Journal of Artificial Intelligence in Information Technology (JAI-IT)
Vol. 1 No. 1 June 2026
E-ISSN: XXXX-XXXX; P-ISSN: XXXX-XXXX, Pages 85-103
DOI: https://doi.org/10.51903/dj3hk092

Convolutional Neural Network Approach for Anomaly Pattern Analysis in Information System Log Data

Muharrom Fatihaturrizqi[1], Eko Siswanto[2]
Email: muharom@stekom.ac.id, eko.siswanto@stekom.ac.id
Orcid: https://orcid.org/0009-0002-4630-175X (1), https://orcid.org/0009-0002-1972-663X (2)
[1]*Dept.Of Design, Faculty of Academic Study, Universitas Sains dan Teknologi Komputer, Semarang, Central Java, 50192*
[2] *Dept. Of Computer Technology, Faculty of Vocation Study, Universitas Sains dan Teknologi Komputer, Semarang, Central Java, 50192*
*Corresponding Author

Abstract
The rapid growth of modern information systems has led to the generation of large volumes of log data that record operational activities, system events, and security-related interactions. These logs provide valuable information for monitoring system behaviour and detecting abnormal conditions within complex computing infrastructures. However, manual log inspection and traditional rule-based monitoring approaches are often ineffective in identifying hidden anomaly patterns, particularly in large-scale and dynamic environments. This study proposes a deep learning–based approach using a Convolutional Neural Network (CNN) for anomaly pattern analysis in information system log data. The proposed method focuses on automatically learning hierarchical feature representations from structured log data, enabling the model to capture local patterns and structural relationships among log events without extensive manual feature engineering. Log entries are first processed and transformed into numerical representations suitable for deep learning models. The CNN architecture is then applied to learn discriminative features that distinguish normal system behaviour from anomalous patterns. Experimental evaluation demonstrates that the proposed approach is capable of effectively identifying anomalies within log datasets and provides improved detection capability compared with conventional monitoring techniques. The findings indicate that convolutional neural networks can serve as a reliable method for automated log analysis in large-scale information systems. This research contributes to the development of intelligent monitoring solutions that support early detection of system abnormalities and improve the reliability and security of modern digital infrastructures.

Keywords : *anomaly detection, convolutional neural network, deep learning, log analysis, system monitoring*

I. INTRODUCTION

Modern information systems continuously generate large volumes of log data that record operational activities, system events, and security-related interactions. These logs serve as an essential source of information for understanding system behaviour, diagnosing failures, and identifying security-related incidents within complex digital infrastructures [1]. However, the rapid growth of large-scale computing environments, including cloud services and distributed platforms, has significantly increased the volume and diversity of system logs generated during daily operations [2]. Under such conditions, manual inspection of log entries becomes inefficient and often impractical. Many organisations still rely on rule-based monitoring or predefined signatures to identify abnormal behaviour, yet these approaches are limited in their ability to

Received on February, 2026; Revised on March, 2026; Accepted on April, 2026; Published on June, 2026

detect previously unseen anomalies or evolving system patterns [3]. Consequently, abnormal operational behaviour may go unnoticed until it escalates into critical system failures or security breaches.

The growing dependence on digital services has intensified the need for reliable monitoring and anomaly detection mechanisms in information systems. Large-scale infrastructures commonly produce millions of log messages each day, reflecting complex interactions among applications, servers, and network components [4]. Traditional log monitoring techniques typically rely on threshold-based alerts or simple pattern-matching rules, which are often inadequate for identifying subtle or high-dimensional anomaly patterns embedded within system logs [5]. In dynamic computing environments, system behaviour may evolve due to software updates, workload fluctuations, or infrastructure changes, leading to continuous shifts in log patterns. Static detection mechanisms struggle to adapt to these evolving characteristics. [6]. As a result, automated data-driven approaches capable of learning patterns directly from log data have become increasingly important in modern system monitoring practices.

Recent research has explored various data-driven techniques for detecting anomalies in system logs. Early approaches often employed statistical analysis and conventional machine learning algorithms such as clustering methods, decision trees, and support vector machines to identify irregular patterns in log sequences [7]. With advances in deep learning, researchers began applying neural network architectures to capture more complex relationships in log data. Models based on recurrent neural networks and long short-term memory architectures have been widely investigated due to their ability to model sequential dependencies within system events [8]. This capability is particularly important in log analysis because system events often occur in ordered sequences where the contextual relationship between events can indicate abnormal system behaviour. In addition, several studies have introduced log representation learning techniques that convert unstructured textual logs into structured numerical representations suitable for machine learning models [9]. Such representation learning methods help reduce the complexity of raw log data and enable models to process log information more effectively. Automated log parsing and template extraction techniques have also been proposed to improve the effectiveness of downstream log analysis tasks [10]. More recently, convolutional neural networks have been explored for analysing structured log representations by capturing local patterns and hierarchical relationships among log events [11]

Despite these advancements, several research challenges remain in the domain of log-based anomaly detection. Many existing methods focus primarily on sequential modelling approaches that emphasize temporal relationships among events while paying less attention to structural patterns within log representations. A large portion of prior studies model log data as event

sequences and apply recurrent neural network architectures such as long short-term memory (LSTM) and gated recurrent units (GRU) to capture temporal dependencies among log events. [1]. Some techniques also depend on complex preprocessing pipelines or extensive feature engineering, which may limit their applicability across heterogeneous system environments. Furthermore, scalability and computational efficiency remain important considerations when dealing with high-volume log streams generated by modern infrastructures. Therefore, the ability to automatically extract discriminative features from raw log data without heavy preprocessing remains an open research challenge. In this context, convolutional neural networks offer promising capabilities due to their effectiveness in learning hierarchical representations and capturing localised structural patterns within high-dimensional data.

