

Enhancing Lightweight Log Anomaly Detection with Deep Semantic Encoding and Nonlinear Feature Compression

Aziza Zizouan*, Imad Hafidi, Noureddine Aboutabit

IDIA Laboratory, National School of Applied Sciences, Sultan Moulay Slimane University, Morocco

Abstract

Anomaly detection within system logs remains a critical task aimed at maintaining the stability as well as reliability of modern software systems. As large-scale cloud and distributed computing environments continue to expand, system logs have become increasingly voluminous, heterogeneous, and complex, making traditional rule-based and manual analysis approaches less effective. Although learning-based methods have shown promising potential, most existing approaches still struggle to maintain an effective trade-off across detection performance and computational efficiency, particularly in resource-constrained environments. In this work, a lightweight and context-aware log anomaly detection approach, referred to as Lightweight Semantic TCN (LS-TCN) is introduced based on an enhanced Temporal Convolutional Network (TCN). The proposed method relies on compact semantic log representations and incorporates a nonlinear dimensionality reduction strategy based on PCA-tSNE to preserve discriminative information while reducing model complexity. In addition, the TCN architecture is designed to efficiently capture temporal dependencies in log sequences. Experimental results across the BGL and HDFS benchmark datasets indicate that our method achieves competitive or improved detection performance, while maintaining lower computational cost.

Keywords Log anomaly detection, depthwise convolution, resource-constrained systems, Temporal Convolutional Network

DOI: 10.19139/soic-2310-5070-4192

1. Introduction

The evolution of modern software ecosystems, including large-scale cloud-based infrastructures and Internet of Things (IoT)-oriented systems, has resulted in the complex management and monitoring of these systems. In this context, the continuous generation of massive amounts of log data, which records different types of system operations and states, has become inevitable in the development of complex software systems [1, 2]. In this context, the role of log data has become essential in ensuring the reliability of complex software systems and in identifying any types of failures and potential security threats [3, 4]. However, as log data continues to grow exponentially and the diversity of operations and states recorded in these data, the conventional approaches of inspecting and analyzing the data have become impossible and prone to errors.

Early automated approaches were based on conventional machine learning techniques, such as Support Vector Machines [24], clustering-based models [25], and statistical approaches including PCA [30] and Isolation Forest [26]. However, these methods depend heavily on manual feature engineering and struggle to process the unstructured nature of log messages as well as the long-range temporal dependencies inherent in system execution traces. To address these challenges, deep learning approaches emerged as a natural alternative, offering the ability to learn representations directly from raw data. Models such as RNNs [29], LSTMs [27], and GRUs [28] have been

*Correspondence to: Aziza Zizouan (Email: zizouanaziza@gmail.com). IDIA Laboratory, National School of Applied Sciences, Sultan Moulay Slimane University. Bd Béni Amir, BP 77, Khouribga, 25000, Morocco.

introduced to capture sequential patterns in log streams. Despite their advantages, these recurrent architectures often suffer from high computational cost, vanishing gradients, and limited scalability when applied to large-scale log datasets.

In recent times, with the increasing demand for efficient and scalable solutions to analyze logs, there have been several efforts to develop solutions that are efficient enough to run under tight resource constraints. The solution LightLog [10] is one such example of an early approach for solving the problem of anomaly detection in logs under tight resource constraints. This is proof that log monitoring can indeed be achieved efficiently with low computational costs. But at the same time, this efficiency has come at the expense of expressive power when it comes to semantics and modeling. The light-weight nature of the algorithm makes it unable to model complex log data.

In this work, our LS-TCN is presented, where all the stages of the log anomaly detection process are revisited to overcome the limitations of existing approaches. Firstly, the Drain parser is employed to ensure the consistent and reliable extraction of structured templates from raw log data. Secondly, the framework incorporates a BERT-based contextual semantic encoding stage, where conventional word-level embeddings are replaced by richer contextual representations before dimensionality reduction. To obtain compact contextual representations, we compare the PCA-PPA strategy used in LightLog [10] with an offline PCA-tSNE alternative based on the established t-SNE method [18]. The contribution therefore lies in integrating and evaluating this nonlinear compression stage within the proposed lightweight semantic-temporal pipeline, rather than in introducing a new formulation of t-SNE. Finally, our solution introduces a Lightweight Semantic version of the TCN architecture where residual connections, dilated causal convolutions, and depthwise convolutions are combined with global max and average pooling operations to improve temporal feature extraction while maintaining efficiency. Together, these contributions form a coherent and efficient solution that significantly strengthens the semantic expressiveness, temporal modeling capacity, and overall anomaly detection performance while still suitable for resource-constrained environments. BERT itself is not introduced as a novel component; rather, its role is to provide contextual semantic representations within the proposed lightweight processing pipeline.

