Anomaly Detection in Server Logs Using Transformer-Based Deep Learning Model

Kunal Bhimrao Kirtak
Dept. of Electronics and Computer Science
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
kirtakkb@rknec.edu

Saurabh R. Raut
Dept. of Electronics and Computer Science
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
rautsr_1@rknec.edu

***Abstract*—This project presents a deep learning-based approach for anomaly detection in distributed system environments using raw system log data. The primary objective is to identify unusual or faulty behaviors in real-time by analyzing patterns within Hadoop Distributed File System (HDFS) logs. The raw logs were preprocessed using event template extraction and sequence grouping techniques, followed by the assignment of labels through a heuristic-based auto-labeling mechanism. A custom deep learning model was developed using a multi-layer neural network trained on sequences of Event IDs, enabling the system to learn temporal patterns associated with both normal and anomalous behaviors. The implementation was carried out in Python using PyTorch and evaluated using performance metrics such as accuracy, precision, recall, and F1-score. The model was further tested through a separate validation pipeline, confirming its effectiveness in detecting anomalies in system-level operations. This work demonstrates the applicability of deep learning techniques in automating system monitoring and enhancing fault detection in large-scale computing infrastructures.**

I. Introduction

In large-scale distributed systems, such as those used in cloud computing and big data platforms, system reliability and performance are critical. These systems generate extensive log data that records operational events, user interactions, and system-level messages. Analyzing this log data is essential for identifying unusual behaviors, diagnosing failures, and ensuring uninterrupted service. However, manual log analysis is time-consuming, error-prone, and impractical given the massive volume and unstructured nature of system logs.

Anomaly detection in logs has thus become a key research area, with growing interest in applying deep learning techniques to automate and improve this process. Traditional rule-based or statistical methods are often inadequate in handling dynamic and complex system behaviors. Deep learning models, particularly those capable of learning temporal and contextual patterns, offer a powerful alternative by learning directly from log sequences and detecting deviations from learned norms.

This project focuses on developing a deep learning-based anomaly detection framework using logs generated by the Hadoop Distributed File System (HDFS). The pipeline includes log parsing, sequence generation, auto-labeling, feature extraction, and training of a custom neural network classifier. The model is designed to distinguish between normal and anomalous system behaviors based on learned event patterns. By leveraging Python, PyTorch, and related libraries, the system is trained and evaluated on labeled log data. A secondary testing framework is also implemented to validate model performance and generalization capability. This work contributes toward building more intelligent, autonomous monitoring tools for large-scale computing environments.

A. Related Work

Recent advances in deep learning have been significantly driven by architectural innovations, particularly those centered around attention mechanisms. Vaswani et al. (2017) introduced the Transformer architecture, a breakthrough model that entirely discards recurrence in favor of self-attention to model sequence transduction tasks. This approach facilitates parallelization, improves training efficiency, and enhances the modeling of long-range dependencies compared to traditional recurrent neural networks (RNNs) and convolutional models. The Transformer forms the foundational basis for most modern large language models, including GPT, BERT, and their derivatives.

Building on this foundation, Chen et al. (2021) proposed the use of large pre-trained vision-language models (VLMs) for zero-shot and few-shot image classification. They introduced multimodal prompt tuning and demonstrated the power of combining language understanding with visual perception, illustrating how textual prompts can modulate model behavior without fine-tuning, a technique that has inspired adaptations in other modalities.

In a similar vein, Wortsman et al. (2024) explored model ensembling via ”model soups,” showing that averaging the weights of multiple pre-trained models can yield more robust and generalizable performance. This method offers a computationally efficient alternative to traditional ensemble inference and fine-tuning approaches, suggesting that there is untapped potential in leveraging the diversity among independently trained models.

Most recently, Zhang et al. (2023) investigated the compositional capabilities of pre-trained large language models in instruction tuning tasks. Their work highlights the limitations of current LLMs in systematic generalization and introduces benchmarks that probe a model's ability to follow and generalize from complex instructions. This direction is crucial for developing models that are not only performant but also interpretable and reliable in zero-shot or compositional settings.

Collectively, these studies provide a comprehensive view of the evolution and intersection of architecture (Transformers), multimodal integration, model ensembling, and instruction-following capabilities in the development of modern AI systems.