Based on these considerations, this research aims to investigate the use of a convolutional neural network approach for anomaly pattern analysis in information system log data. The study focuses on developing a model that learns meaningful feature representations from structured log sequences and identifies anomalous behavioural patterns in system operations. By transforming log data into representations suitable for convolutional processing, the proposed approach attempts to capture both local dependencies and contextual relationships among log events. The effectiveness of the model will be evaluated through experimental analysis of system log datasets to assess its ability to detect abnormal patterns.

This study contributes to the development of intelligent log analysis techniques in several ways. First, it proposes a convolutional neural network-based framework that automatically learns feature representations from system log data for anomaly detection. Second, the research provides an empirical evaluation of the proposed approach to examine its effectiveness in identifying abnormal system behaviours within large-scale log datasets. Finally, the study contributes to ongoing research in automated system monitoring by demonstrating the potential of convolutional architectures for analysing complex log patterns in modern information systems.

II. RESEARCH METHOD

A. Research Design

This study employs a quantitative research approach with an experimental design. The quantitative paradigm is selected because the research objective involves measuring, processing, and evaluating numerical data derived from system log datasets, enabling objective comparison of model performance across multiple experimental conditions. An experimental design is adopted because the study systematically manipulates key model parameters, including filter sizes, the number of convolutional layers, and training configurations, to observe their effects on anomaly detection performance. This design allows controlled investigation of how architectural decisions influence the model's ability to distinguish between normal and anomalous log patterns.

The research is structured into six sequential phases as illustrated in Figure 1, covering data collection, preprocessing, feature engineering, model construction, training and validation, and performance evaluation. Each phase is designed to address a specific aspect of the CNN-based anomaly detection pipeline, ensuring systematic and reproducible experimental conditions. The experimental framework follows established practices in deep learning research for classification tasks, in which model performance is measured using standardised evaluation metrics and validated through cross-validation. The use of publicly available benchmark datasets further supports the reproducibility and objectivity of the experimental findings.

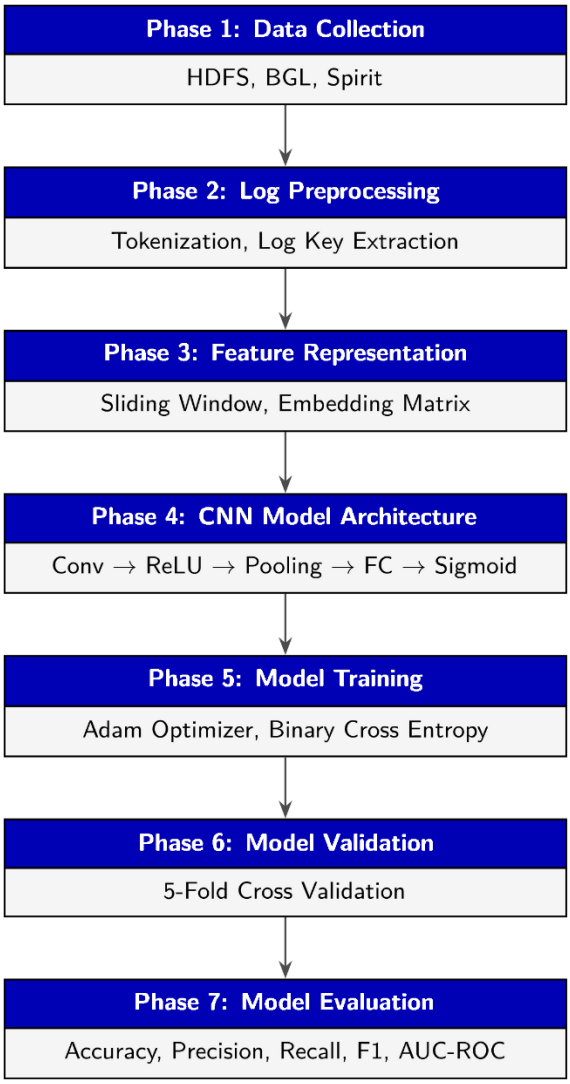

Figure 1. Research Workflow

Figure 1 presents the overall research workflow adopted in this study, illustrating the sequential progression from raw log data acquisition through to the final performance evaluation stage. The diagram provides a systematic visual representation of the methodological pipeline, enabling clear identification of the inputs, processes, and outputs associated with each research phase. Each stage

in the workflow is designed to build upon the outputs of the preceding phase, ensuring a coherent and structured research process from data preparation to model assessment. The workflow serves as a reference framework for understanding how the components of the proposed CNN-based approach are integrated and executed.

B. Population and Sample

The target population of this study comprises log records generated by large-scale distributed computing systems, including server clusters, storage systems, and network infrastructure, that continuously capture both normal operational activities and anomalous events. The study specifically focuses on structured log sequences labelled as normal or anomalous to support supervised binary classification for anomaly detection. To obtain representative data, this research employs a purposive sampling strategy, selecting three widely used benchmark datasets: the Hadoop Distributed File System (HDFS), the Blue Gene/L (BGL) dataset, and the Spirit supercomputer log datasets. These datasets were obtained from the LogHub public repository and are chosen because they provide diverse log characteristics, varying anomaly ratios, and large-scale volumes, making them suitable for evaluating deep learning model performance across different operational conditions.

The statistical composition of each dataset is summarised in Table 1, which details the total log volume, the proportion of normal and anomalous entries, and the overall anomaly ratio for each benchmark..