2. Related Work

Deep learning has shown a strong effectiveness in the log anomaly detection, where models can automatically learn both temporal dynamics and semantic relationships from the raw log data. A variety of architectures have been explored here, including recurrent models (e.g., LSTM and Bi-LSTM) and attention-driven approaches such as Transformers to effectively model sequential structures and contextual dependencies, and thereby enhancing detection performance compared to conventional methods.

DeepLog [5] proposed one of the first deep learning frameworks for log anomaly detection, where log sequences were modeled using stacked LSTM-based architectures. Each log event is encoded as a one-hot vector, and anomaly detection occurs when the predicted event differs from the actual event. This method successfully captures temporal information in log sequences and demonstrates high detection accuracy but is not semantically aware and is not scalable when the number of log templates is large.

LogAnomaly [6] improves upon DeepLog by incorporating a new feature called Template2Vec, which relies on a lexical contrast embedding model (dLCE) [7] to create semantic template vectors by deriving a weighted average of word embeddings. This allows for the detection of sequential as well as quantitative anomalies by using a combination of template vector sequences and template count features. During inference, LogAnomaly works by comparing the predicted and actual log template IDs to detect anomalies, while unseen templates are approximated using similarity. While this improves the semantic capabilities of the system, the high computational cost of using LSTM remains a problem for large-scale applications.

RobustLog [8] is another log anomaly detection method that extends prior approaches by leveraging a Bi-LSTM-based architecture combined with an attention module to improve log anomaly detection. This improves its robustness to unstable log data and provides better detection accuracy compared to other techniques such as

DeepLog and LogAnomaly. However, its computational complexity is still high because of its use of LSTM and attention layers.

HitAnomaly [9] is a hierarchical framework based on the Transformer model, which can jointly process the sequential and quantitative characteristics of the log data. The proposed method uses two attention layers, the first of which is for the contextual dependencies between the log templates, and the second of which is for the dependencies between the values of the log templates. This method can efficiently detect anomalies with high accuracy. In contrast to such sequence-level attention-based approaches, LS-TCN employs BERT as an offline template-level semantic encoder with fixed precomputed representations, while temporal dependencies among log events are subsequently modeled by the lightweight TCN.

Although the above-mentioned learning-driven approaches are known to enhance log anomaly detection accuracy, they still involve high computational overhead and high-dimensional representations, which make them less suitable for edge devices. In addition, the use of recurrent or attention-based architectures and high-dimensional representations can increase computational overhead and limit efficiency in resource-constrained settings. In this context, LightLog [10] was introduced as an efficient log anomaly detection approach that combines dimensionality reduction methods such as PCA and Post-Processing Algorithms (PPA) [11] with a TCN-based architecture for edge-oriented log anomaly detection. However, LightLog focuses more on efficiency than on detection accuracy. As also noted in the original paper, the model gives more importance to overall runtime efficiency than to reducing false negatives.

More broadly, lightweight deep learning can also be achieved through model-compression strategies such as pruning, quantization, and knowledge distillation, which reduce model size or computational requirements through different mechanisms [32, 33, 34]. These approaches are complementary to the design adopted in LS-TCN, which targets efficiency directly through compact semantic representations and a lightweight temporal architecture rather than through explicit model-compression techniques or teacher–student transfer. Such compression strategies could be further combined with LS-TCN in future work.

Table 1. Methodological comparison between LightLog and the proposed LS-TCN.