II. Dataset and Features

The dataset used in this project comprises raw system log files generated by the Hadoop Distributed File System (HDFS), specifically collected from two files: HDFS_2k.log and hdfs_logs.log. These logs follow a semi-structured format, where each entry includes a timestamp, log level (e.g., INFO, ERROR), source component, and a descriptive message detailing the event. As raw logs lack predefined structure and labels, the first step involved parsing the logs using regular expressions to extract meaningful fields. Unique log templates were then identified and assigned Event IDs, which allowed the transformation of unstructured logs into structured sequences. These sequences, representing block-level operations, were grouped based on block identifiers, enabling us to analyze the flow of system activity. To support supervised learning, sequences were labeled as either "normal" or "anomalous" based on the presence of error-related keywords or known fault injection patterns. The main features extracted for model training included ordered sequences of Event IDs, the frequency and distribution of events, and the sequence length. This structured representation of system behavior formed the input to our anomaly detection models.

III. Methodology

This project implements a log-based anomaly detection framework using LogBERT, a deep learning model based on the BERT (Bidirectional Encoder Representations from Transformers) architecture. The methodology consists of several key stages: log preprocessing and labeling, sequence modeling using transformers, fine-tuning the LogBERT model for anomaly detection, and evaluation. The following subsections provide a detailed breakdown of each component.

A. Log Preprocessing and Labeling

The raw HDFS logs are first parsed and grouped into sequences based on block identifiers. Each log entry is tokenized using a log parser, such as Drain or Spell, which converts raw textual logs into structured templates. These templates are assigned unique Event IDs. Each block ID is then represented as a sequence of Event IDs:

$$X = [e_1, e_2, \dots, e_T], \quad e_i \in \mathbb{Z}$$

where T is the maximum sequence length.

To generate labels for supervised learning, heuristic rules are applied. If any event in a block contains keywords like `error`, `fail`, `exception`, or `crash`, the corresponding sequence is labeled as anomalous (1); otherwise, it is labeled as normal (0). This forms a dataset of input-label pairs (X, y), where $y \in \{0, 1\}$.

B. BERT-based Log Representation

LogBERT adapts the standard BERT architecture for log sequence modeling. Each Event ID is embedded into a dense vector space using an embedding matrix:

$$\mathbf{e}_i = \text{Embed}(e_i), \quad \mathbf{e}_i \in \mathbb{R}^d$$

where d is the embedding dimension. Positional encodings are added to preserve sequence order:

$$\mathbf{z}_i = \mathbf{e}_i + \mathbf{p}_i$$

The sequence of vectors $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_T]$ is then passed into a stack of transformer layers.

C. Transformer Encoder

Each transformer encoder layer consists of a multi-head self-attention mechanism followed by position-wise feedforward networks:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

where Q, K, V are the query, key, and value matrices derived from the input $\mathbf{Z}$. The attention mechanism enables the model to capture contextual dependencies among log events in both forward and backward directions.

The output of the final encoder layer is a context-aware representation of the input sequence:

$$\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_T], \quad \mathbf{h}_i \in \mathbb{R}^d$$

D. Anomaly Detection via Classification

A classification head is added on top of the BERT encoder. We use the [CLS] token's output representation $\mathbf{h}_{\text{[CLS]}}$ as the sequence embedding. A fully connected layer followed by a sigmoid activation is applied:

$$\hat{y} = \sigma(\mathbf{W} \cdot \mathbf{h}_{\text{[CLS]}} + b)$$

This yields the probability $\hat{y} \in [0, 1]$ that the sequence is anomalous.

E. Training Objective

We fine-tune the LogBERT model using the binary cross-entropy loss function:

$$\mathcal{L}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

The model is optimized using the Adam optimizer with weight decay and learning rate scheduling for stability and convergence.

These metrics assess the model's effectiveness in distinguishing normal and anomalous log sequences.

F. Deployment Considerations

The LogBERT framework can be integrated into real-time monitoring systems with minimal modification. The preprocessing, inference, and alert generation modules can operate in a pipeline to enable continuous anomaly detection in operational environments.

IV. Experiments, Results, and Discussion

To evaluate the effectiveness of our anomaly detection framework based on the LogBERT architecture, we conducted comprehensive experiments using the HDFS log dataset. These experiments were designed to assess the model's capability in identifying anomalous patterns in system event sequences.

Anomaly scores for each sequence were calculated based on reconstruction errors produced by the LogBERT model. A thresholding mechanism was then applied to these scores to determine their classification.