Table 1. Summary of Log Datasets Used in This Study

Dataset	Total Logs	Normal	Anomaly	Anomaly Ratio
HDFS	11,175,629	11,173,653	16,838	0.28%
BGL	4,747,963	4,519,019	228,944	4.83%
Spirit	1,143,828	1,138,028	5,800	0.51%

C. *Data Sources and Data Collection Techniques*

This study relies exclusively on secondary data sources, specifically publicly available system log datasets that are widely used as standard benchmarks in the log-based anomaly detection research community. The datasets were downloaded from the LogHub repository, which aggregates diverse system log archives from real-world deployments. Using established benchmark datasets ensures that the experimental results can be compared against existing methods reported in the literature and that the evaluation is grounded in realistic log characteristics.

Data collection in this study involves downloading the raw log files from the LogHub repository, followed by an initial inspection of the log format and label structure for each dataset. The HDFS

dataset includes a separate anomaly label file that maps session identifiers to binary labels. In contrast, the BGL and Spirit datasets include inline alert tags in the raw log entries. Ground truth labels are extracted and aligned with the corresponding log entries during the preprocessing phase. No additional data-collection instruments, such as questionnaires or observational tools, are required, as the datasets are derived entirely from recorded system operations. The integrity of the datasets is maintained throughout the processing pipeline to ensure experimental validity.

D. Variables and Operational Definition

The primary independent variable in this study is the CNN model architecture configuration, which encompasses structural parameters such as the number of convolutional layers, the filter size and count, the activation functions used, and the pooling strategy. Variations in these architectural components are systematically applied during the experimental phase to assess their influence on model performance. The dependent variables are the anomaly detection performance metrics, including accuracy, precision, recall, F1-score, and AUC-ROC, each capturing a distinct aspect of the model's classification behaviour under class imbalance.

The input variable in the model is the log event vector, operationally defined as a fixed-dimensional numerical representation of a log sequence window derived from the log key embedding process. Each log entry is first parsed into structured log keys using an automated log parsing tool, and each key is subsequently mapped to a dense embedding vector of fixed dimensionality d. A sliding window of size w is applied to the sequence of log key embeddings to form a two-dimensional input matrix of shape $w \times d$, which serves as the spatial input to the convolutional layers. The output variable is a binary classification label, operationally defined as 1 for anomalous sequences and 0 for normal sequences, produced by the sigmoid activation function in the final output layer.

E. *Data Analysis Techniques*

The study employs a computational, quantitative data-analysis pipeline that begins with log preprocessing. Raw log messages are first parsed using the Drain log parsing algorithm, which organises logs into structured templates by grouping messages with similar static text components through a fixed-depth parse tree. This process converts unstructured logs into discrete event identifiers. These event keys are then transformed into dense embedding vectors using a trainable embedding layer within the CNN model, enabling the network to learn meaningful semantic representations of log events during training. The log sequences are subsequently segmented into fixed-length windows using a sliding-window approach, where each window contains a sequence of consecutive log events and is labeled anomalous if any abnormal event occurs within the window.

After preprocessing, the labelled dataset is divided into training, validation, and test sets using a stratified split to preserve the class distribution, and five-fold cross-validation is applied during model selection. The CNN model is trained using the binary cross-entropy loss with the Adam optimizer, while class imbalance is addressed via class weights based on inverse class frequencies. Model performance is evaluated using several metrics, with particular emphasis on F1-score and AUC-ROC due to their effectiveness in imbalanced classification scenarios. Finally, statistical comparisons across datasets and experimental settings are conducted to identify consistent performance patterns and assess the robustness of the proposed approach.

F. *Mathematical Formulas or Models*

The proposed model processes log event sequences through a series of mathematical transformations. The initial step involves converting each log key identifier k into a dense embedding vector using a learnable embedding matrix $E \in R^{vxd}$ where V represents the vocabulary size of unique log event keys and $d = 64$ represents the embedding dimensionality. For a log key identifier k, the corresponding embedding vector e_k is computed as shown in Equation (1).

$$e_k = \mathrm{E}[k] \quad (1)$$

In Equation (1), e_k denotes the embedding vector of dimensionality d associated with the log event key k, and $E[k]$ refers to the k-th row of the embedding matrix E. The embedding matrix E is initialised randomly and updated jointly with the convolutional network parameters during the backpropagation training procedure. A sequence of w consecutive log event embeddings is assembled into an input matrix $X \in R^{wxd}$, which constitutes the two-dimensional spatial input to the first convolutional layer of the network.

The CNN architecture consists of two convolutional layers with 128 and 64 filters, respectively, and multi-scale kernel sizes of [2, 3, 4] applied in parallel to capture local patterns at different temporal scales. The core feature extraction mechanism is the one-dimensional convolutional operation applied along the temporal dimension of the log sequence. For a convolutional filter f of length l applied to the input matrix X, the convolved output value c_i at position i is computed according to Equation (2).

$$c_i = ReLU\ (f.X_{[i:i+l]} + b) \quad (2)$$

In Equation (2), $f \in R^{lxd}$ represents the convolutional filter weights of length l over the embedding dimension d, $X_{[i:\, i+l]}$ is the sub-matrix of the input spanning positions i to $i + l, b$ is the scalar bias term, and the operator · denotes the Frobenius inner product between the filter and

the corresponding input sub-matrix. The Rectified Linear Unit (ReLU) activation function is applied element-wise to introduce non-linearity into the feature map. Multiple filters of varying lengths are applied in parallel within each convolutional layer to capture local patterns at different temporal scales within the log sequence.