Component	LightLog	LS-TCN
Semantic representation	Word2Vec	BERT contextual embeddings
Dimensionality reduction	PCA–PPA	PCA–tSNE
Temporal model	Lightweight TCN	Enhanced lightweight TCN
Temporal convolutions	TCN-based convolutions	Dilated causal + depthwise convolutions
Feature aggregation	Global average pooling	Global average + global max pooling

As summarized in Table 1, the distinction of LS-TCN lies in redesigning the lightweight detection pipeline at both the representation and temporal-modeling levels. Specifically, LS-TCN replaces the Word2Vec–PCA–PPA representation used in LightLog with offline contextual BERT representations followed by nonlinear PCA–tSNE compression, while the temporal architecture incorporates dilated causal and depthwise convolutions together with complementary global average and max pooling. The contribution therefore lies in the integration and evaluation of these modifications within a lightweight log anomaly detection pipeline rather than in introducing BERT or t-SNE individually. The controlled experiment in Table 7 further isolates the architectural contribution by evaluating LightLog-TCN and LS-TCN under the same BERT–PCA–tSNE representation.

3. Methodology

3.1. Log Preprocessing

Before training the classification model, raw system logs go through a structured preprocessing pipeline to obtain informative low-dimensional representations. As shown in Figure 1, the process begins with an automatic parsing step, where the Drain method derives structured templates from the original log messages. These

templates are then organized into fixed-length sessions in order to preserve temporal order as well as contextual relationships. Each template is subsequently transformed into a semantic vector using a BERT encoder. Finally, dimensionality reduction techniques, including PCA-PPA and PCA-t-SNE, are applied to project the high-dimensional embeddings into a more compact semantic space, resulting in low-dimensional vectors that are used as input features for the classification model.

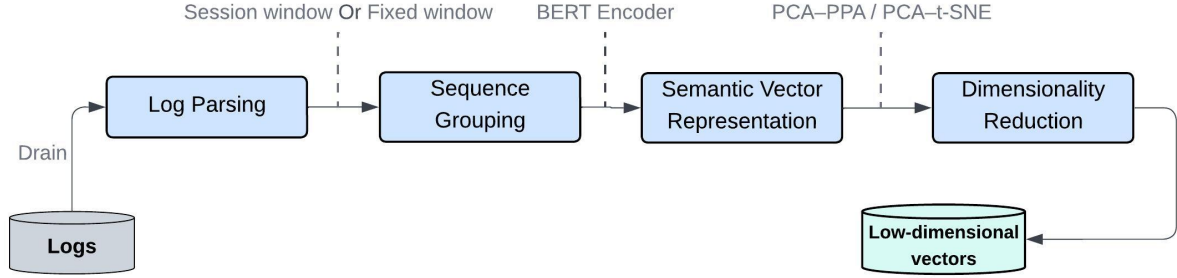


Figure 1. Log preprocessing workflow.

Log Parsing

The first step in the preprocessing phase is to transform raw and unstructured log messages into a structured template, which can then be used for further processing and analysis. This, as depicted in Figure 2, has been carried out using the Drain log parser [12], which structures log messages into a fixed-depth hierarchical tree structure. This allows for efficient parsing even when no prior knowledge of the log format is available. For instance, a raw log message such as 2005-06-03-15.47.20.461505 R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction cache parity error corrected has been transformed into a structured template such as <*> <*> RAS KERNEL INFO instruction cache parity error corrected, with variable parts such as timestamps and identifiers replaced with generic placeholders such as <*>. This helps to reduce noise while retaining the important information associated with a log message, ensuring consistency and stability in sequence modeling. This implementation has been carried out using the open-source LogPai framework [13] and hence can scale effectively while ensuring reproducibility.

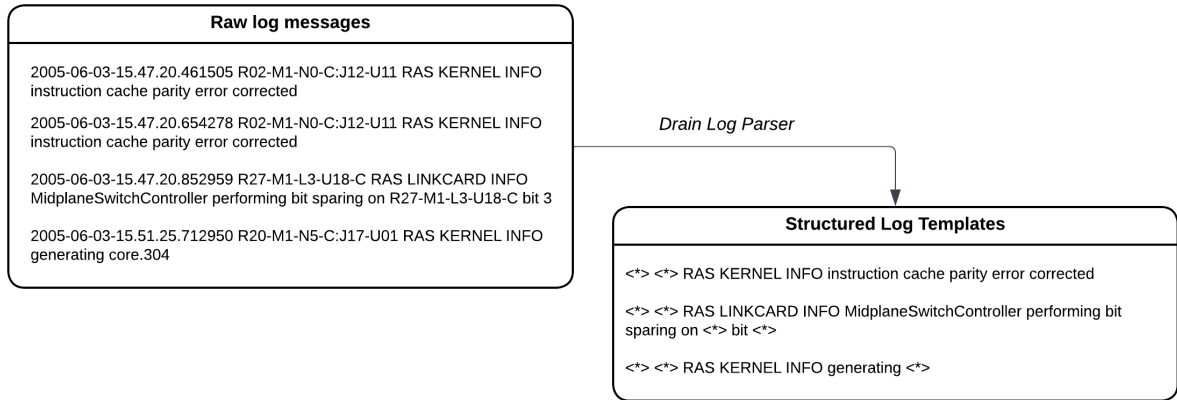


Figure 2. Log parsing process using the Drain algorithm on a sample log from the BGL dataset.