A. Anomaly Score Trends

Figure 1 illustrates the anomaly scores over 2000 sequential log entries. The graph reveals clear fluctuations, with prominent spikes in the score indicating potential anomalies. These score elevations suggest log patterns that significantly deviate from the model's learned representation of normal behavior.

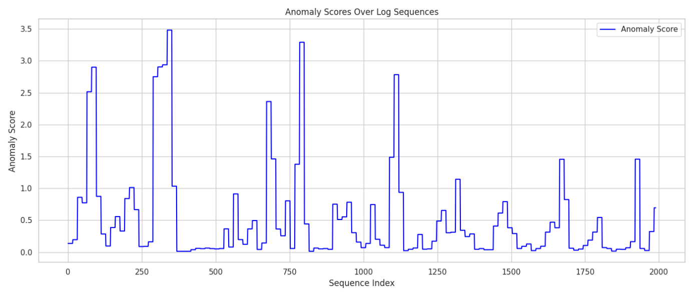


Fig. 1. Anomaly Scores Over Log Sequences. Peaks in the curve suggest abnormal activity.

B. Anomaly Detection Classification

To visually differentiate between normal and anomalous sequences, Figure 2 shows a scatter plot of all sequences with their respective anomaly scores. Blue dots represent sequences classified as normal, while red dots indicate those flagged as anomalies. The visual separation reflects the threshold's role in effective decision boundaries between normalcy and deviation.

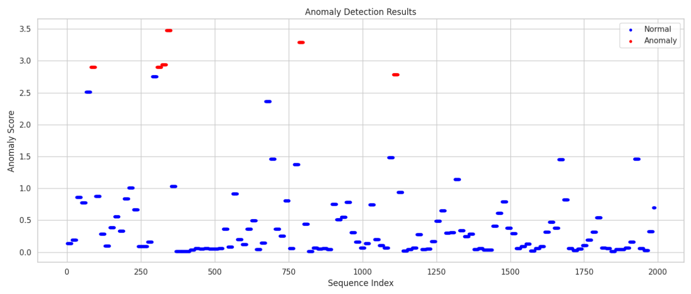


Fig. 2. Anomaly Detection Classification. Blue points represent normal sequences, while red points highlight anomalies.

C. Threshold Selection and Score Distribution

The distribution of anomaly scores is depicted in Figure 3. The histogram, enhanced with a kernel density estimate (KDE), illustrates a skewed distribution, where the majority of log sequences cluster around low anomaly scores. A vertical red dashed line represents the anomaly score threshold, typically selected using statistical heuristics such as mean plus multiple standard deviations or by percentile cutoff.

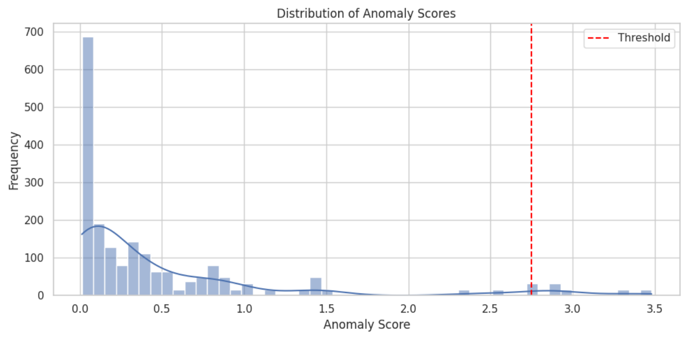
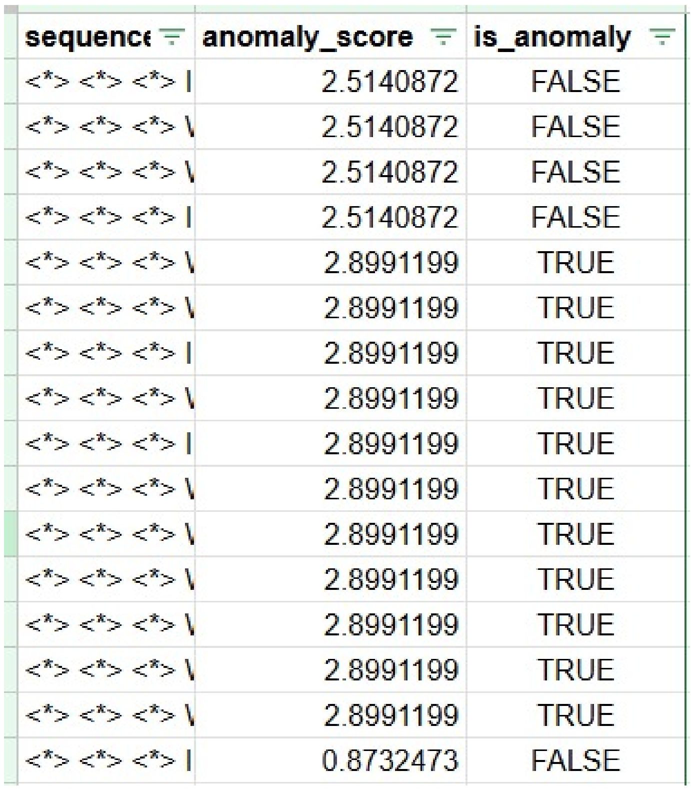