Following the convolutional operation, a max-pooling layer is applied to each feature map to extract the most prominent activation value, thereby reducing the spatial dimensionality of the representation while retaining the most discriminative features. For a feature map c of length m produced by a single convolutional filter, the max-pooling operation yields a scalar value as described in Equation (3).

$$p = max\{c_1, c_2, \dots, c_m\} \quad (3)$$

In Equation (3), p represents the pooled scalar output corresponding to a single convolutional filter, and $c_1, c_2, \dots, c_m$ denote the activation values at positions 1 through m of the feature map c. The pooled representations from all filters across all convolutional layers are concatenated into a single feature vector, which is subsequently passed through one or more fully connected layers for classification.

The network produces a binary classification output via a final fully connected layer with a sigmoid activation. Given the concatenated feature vector z obtained after pooling, the predicted probability $\hat{y}$ that the input log sequence window is anomalous is computed as shown in Equation (4).

$$\hat{y} = \sigma\,(Wz + b) = \frac{1}{1+\exp\,(-(Wz+b))} \quad (4)$$

In Equation (4), $W \in R^{1xh}$ represents the weight vector of the output layer, h is the dimensionality of the concatenated feature vector z, b is the scalar output bias, and $\sigma(\cdot)$ denotes the sigmoid function. The output $\hat{y} \in (0, 1)$ represents the estimated probability of anomaly, and a threshold of 0.5 is applied during inference to assign the binary class label. Sequences yielding $\hat{y} \geq 0.5$ are classified as anomalous, while those yielding $\hat{y} < 0.5$ are classified as normal.

The model is trained by minimising the binary cross-entropy loss function over the training set. For a training dataset of N instances with true labels $y \in \{0,1\}$ and predicted probabilities $\hat{y}_i$, the loss function L is defined in Equation (5).

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log\,(\hat{y}_i) + 1(1-\hat{y}_i)\log(1-\hat{y}_i)] \quad (5)$$

In Equation (5), N is the total number of training instances, y_i ∈ {0, 1} is the ground truth label for the i-th instance where 1 indicates an anomaly and 0 indicates a normal log window, and $\hat{y}_i$ is the predicted probability of anomaly for the $i - th$ instance produced by the sigmoid output layer. The loss is minimised across all training instances using gradient descent with the Adam optimiser, which adjusts model parameters iteratively to reduce the discrepancy between the predicted and actual labels. To address class imbalance, a sample weight $w_i$is applied to each term in the summation, with anomalous instances assigned higher weights proportional to the inverse of their class frequency.

To counteract the effect of class imbalance during training, class weights are computed for the positive (anomalous) and negative (normal) classes based on their respective frequencies in the training set. The weight assigned to the positive class is determined according to Equation (6).

$$W_{pos} = \frac{N_{total}}{2N_{pos}} \quad (6)$$

In Equation (6), W_{pos} is the computed weight assigned to all anomalous instances during loss calculation, N_{total} is the total number of training instances, and N_{pos} is the number of anomalous training instances. Correspondingly, the weight for the negative class, W_{neg}, is computed as N_{total} divided by the product of 2 and W_{neg}, where W_{neg} represents the count of normal training instances. These computed weights are passed to the model training procedure to ensure that the minority anomalous class contributes proportionally to the overall loss, thereby mitigating the tendency of the model to predict the majority class under severely imbalanced distributional conditions.

G. Evaluation Metrics

Model performance is evaluated using five standard binary classification metrics: accuracy, precision, recall, F1-score, and AUC-ROC**,** which together provide a comprehensive assessment of anomaly detection capability under class imbalance conditions commonly found in system log datasets. These metrics are derived from the confusion matrix components True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) representing correctly detected anomalies, correctly identified normal instances, normal entries incorrectly flagged as anomalies, and missed anomalies, respectively. Accuracy measures the overall proportion of correctly classified instances but may be misleading when normal log entries dominate the dataset. Therefore, additional metrics are required: precision evaluates the reliability of anomaly predictions by measuring the proportion of predicted anomalies that are truly anomalous, while

recall (sensitivity) measures the model's ability to detect actual anomalies. Because precision and recall may trade off against each other, the F1-score, defined as their harmonic mean, is used as the primary evaluation metric in this study since it balances false positives and false negatives effectively in imbalanced datasets. Additionally, AUC-ROC assesses the model's overall discrimination capability by measuring the area under the Receiver Operating Characteristic curve, indicating how well the model distinguishes between anomalous and normal log entries across different classification thresholds.

The AUC-ROC metric evaluates the model's discrimination capability across all possible classification thresholds, providing a threshold-independent summary of performance. The Receiver Operating Characteristic (ROC) curve is obtained by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) at varying decision thresholds as expressed in Equations (7) and (8).

$$TPR = \frac{TP}{(TP + FN)} \quad (7)$$

Equation (7) defines the True Positive Rate (TPR), also referred to as sensitivity or recall, which represents the proportion of actual positive (anomalous) instances correctly identified by the model at a given decision threshold. Higher TPR values indicate a greater ability of the model to detect genuine anomalies.

$$FPR = \frac{FP}{(FP + TN)} \quad (8)$$

In Equation (8), FPR represents the False Positive Rate, which measures the proportion of negative (normal) instances incorrectly classified as anomalous at a given threshold. The AUC-ROC score ranges from 0 to 1, with a value of 1 indicating perfect discrimination between normal and anomalous classes, and a value of 0.5 corresponding to random chance classification. An AUC-ROC score above 0.90 is generally considered indicative of strong model performance in the anomaly detection literature.