Sequence Grouping

Following the parsing, the logs are then segmented into sequences. In this work, we have used two major approaches to segment the logs, namely the fixed window approach and the session window approach. The fixed window approach, as shown in Figure 3, segments the flow of the parsed logs into non-overlapping sequences, each consisting of a certain number of log events, for example, four events at a time [14]. This helps to ensure that each sequence of the logs has a uniform size, thereby allowing for a more structured form of temporal information. The session window approach, on the other hand, involves the segmentation of the log events based on the session ID or the operational activities of the events, for example, a certain block ID [14]. This helps to ensure that all the logs pertaining to a certain session ID are grouped together as a single sequence, thereby maintaining the logical relationships between the events.

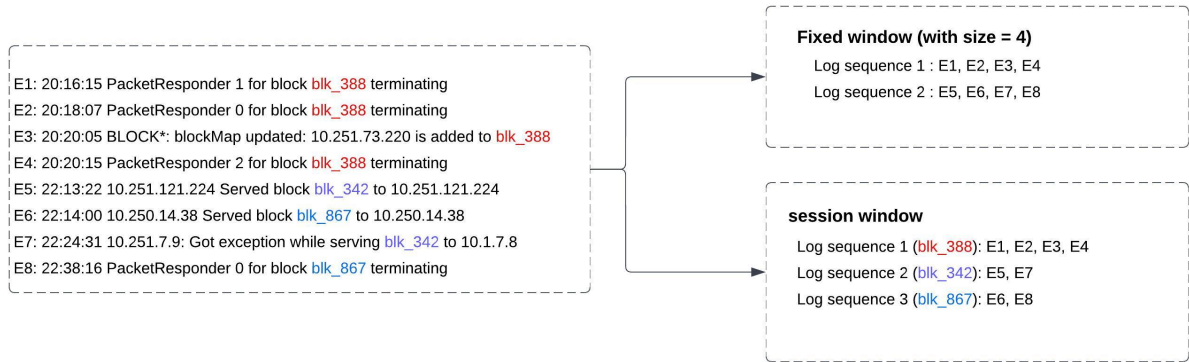


Figure 3. Log sequence grouping using fixed window and session window methods.

Semantic Vector Representation

To capture the contextual semantics of log templates, each structured message is converted into an embedding using the BERT-Base, Cased model [16]. BERT [15] is a bidirectional Transformer-based language model pre-trained on large-scale corpora such as BooksCorpus and English Wikipedia, designed to learn deep contextual relationships between words and subwords. In this work, each tokenized log template is fed into BERT, which generates a dense, fixed-size vector representation of 768 dimensions from its final encoder layer. These embeddings encode both syntactic and semantic information, enabling templates with similar meanings but different surface forms to be positioned closely in the latent space. The use of BERT therefore provides a richer and more discriminative semantic representation of log events compared with traditional word-level embeddings such as Word2Vec [17]. Context-aware Transformer representations have also shown effectiveness in security-oriented detection tasks, where modeling contextual relationships can improve the identification of complex or obfuscated patterns [35].

In this work, BERT embeddings are precomputed offline and stored prior to training and inference. The effect of replacing conventional Word2Vec representations with BERT-based contextual embeddings was investigated in our previous work [31], where the semantic representation stage of the LightLog framework was specifically evaluated. The results of that study provide the empirical motivation for adopting BERT as the semantic encoder in the present framework. Since the objective of the present work is not to benchmark general-purpose embedding models, we retain the previously validated BERT representation and focus the experimental analysis on the new nonlinear PCA–tSNE dimensionality-reduction and enhanced lightweight TCN components introduced here. This design avoids the computational overhead associated with online embedding generation and contributes to the overall efficiency of the proposed approach. Accordingly, BERT acts as a fixed offline semantic encoder and is not jointly optimized with the lightweight TCN classifier.