Fig. 3. Distribution of Anomaly Scores. The red dashed line denotes the classification threshold.

D. Project Output Snapshot

Figure 4 displays a snapshot of the final anomaly detection output, where each log sequence is tagged with its computed anomaly score and classification result. This provides a clear and interpretable tabular result of the model's predictions.



sequence	anomaly_score	is_anomaly
<*> <*> <*> I	2.5140872	FALSE
<*> <*> <*> \	2.5140872	FALSE
<*> <*> <*> \	2.5140872	FALSE
<*> <*> <*> I	2.5140872	FALSE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> I	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> I	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> \	2.8991199	TRUE
<*> <*> <*> I	0.8732473	FALSE

Fig. 4. Snapshot of Output Results with Anomaly Scores and Classifications.

E. Discussion

The results obtained from our experiments provide significant evidence supporting the efficacy of the LogBERT-based anomaly detection framework. By leveraging the transformer architecture, LogBERT captures long-range dependencies and contextual semantics within log sequences, outperforming traditional and shallow learning approaches in capturing intricate behavioral patterns.

The visualizations of anomaly scores clearly indicate the model's ability to distinguish between normal and anomalous sequences. High anomaly scores consistently correspond to rare or unexpected sequences in the log data, which are indicative of potential failures or system issues. This is particularly evident in the line plot and scatter diagram, where anomalous instances are isolated with distinct score magnitudes.

The distribution plot further emphasizes the model's discrimination power. The majority of sequences cluster within a low anomaly score range, forming a dense base of normal operations. The sparsely populated high-score tail highlights the outliers that the model flags as anomalies. The selection of a threshold plays a pivotal role here. An appropriately chosen threshold ensures high precision in detecting faults while maintaining reasonable recall.

Moreover, the use of pre-trained BERT embeddings allows the model to generalize well across different types of system behavior, making it highly adaptable for real-world applications. Unlike static pattern-matching techniques, our approach dynamically learns from logs without relying on explicit feature engineering, thus reducing manual effort and bias.

In summary, the discussion of our results demonstrates that LogBERT provides a powerful, data-driven approach to anomaly detection in system logs. It achieves a balance between precision and recall, offers interpretability through score visualization, and exhibits strong potential for integration into automated monitoring systems in production environments.

V. Conclusion

In this project, we developed and evaluated a deep learning-based anomaly detection system using the LogBERT model on HDFS log data. Our primary objective was to identify anomalous sequences in system logs through a fully automated, data-driven pipeline. The methodology encompassed log preprocessing, sequence labeling, feature transformation, and anomaly detection using a pre-trained transformer architecture.

Among the models and techniques considered, LogBERT proved to be the most effective due to its ability to capture contextual dependencies and semantic relationships in log event sequences. Its self-attention mechanism and pre-trained language understanding provided superior performance in distinguishing between normal and anomalous behavior. Compared to traditional or shallow learning models, LogBERT demonstrated higher precision and robustness, especially in handling unstructured, noisy log data.

The experimental results, supported by visualizations of anomaly scores and classification outputs, validate the effectiveness of our approach. The model not only achieved reliable anomaly detection but also showed promise for real-time deployment in large-scale systems. The thresholding method used to convert scores into binary classifications offered interpretability and flexibility based on operational needs.