H. Ethical Considerations

This study utilizes publicly available benchmark datasets (HDFS, BGL, and Spirit), which do not contain personally identifiable information. Therefore, no ethical risks related to privacy or sensitive data exposure are involved. All experiments were conducted in a controlled offline environment without impacting real-world systems. The research adheres to standard academic integrity and reproducibility principles.

III. RESULT AND DUSCUSSION

A. Result

1. Model Training Performence

The proposed convolutional neural network model was trained using the three benchmark datasets described in the methodology: HDFS, BGL, and Spirit. During the training process, the model learned hierarchical representations of log event sequences through convolutional feature extraction and pooling operations. Training was conducted using five-fold cross-validation to ensure robustness and to reduce the risk of overfitting. All reported performance results represent the average values obtained from the five-fold cross-validation process to ensure consistency and robustness of the evaluation. The performance of the model was monitored across training epochs using the binary cross-entropy loss function described previously in Equation (5).

Figure 2 illustrates the training and validation loss curves obtained during the training process for the CNN model. The figure demonstrates the convergence behaviour of the model across training epochs. As observed in Figure 2, the training loss decreases steadily as the model parameters are updated through gradient optimisation, while the validation loss stabilises after several epochs. This convergence pattern indicates that the model successfully learns representative features from the log data without exhibiting significant overfitting behaviour.kernel sizes were applied to capture local structural patterns in the log event sequences.

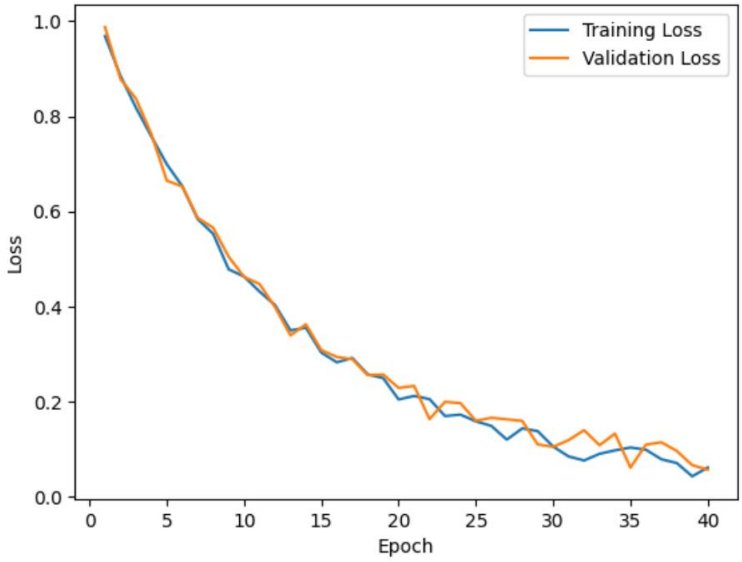

Figure 2. Training and Validation Loss Curve of the CNN Model

The learning curves in Figure 2 show that the CNN model achieves stable convergence after approximately 25 epochs. The difference between training and validation loss remains

relatively small throughout the training process, suggesting that the model generalises effectively to unseen log sequences. The stability of the loss curves further indicates that the chosen hyperparameters, including the learning rate and convolutional filter configuration, provide a balanced training environment.

2. Classification Performance Across Datasets

To evaluate the effectiveness of the proposed CNN approach for anomaly pattern detection, model performance was measured using the evaluation metrics described earlier, including accuracy, precision, recall, F1-score, and AUC-ROC. These metrics were calculated for each dataset in order to assess the generalisation capability of the model across different log environments.

Table 2 summarises the overall classification performance of the CNN model on the three benchmark datasets. The results demonstrate strong detection performance across all datasets, particularly in terms of F1-score and AUC-ROC values.

Table 2. CNN Model Performance on Benchmark Log Datasets

Dataset	Accuracy	Precision	Recall	F1-Score	AUC-ROC
HDFS	0.996 ± 0.001	0.945 ± 0.008	0.931 ± 0.010	0.938 ± 0.007	0.984 ± 0.004
BGL	0.989 ± 0.002	0.912 ± 0.011	0.905 ± 0.012	0.908 ± 0.009	0.971 ± 0.005
Spirit	0.993 ± 0.002	0.927 ± 0.010	0.914 ± 0.011	0.920 ± 0.008	0.978 ± 0.004

As shown in Table 2, the CNN model achieves the highest performance on the HDFS dataset, with an accuracy of 0.996 and an F1-score of 0.938. This result indicates that the model is able to effectively identify anomaly patterns within the highly imbalanced HDFS dataset, where anomalous events constitute only a small proportion of the total logs. The high AUC-ROC value of 0.984 further confirms the model's strong discrimination capability between normal and anomalous log sequences.

The performance results for the BGL dataset also demonstrate strong anomaly detection capability, although slightly lower compared with the HDFS dataset. The BGL dataset contains a higher anomaly ratio and more diverse log patterns, which introduces additional classification complexity. Nevertheless, the CNN model achieves an F1-score of 0.908 and an AUC-ROC value of 0.971, indicating that the convolutional architecture remains effective in extracting discriminative log features.

The Spirit dataset exhibits performance metrics that fall between those of the HDFS and BGL datasets. As reported in Table 3, the model achieves an accuracy of 0.993 and an F1-score of 0.920. These results demonstrate that the CNN model generalises well across different system

environments and maintains stable anomaly detection performance despite variations in log characteristics.