Dimensionality Reduction

Dimensionality reduction is an important part of the preprocessing phase of data processing, in which an attempt is made to simplify complex high-dimensional representations while retaining the most informative features. This phase was first introduced in the LightLog framework in [10], in which the PCA-PPA method was used to improve feature orthogonality by reducing redundancy. This transformation not only minimizes the complexity of the embedding space but also decreases storage space as well as computational complexity, making it easier to deploy such systems even in resource-constrained devices. Related security-oriented studies similarly show that compact feature representations can substantially reduce dimensionality and inference cost while retaining discriminative information for detection [36].

For dimensionality reduction in this study, we adopt a PCA-tSNE pipeline, in which PCA is followed by the established nonlinear t-SNE method [18]. In this method, first, PCA is used to reduce noise in the data while retaining principal components. Then, t-SNE maps the data onto a low-dimensional manifold that preserves local neighborhood structures. Unlike the linear PCA-PPA transformation, PCA-tSNE introduces a nonlinear neighborhood-preserving stage after PCA. This choice is motivated by the contextual BERT representation, where semantically related log templates may exhibit local similarity relationships that are not necessarily preserved by a purely linear projection. By modeling pairwise neighborhood similarities, t-SNE provides a mechanism for retaining such local relationships in the compact representation. We therefore consider PCA-tSNE as a motivated nonlinear alternative rather than assuming a theoretical guarantee of superior performance, and evaluate its effectiveness empirically on both HDFS and BGL.

3.2. Proposed Enhanced TCN Architecture

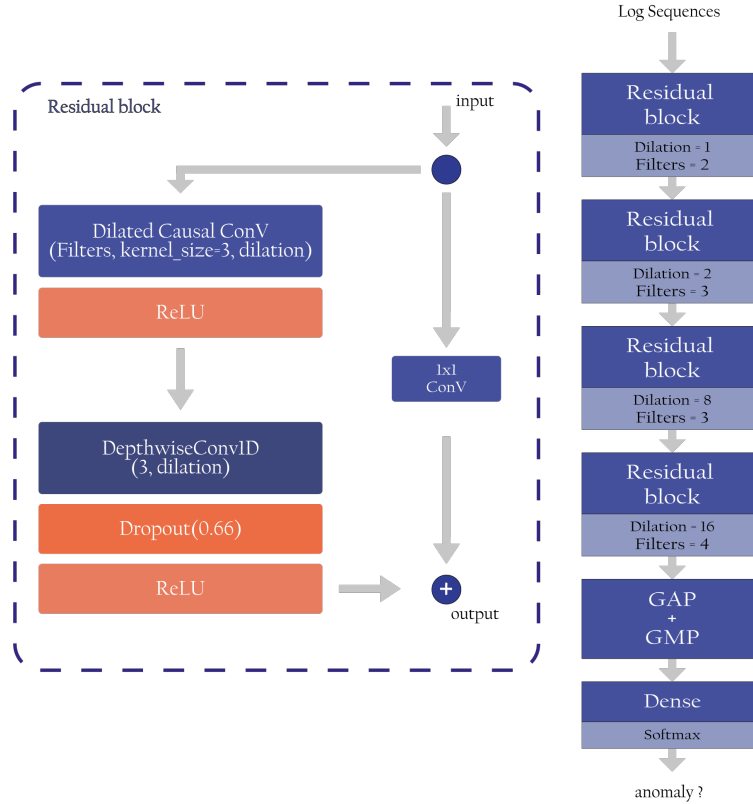


Figure 4. Proposed TCN architecture for log sequence classification.

Figure 4 illustrates an overview of the architecture of the proposed LS-TCN model. The main components of the LS-TCN architecture are four residual blocks, followed by the Global Average Pooling (GAP) layer [20] and the Global Max Pooling (GMP) layer. The feature representation obtained from the data is then passed through the dense layer to predict the class of the log sequences, i.e., normal or anomalous. GAP and GMP are used to provide complementary summaries of the temporal feature maps. GAP summarizes the overall activation pattern across the temporal dimension, whereas GMP retains the strongest localized responses. Their joint use therefore allows the final classifier to exploit both distributed sequence-level information and highly activated features that may correspond to salient log events.