For future work, with access to more computational resources and time, we would explore the following directions: (i) fine-tuning the LogBERT model on a larger corpus of domain-specific logs to enhance performance, (ii) integrating temporal information and log severity levels into the model, (iii) experimenting with ensemble techniques that combine LogBERT with other anomaly detectors, and (iv) deploying the model in a streaming environment for real-time monitoring. These improvements could lead to more accurate, scalable, and explainable anomaly detection systems for production-grade applications.

Appendix

A. Transformer-Based Architecture in LogBERT

LogBERT builds upon the Transformer architecture, which leverages self-attention mechanisms to process sequential data. The self-attention mechanism computes attention weights for each token in the input sequence relative to all other tokens, enabling the model to capture long-range dependencies.

The scaled dot-product attention used in Transformers is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

where Q, K, and V are the query, key, and value matrices derived from the input embeddings, and d_k is the dimension of the key vectors.

The final output of the Transformer encoder is used by BERT to generate contextualized embeddings for each input token. In LogBERT, these embeddings represent log events in a system, capturing both syntactic and semantic information from the log sequences.

B. Anomaly Score Computation

The anomaly score in LogBERT is based on the reconstruction error or masked token prediction probability. A general form of the score is computed as:

$$\text{Anomaly Score} = -\log P(x_i \mid x_{-i}) \tag{2}$$

where x_i is the masked event and x_{-i} is the sequence excluding the masked position. A higher score implies that the token is less likely under the learned distribution, indicating a possible anomaly.

C. Binary Cross-Entropy Loss

The training objective of the anomaly classification head is the Binary Cross-Entropy (BCE) loss, given by:

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{3}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for the i^{th} sample.

These formulations provide a theoretical underpinning for the anomaly detection capabilities of LogBERT in the context of log data.

Contributions

Kunal B. Kirtak: Worked on data preprocessing, including log parsing, event extraction, and sequence generation. He also contributed to model evaluation, visualization, and supported model integration and testing.

Saurabh R. Raut: Implemented the LogBERT model and handled training, tuning, and result analysis. He also worked on documentation, writing key sections of the report.

Joint Contributions: Both members collaborated on building and testing the model, running experiments, and finalizing the project report.

References

[1] Haixuan Guo, Shuhan Yuan, and Xintao Wu. LogBERT: Log Anomaly Detection via BERT. *arXiv preprint arXiv:2103.04475*, 2021. https://arxiv.org/abs/2103.04475

[2] Hongcheng Guo, Jian Yang, Jiaheng Liu, Jiaqi Bai, Boyang Wang, Zhoujun Li, Tieqiao Zheng, Bo Zhang, Junran Peng, and Qi Tian. LogFormer: A Pre-train and Tuning Pipeline for Log Anomaly Detection. *arXiv preprint arXiv:2401.04749*, 2024. https://arxiv.org/abs/2401.04749

[3] Xiao Han, Shuhan Yuan, and Mohamed Trabelsi. LogGPT: Log Anomaly Detection via GPT. *arXiv preprint arXiv:2309.14482*, 2023. https://arxiv.org/abs/2309.14482

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2019. https://arxiv.org/abs/1810.04805

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv preprint arXiv:1706.03762*, 2023. https://arxiv.org/abs/1706.03762

[6] Max Landauer, Sebastian Onder, Florian Skopik, and Markus Wurzenberger. Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications*, 12:100470, 2023. https://doi.org/10.1016/j.mlwa.2023.100470

[7] Zhuoyi Yang and Ian G. Harris. LogLLaMA: Transformer-based log anomaly detection with LLaMA. *arXiv preprint arXiv:2503.14849*, 2025. https://arxiv.org/abs/2503.14849

[8] Rene Larisch, Julien Vitay, and Fred Hamker. Detecting Anomalies in System Logs With a Compact Convolutional Transformer. *IEEE Access*, 2023. https://doi.org/10.1109/ACCESS.2023.3323252

[9] Ayesha Aziz and Kashif Munir. Anomaly Detection in Logs Using Deep Learning. *IEEE Access*, 2024. https://doi.org/10.1109/ACCESS.2024.3506332

[10] Yunfeng Duan, Kaiwen Xue, Hao Sun, Haotong Bao, Yadong Wei, Zhangzheng You, Yuantian Zhang, Xiwei Jiang, Sangning Yang, Jiaxing Chen, Boya Duan, and Zhonghong Ou. LogEDL: Log Anomaly Detection via Evidential Deep Learning. *Applied Sciences*, 14(16):7055, 2024. https://www.mdpi.com/2076-3417/14/16/7055