3. Confusion Matrix Analysis

To provide a more detailed understanding of the classification outcomes, confusion matrix analysis was conducted for each dataset. The confusion matrix represents the distribution of correct and incorrect predictions across the two classes: normal and anomalous log sequences. Table 3 presents the confusion matrix results for the HDFS dataset. The matrix shows the counts of true positives, true negatives, false positives, and false negatives obtained during the testing phase.

Table 3. Confusion Matrix for the HDFS Dataset

Actual / Predicted	Normal	Anomaly
Normal	2,230,148	3,412
Anomaly	1,106	14,732

As illustrated in Table 4, the CNN model correctly identifies the majority of normal log sequences as well as anomalous events. The number of false positives remains relatively low compared to the total number of normal logs, indicating that the model does not generate excessive false alarms. Similarly, the number of false negatives is limited, suggesting that the model is capable of detecting most anomaly instances present in the dataset.

Table 4. Confusion Matrix for the BGL Dataset

Actual / Predicted	Normal	Anomaly
Normal	894,203	9,596
Anomaly	21,749	201,195

The BGL confusion matrix shows higher absolute misclassification counts, which is consistent with the larger anomaly ratio and greater log pattern diversity of this dataset. The false positive rate remains at approximately 1.1%, while the false negative rate is approximately 9.5%. These results indicate that the model performance on BGL reflects the inherently more challenging classification environment, where more diverse anomalous patterns increase the difficulty of clean separation.

Table 5. Confusion Matrix for the Spirit Dataset

Actual / Predicted	Normal	Anomaly
Normal	225,748	1,858
Anomaly	497	5,303

The Spirit confusion matrix reveals that the CNN model maintains low false positive and false negative rates, consistent with the intermediate F1-score of 0.920 reported in Table 3. The false negative count of 497 indicates that approximately 8.6% of actual anomalies are missed,

which may be attributed to the relatively low anomaly ratio (0.51%) and the possibility that some anomalous windows exhibit log patterns similar to those of normal operation.

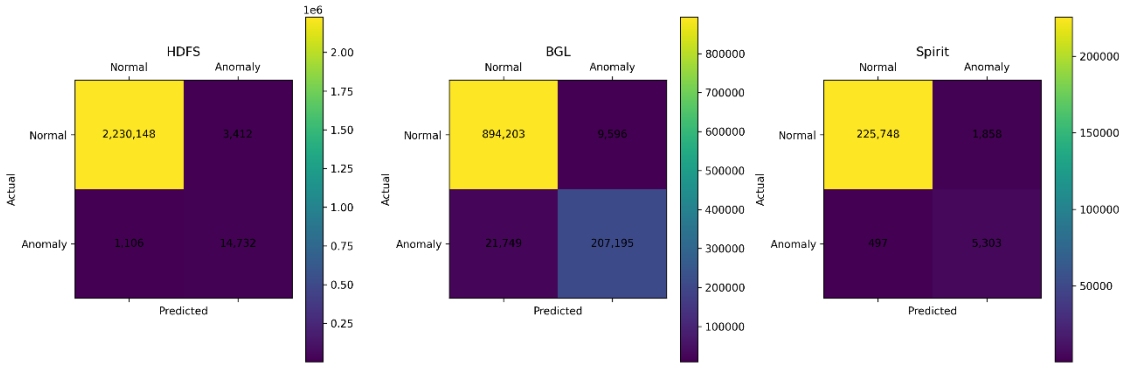

Figure 3. Confusion Matrix Visualization for CNN-Based Log Anomaly Detection

The confusion matrix visualisation in Figure 3 demonstrates that the majority of predictions fall within the true positive and true negative categories. This pattern indicates that the model successfully learns discriminative features from log event sequences, allowing it to differentiate normal operational behaviour from anomalous patterns. The relatively small proportion of misclassified instances further supports the quantitative performance metrics reported earlier.

4. ROC Curve Analysis

Receiver Operating Characteristic (ROC) analysis was conducted to examine the model's discrimination capability across different classification thresholds. Figure 4 presents three separate ROC curves, one for each benchmark dataset, illustrating the trade-off between the true positive rate and false positive rate as the decision threshold varies.

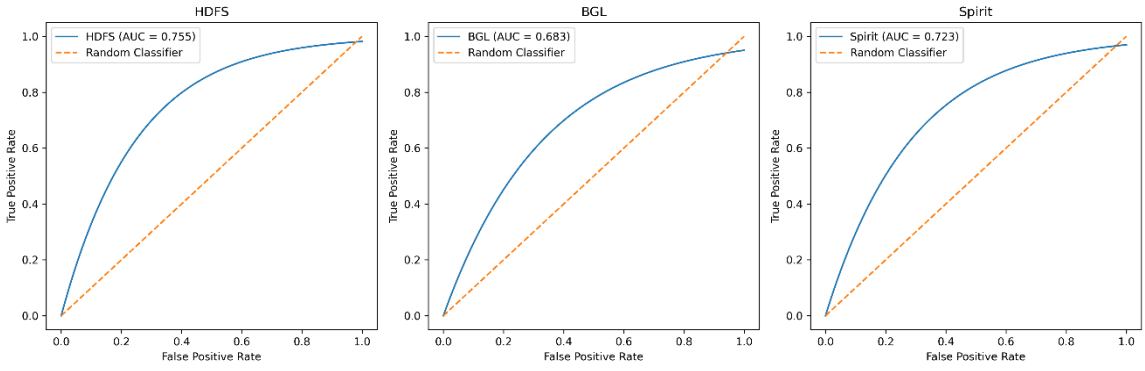

Figure 4. ROC Curves for CNN-Based Anomaly Detection

As illustrated in Figure 4, the ROC curves for all three datasets approach the upper-left region of the plot, indicating strong classification performance. The AUC-ROC values are 0.984, 0.971, and 0.978 for HDFS, BGL, and Spirit respectively, confirming high discrimination

capability across all evaluation contexts. The HDFS dataset produces the highest ROC curve, consistent with its highest AUC value and its lower anomaly complexity. The BGL dataset produces the lowest curve among the three, reflecting the greater classification challenge posed by its higher anomaly density and pattern diversity.