The architecture of the residual blocks in the proposed TCN architecture is based on the dilated causal convolution operation, followed by the ReLU activation function, the Depthwise Conv1D operation, dropout, and the ReLU activation function. Additionally, the residual connection connects the input tensor to the output of the residual blocks, which stabilizes the gradients and prevents the degradation of the model when the depth of the network increases. To progressively expand the scope of the network, the dilation rates are progressively increased in the residual blocks, while the number of parameters in the network is not increased, which enables the network to learn the short-term as well as the long-term patterns in the input data. In addition, the causal convolution operation ensures the chronological order of the input data, which is required in the case of the anomaly detection task.

In depthwise convolution, different convolutional filters are used for different channels, and this significantly reduces the computational cost and the number of parameters compared to standard convolutional operations [21]. This choice has been made to improve efficiency and accuracy in terms of model performance. This is because, by doing so, the model becomes simpler and its training time is reduced, leading to a lightweight and fast model with improved accuracy in recognizing patterns from sequential data.

4. Experimental Results and Analysis

4.1. Datasets

To validate our LS-TCN approach, using the two known HDFS and BGL datasets, which are commonly used to evaluate methods of log anomaly detection. These two used datasets are different in structure, in scale, and variability, and offering a well rounded evaluation of our approach across different conditions. The main characteristics of them are summarized in Table 2.

Dataset	#Logs	#Anomalies	Grouping	#Sequences	Train seq	Test seq
HDFS	11,175,629	16,838 (blocks)	Session	575,061	12,000 (6,000 anomalous)	500,000 (10,000 anomalous)
BGL	4,747,963	348,460 (logs)	Fixed	94,931	18,985 (1,716 anomalous)	75,946 (7,080 anomalous)

Table 2. Summary of the datasets used in our experiments.

The used HDFS dataset is based on logs collected from 200 node Hadoop Distributed File System cluster, which is widely used for large scale distributed applications [22]. It comprises more than 11 million log messages, each identified by a block ID representing all these log messages related to one file operation. We used session window method to aggregate all these log messages into 575 061 sequences, and each marked as normal or anomalous, based on error reports from system. To train our model we used 12,000 sequences, including 6,000 anomalous while 500,000 sequences including 10,000 anomalous used for our testing. This dataset contains logs with regular and repetitive structures making it works well for evaluating approaches that are based on template extraction and sequential modeling.

BGL dataset, consists of system logs generated by the IBM Blue Gene/L supercomputer at the Lawrence Livermore National Laboratory (LLNL) [23]. It covers about 4.7 M log entries with many identified as anomalies based on recorded failure events. In our study, we used the fixed window strategy to group the logs resulting in 94,931

sequences, with 18, 985 used for training (1, 716 of them are anomalous) and 75, 946 for testing (7, 080 of them are anomalous).

4.2. Reproducibility and Experimental Setup

All experiments are conducted over five independent runs using different random seeds and deterministic configurations to ensure reproducibility. The same preprocessing pipeline, the model architecture, and the hyperparameters are applied across all experiments. The reported results correspond to the average performance over the five independent runs.

Table 3. Reproducibility and Experimental Configuration

Parameter	Value
Number of Runs	5
Random Seeds	42, 123, 2024, 3407, 777
Sequence Length	50
Embedding Dimension	17
Batch Size	64
Epochs	80
Optimizer	Adam
Learning rate	0.01
Loss Function	Binary Cross-Entropy

The experiments were carried out on a machine equipped with an NVIDIA Tesla T4 GPU (16 GB VRAM), approximately 16 GB of RAM, and standard CPU resources. The use of offline precomputed embeddings further reduces runtime requirements during both training and inference.

4.3. Effect of Dimensionality Reduction Methods

For this section, we study the impact of different dimensionality reduction strategies on the performance and efficiency of this proposed LS-TCN. This linear PCA-PPA technique used in the LightLog approach [10] is compared with a non linear PCA-tSNE method we propose in this paper. We use the HDFS and BGL datasets, we evaluate both detection performance and the associated computational cost and runtime. To ensure a fair comparison, both approaches are evaluated under the same experimental protocol and TCN architecture, while retaining their respective dimensionality-reduction configurations.

Table 4 provides a comparative evaluation of PCA-PPA and PCA-tSNE in terms of precision, recall, and F1-score on the HDFS and BGL datasets. On HDFS, PCA-tSNE improves all three evaluation metrics, increasing the F1-score from 0.956 to 0.967. On BGL, PCA-tSNE achieves a higher recall and F1-score, reaching 0.963 and 0.954, respectively, while maintaining a comparable precision to PCA-PPA. These results indicate that the nonlinear PCA-tSNE representation provides more favorable detection performance across both datasets under the evaluated configuration.

