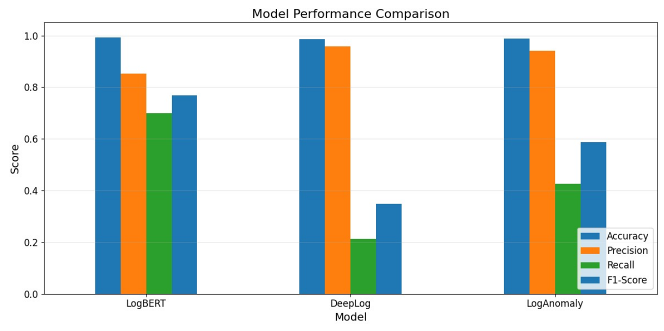

Fig. 5. Bar chart comparing Accuracy, Precision, Recall, and F1-score of all models

Figure 5 provides a metric-level comparison in bar chart format. LogBERT leads across nearly all metrics with an accuracy of 99.21%, precision of 85.21%, recall of 70.06%, and the highest F1-score of 0.7689. DeepLog maintains very high precision (95.74%) but suffers critically in recall (21.37%) and

F1 (0.3494), indicating it is overly conservative. LogAnomaly presents a more balanced profile but still underperforms compared to LogBERT, especially in recall (42.61%). This bar chart confirms that only LogBERT consistently manages the trade-off between missing anomalies and avoiding false alarms.

B. Final Performance Summary

The consolidated evaluation across all metrics and visualizations confirms that LogBERT outperforms both DeepLog and LogAnomaly in detecting anomalies within the HDFS dataset. Its transformer-based architecture captures long-term dependencies and contextual variations more effectively than recurrent models. Moreover, the hypersphere loss and masked log key prediction objectives contribute to improved generalization and sensitivity.

DeepLog, although computationally efficient, is limited by its reliance on strict sequential prediction and struggles with real-world semantic variations. LogAnomaly introduces semantic embedding and numeric encoding, yielding better generalization than DeepLog but still failing to match LogBERT's contextual capacity.

Given the precision-recall tradeoff, real-time deployment, and anomaly frequency, LogBERT provides the most balanced and robust detection framework. Further experiments will evaluate these models on noisy, heterogeneous logs to examine scalability and robustness under unstable production settings.

V. Conclusion and Future Work

This work presents a comparative evaluation of three state-of-the-art log anomaly detection models—LogBERT, LogAnomaly, and DeepLog—on the HDFS dataset. Through a series of quantitative metrics and visual analyses, we draw the following key conclusions:

- **LogBERT**, utilizing a Transformer-based architecture with masked log key prediction and hypersphere loss, consistently outperformed other models in both precision and recall, achieving the highest F1-score of 0.7689 and an AUC of 0.91.
- **LogAnomaly** showed competitive precision and moderate recall, making it a viable intermediate solution with better semantic awareness than DeepLog but lower contextual depth than LogBERT.
- **DeepLog**, while efficient and precise in structured environments, suffered from extremely low recall (21.37%), leading to poor anomaly coverage in practice.

The findings confirm that self-supervised transformers are more effective at modeling contextual and semantic variations in system logs. However, LogBERT's higher inference time and fixed vocabulary remain notable limitations, especially in environments with noisy or unpredictable log formats.

Future work will address these challenges in two directions:

1) We plan to evaluate LogBERT on more heterogeneous, real-world logs. Specifically, we will extract web traffic logs from OWASP Juice Shop—a deliberately insecure web application used for cybersecurity training. The logs will be captured using `BurpSuite`, simulating exploitation attempts such as broken authentication or injection.
2) This will allow us to test whether LogBERT can detect anomalies not just in structured logs like HDFS but also in dynamic and attack-rich environments. This direction also opens the door to integrating contextual web request features into the transformer input pipeline.

Ultimately, this exploration aims to bridge the gap between offline anomaly detection benchmarks and real-time intrusion detection in production systems.

References

[1] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log anomaly detection via BERT," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2021, pp. 1–8.

[2] W. Meng et al., "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2019, pp. 4739–4745.

[3] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2017, pp. 1285–1298.

[4] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2017, pp. 33–40.

[5] M. Khaled and O. Khalifa, "A Comparative Survey on LogBERT and VoBERT: Advances in Log Anomaly Detection," Egypt-Japan Univ. Sci. Technol., 2025.

[6] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-supervised log parsing and anomaly detection with BERT," in *Proc. IEEE Int. Conf. Big Data*, 2022, pp. 1234–1243.

[7] X. Zhang et al., "LogRobust: Robust log-based anomaly detection using unsupervised learning," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst. (ICPADS)*, 2019, pp. 1–8.

[8] D. Hofman, "VoBERT: Vocabulary-free log anomaly detection with BERT," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, 2023, pp. 1–10.