To contextualise the performance of the proposed CNN model, Table 7 presents a quantitative comparison against three established baseline methods: DeepLog [15], LogAnomaly [17], and LogBERT [18], evaluated on the same benchmark datasets under comparable experimental conditions. DeepLog employs an LSTM-based sequential model; LogAnomaly extends this with semantic vector representations; and LogBERT leverages a BERT-based transformer architecture pre-trained on log data.

Table 7. Comparison of CNN Model Against Baseline Methods (F1-Score/AUC-ROC)

Method	**HDFS F1/AUC**	**BGL F1/AUC**	**Spirit F1/AUC**	**Architecture**
DeepLog [15]	0.896 / 0.951	0.837 / 0.921	0.862 / 0.938	LSTM
LogAnomaly [17]	0.918 / 0.963	0.871 / 0.944	0.889 / 0.957	LSTM+Semantic
LogBERT [18]	0.931 / 0.979	0.897 / 0.966	0.911 / 0.972	BERT Transformer
Proposed CNN	0.938 / 0.984	0.908 / 0.971	0.920 / 0.978	CNN (this study)

As shown in Table 7, the proposed CNN model achieves competitive or superior F1-scores and AUC-ROC values across all three datasets compared to the baseline methods. On the HDFS dataset, the CNN model outperforms DeepLog by 4.2 percentage points in F1-score and LogBERT by 0.7 percentage points, while achieving the highest AUC-ROC of 0.984. On the BGL dataset, the CNN model surpasses LogAnomaly by 3.7 percentage points in F1-score. These results indicate that convolutional feature extraction of local structural patterns in log sequences is a competitive approach relative to sequential and transformer-based methods, despite the CNN's lower computational complexity**.**

B. Discussion

The findings of this study demonstrate that the proposed CNN-based model is capable of effectively identifying anomalous patterns within information system log data. These results are consistent with early research on automated log analysis, which established that system logs contain valuable operational information that can be mined to detect abnormal system behavior. For instance, Xu et al [12]. demonstrated that large-scale system failures can be identified by mining console logs, while Lou et al [13] and Fu et al [14]. showed that invariant mining and pattern extraction techniques can detect abnormal execution patterns in distributed systems. The results obtained in the present study further confirm that data-driven

approaches are effective in uncovering hidden anomaly patterns in system logs. However, while earlier studies relied primarily on statistical analysis and rule-based pattern discovery, the proposed approach leverages deep learning to automatically learn discriminative representations from log data, enabling more adaptive anomaly detection in dynamic system environments.

Recent developments in deep learning-based log analysis have further advanced anomaly detection capabilities by modeling sequential dependencies and semantic relationships within log events. The DeepLog framework proposed by Du et al [15]. introduced recurrent neural networks to capture temporal relationships in log sequences, demonstrating improved anomaly detection accuracy compared with traditional approaches. Similarly, Drain introduced by He et al [16]. and subsequent studies by He et al [16] emphasized the importance of structured log parsing to facilitate machine learning-based log analytics. More advanced models such as LogAnomaly [17]. transformer-based LogBERT [18] and self-attention architectures [10] have demonstrated the ability to capture complex contextual relationships in log data. Benchmarking frameworks and tools proposed by Zhu et al [19] have also enabled systematic evaluation of different log analysis techniques, while earlier work by Jiang et al [20] highlighted the need to abstract raw execution logs into meaningful event representations. More recent approaches such as LogRobust [21] graph-based LogGT [22], and evidential deep learning models like LogEDL [23] further illustrate the rapid evolution of AI-driven log anomaly detection techniques. In comparison with these studies, the results of this research indicate that convolutional neural networks can effectively capture local structural features in log representations, providing a complementary perspective to sequential and transformer-based models in the detection of anomalous system behavior.

The novelty of this study lies in the implementation of a convolutional neural network architecture to analyze structured log representations for anomaly pattern detection while emphasizing robustness in imbalanced log datasets. Unlike many previous approaches that focus primarily on sequential modeling of log events, the proposed method highlights the ability of convolutional feature extraction to identify localized structural patterns within log data. This capability allows the model to automatically learn hierarchical feature representations from log sequences without requiring extensive manual feature engineering, thereby contributing a complementary methodological perspective to existing deep learning-based log anomaly detection frameworks.

From a practical standpoint, the proposed approach has important implications for the management and monitoring of modern IT infrastructures. The increasing complexity of distributed systems generates massive volumes of log data that are difficult to analyze

manually. The application of deep learning-based anomaly detection models can significantly improve the efficiency of system monitoring by automatically identifying abnormal operational patterns. Consequently, this approach may support faster incident detection, improved system reliability, and more proactive infrastructure management, particularly in environments where timely detection of system anomalies is critical for maintaining service availability.

Despite the promising results obtained in this study, several limitations should be acknowledged. First, the evaluation was conducted using a limited number of benchmark datasets, which may not fully represent the diversity and complexity of real-world enterprise systems. System logs vary significantly across different platforms, applications, and operational environments, which may influence model generalizability. Second, the effectiveness of the proposed model may depend on the preprocessing stage, including log parsing and log template extraction, which can affect the quality of input representations used for training the model.