The effects of dimensionality reduction on model complexity are measured in terms of FLOPs and parameters and are presented in Figure 5. Comparing PCA-PPA with PCA-tSNE, a reduction of approximately 7.8% in FLOPs and parameters is observed under the evaluated configurations. This reduction is a consequence of the more compact input representation used by the downstream TCN, rather than an inherent computational advantage of t-SNE itself. Since PCA-tSNE is performed offline, the reported complexity reduction reflects the resulting TCN configuration during inference.

Table 4. Comparison of reduction methods on HDFS and BGL datasets using precision, recall, and F1-score.

Reduction Method	HDFS			BGL		
	Precision	Recall	F1-score	Precision	Recall	F1-score
PCA-PPA	0.927	0.985	0.956	0.945	0.938	0.942
PCA-tSNE	0.937	0.998	0.967	0.944	0.963	0.954

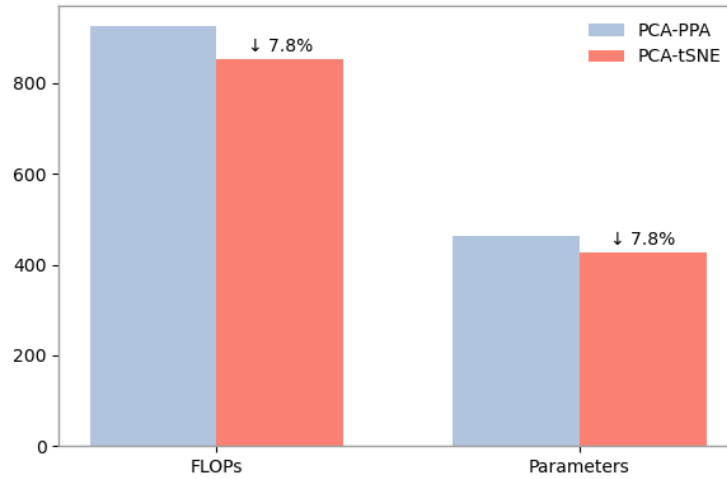


Figure 5. Impact of Dimensionality Reduction on Model Complexity

Runtime efficiency is further analyzed across both datasets, with detailed findings presented in Table 5. T_{infer} denotes the time required by the detection model to classify one log sequence, whereas T_{detect} represents the overall online processing time per sequence, including sequence preparation, retrieval of the in-memory precomputed representation, and model inference. PCA-tSNE consistently exhibits slightly lower inference and detection times than PCA-PPA, while also reducing the overall testing time. The gain is more evident for the HDFS dataset, suggesting better scalability when processing large volumes of logs, whereas runtime differences on BGL remain marginal but consistently favor PCA-tSNE. These findings indicate that the nonlinear dimensionality reduction does not increase inference-time overhead. In the proposed LS-TCN, PCA-t-SNE is applied offline to the template-level semantic representations to construct a fixed low-dimensional lookup table. During inference, log sequences composed of known event identifiers are encoded by retrieving their corresponding precomputed low-dimensional vectors. If an event identifier is not present in the precomputed representation table, the current implementation uses a zero-vector fallback, allowing the sequence to be processed without recomputing PCA-tSNE. However, this fallback does not preserve semantic information for previously unseen templates. Therefore, environments in which the log-template vocabulary evolves substantially may require periodic offline regeneration of the semantic representation table and corresponding updates to the downstream classifier. Developing an explicit out-of-sample mapping mechanism for newly observed templates is left for future work.

The inference pipeline consists of three stages: (1) semantic embedding retrieval, (2) retrieval of precomputed low-dimensional representations obtained using PCA-tSNE, and (3) classification using the TCN model. The dimensionality reduction step is performed offline during preprocessing and does not affect real-time inference.

In terms of complexity, PCA has linear complexity, while t-SNE is more computationally expensive due to pairwise similarity computations. However, since PCA-tSNE is applied offline, inference efficiency is primarily governed by the lightweight TCN architecture. This design therefore introduces additional one-time preprocessing

Table 5. Runtime Comparison of Dimensionality Reduction Methods on HDFS and BGL.