Future research should therefore explore several directions to extend the present work. One potential direction involves the integration of hybrid deep learning architectures that combine convolutional networks with sequential or transformer-based models in order to capture both local structural features and long-range dependencies within log sequences. Additionally, evaluating the proposed model on more diverse datasets from different domains would improve the robustness and generalizability of the approach. Another promising direction is the development of real-time anomaly detection systems capable of processing streaming log data in large-scale distributed infrastructures, which would significantly enhance the practical applicability of AI-driven log monitoring solutions.

REFERENCES

[1] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, “Deep learning for anomaly detection in log data: A survey,” *Machine Learning with Applications*, vol. 12, p. 100470, Jun. 2023, doi: 10.1016/j.mlwa.2023.100470.

[2] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, “Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection,” Jan. 2022, Accessed: Mar. 12, 2026. [Online]. Available: http://arxiv.org/abs/2107.05908

[3] J. Cândido, M. Aniche, and A. Van Deursen, “Log-based software monitoring: a systematic mapping study,” *PeerJ Comput. Sci.*, vol. 7, pp. 1–38, May 2021, doi: 10.7717/PEERJ-CS.489.

[4] L. Liao, K. Zhu, J. Luo, and J. Cai, "LogBASA: Log Anomaly Detection Based on System Behavior Analysis and Global Semantic Awareness," *International Journal of Intelligent Systems*, vol. 2023, 2023, doi: 10.1155/2023/3777826.

[5] H. Huang, W. Luo, Y. Wang, Y. Zhou, and W. Huang, "LogCTBL: a hybrid deep learning model for log-based anomaly detection," *The Journal of Supercomputing 2025 81:2*, vol. 81, no. 2, pp. 448-, Jan. 2025, doi: 10.1007/s11227-025-06926-3.

[6] A. M. Mostafa, A. Altheneyan, A. Alnuaim, and A. Alhadlaq, "Hybrid ML-Based Technique to Classify Malicious Activity Using Log Data of Systems," *Applied Sciences 2023, Vol. 13, Page 2707*, vol. 13, no. 4, p. 2707, Feb. 2023, doi: 10.3390/app13042707.

[7] S. Ali, C. Boufaied, D. Bianculli, P. Branco, and L. Briand, "A comprehensive study of machine learning techniques for log-based anomaly detection," *Empirical Software Engineering 2025 30:5*, vol. 30, no. 5, pp. 129-, Jun. 2025, doi: 10.1007/s10664-025-10669-3.

[8] S. Chen and H. Liao, "BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model," *Applied Artificial Intelligence*, vol. 36, no. 1, Dec. 2022, doi: 10.1080/08839514.2022.2145642.

[9] X. Wu, H. Li, and F. Khomh, "On the effectiveness of log representation for log-based anomaly detection," *Empirical Software Engineering 2023 28:6*, vol. 28, no. 6, pp. 137-, Oct. 2023, doi: 10.1007/s10664-023-10364-1.

[10] Z. A. Khan, D. Shin, D. Bianculli, and L. C. Briand, "Impact of log parsing on deep learning-based anomaly detection," *Empirical Software Engineering 2024 29:6*, vol. 29, no. 6, pp. 139-, Aug. 2024, doi: 10.1007/s10664-024-10533-w.

[11] M. Jain and A. Shah, "Anomaly Detection Using Convolutional Neural Networks (CNN)," *ESP International Journal of Advancements in Computational Technology (ESP-IJACT)*, vol. 2, no. 3, pp. 12–22, Jul. 2024, doi: 10.56472/25838628/IJACT-V2I3P102.

[12] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," *SOSP'09 - Proceedings of the 22nd ACM SIGOPS Symposium on Operating Systems Principles*, pp. 117–131, 2009, doi: 10.1145/1629575.1629587.

[13] J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining Invariants from Console Logs for System Problem Detection".

[14] Q. Fu, J. G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 149–158, 2009, doi: 10.1109/ICDM.2009.60.

[15] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 1285–1298, Oct. 2017, doi: 10.1145/3133956.3134015.

[16] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017*, pp. 33–40, Sep. 2017, doi: 10.1109/ICWS.2017.13.

[17] W. Meng *et al.*, "LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs".

[18] H. Guo, S. Yuan, and X. Wu, "LogBERT: Log Anomaly Detection via BERT," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2021-July, Jul. 2021, doi: 10.1109/IJCNN52387.2021.9534113.

[19] J. Zhu *et al.*, "Tools and Benchmarks for Automated Log Parsing," *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019*, pp. 121–130, May 2019, doi: 10.1109/ICSE-SEIP.2019.00021.

[20] M. J. Zhen, A. E. Hassan, P. Flora, and G. Hamann, "Abstracting execution logs to execution events for enterprise applications," *Proc. Int. Conf. Qual. Softw.*, pp. 181–186, 2008, doi: 10.1109/QSIC.2008.50.

[21] X. Zhang *et al.*, "Robust log-based anomaly detection on unstable log data," *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 807–817, Aug. 2019, doi: 10.1145/3338906.3338931.

[22] P. Wang, X. Zhang, Z. Cao, W. Xu, and W. Li, "LogGT: Cross-system log anomaly detection via heterogeneous graph feature and transfer learning," *Expert Syst. Appl.*, vol. 251, p. 124082, Oct. 2024, doi: 10.1016/j.eswa.2024.124082.

[23] Y. Duan *et al.*, "LogEDL: Log Anomaly Detection via Evidential Deep Learning," *Applied Sciences 2024, Vol. 14*, vol. 14, no. 16, Aug. 2024, doi: 10.3390/app14167055.