Method	HDFS			BGL		
	T_{infer} (ms)	T_{detect} (ms)	T_{test} (s)	T_{infer} (ms)	T_{detect} (ms)	T_{test} (s)
PCA-PPA	0.067	0.264	71.46	0.080	0.153	11.219
PCA-tSNE	0.065	0.241	60.45	0.078	0.152	11.105

cost in exchange for reducing the computational burden of repeated inference, representing a deliberate trade-off between offline setup complexity and online efficiency.

4.4. Comparison with Baseline Methods

To further evaluate the performance of LS-TCN, we compare it against several representative log anomaly detection techniques, including LogRobust, DeepLog, LightLog, HitAnomaly, and LogAnomaly. Together, these methods correspond to recurrent, robust statistical and lightweight semantic-based approaches, respectively. The evaluation is carried out on the two datasets HDFS and BGL using a consistent experimental protocol and standard metrics, that is precision, recall, and F1-score, ensuring a fair comparison across methods.

Table 6. Performance Comparison with Baseline Methods on HDFS and BGL

Dataset Metric	LS-TCN	LightLog	LogRobust	DeepLog	LogAnomaly	HitAnomaly
HDFS	Precision	0.937 ± 0.021	0.830 ± 0.0733	0.958 ± 0.0336	0.896 ± 0.0358	0.950 ± 0.0102
	Recall	0.998 ± 0.0006	0.993 ± 0.0043	0.998 ± 0.0003	0.9654 ± 0.0302	0.942 ± 0.0145
	F1-score	0.967 ± 0.011	0.903 ± 0.0453	0.978 ± 0.0210	0.9292 ± 0.0237	0.946 ± 0.0124
BGL	Precision	0.944 ± 0.013	0.965 ± 0.0169	0.942 ± 0.0048	0.229 ± 0.015	0.868 ± 0.0116
	Recall	0.963 ± 0.0093	0.908 ± 0.0207	0.961 ± 0.006	0.964 ± 0.003	0.902 ± 0.0141
	F1-score	0.954 ± 0.01	0.935 ± 0.0139	0.951 ± 0.004	0.370 ± 0.02	0.885 ± 0.0128

The detection performance comparison is reported in Table 6. On HDFS, LS-TCN achieves a recall of 0.998 ± 0.0006 and an F1-score of 0.967 ± 0.011 . Although its precision is lower than that of LogRobust and HitAnomaly, LS-TCN maintains a high recall with very low variation across the five runs, indicating consistent anomaly detection. LogRobust obtains the highest mean F1-score on HDFS (0.978 ± 0.021). On BGL, LS-TCN achieves the highest mean F1-score (0.954 ± 0.010), with a precision of 0.944 ± 0.013 and a recall of 0.963 ± 0.0093 . It also shows relatively small variations across the five runs. Overall, the results demonstrate that LS-TCN provides competitive and consistent detection performance across both datasets. From an operational perspective, this high recall is particularly important because false negatives correspond to missed anomalies, which may carry greater operational risk than additional false alarms.

Table 7. Controlled comparison of LightLog and LS-TCN architectures using the same BERT-PCA-tSNE representation.

Architecture	HDFS			BGL			Complexity	
	Precision	Recall	F1-score	Precision	Recall	F1-score	Params	FLOPs
LightLog-TCN	0.846	0.996	0.915	0.966	0.914	0.940	450	900
LS-TCN	0.937	0.998	0.967	0.944	0.963	0.954	426	852

To further isolate the contribution of the proposed TCN architecture from that of the semantic representation, we conduct a controlled comparison with the LightLog TCN architecture using the same BERT-PCA-tSNE input representation. As shown in Table 7, LS-TCN improves the F1-score from 0.915 to 0.967 on HDFS and from 0.940 to 0.954 on BGL, while using slightly fewer parameters and FLOPs. Since both models use the same input representation, these results indicate that the performance gains are not solely attributable to BERT-based semantic encoding, but also to the architectural modifications introduced in LS-TCN.

Table 8. Comparison of Model Complexity and Total Testing Time on HDFS and BGL

Methods	FLOPs	Parameters	Model size (KB)	T_{test} (s)	
				HDFS	BGL
DeepLog	53,248	52,252	618	287.62	85.7
RobustLog	354,432	352,385	4,138	882.0	109.0
LogAnomaly	106,496	104,476	1239	51.600	5.778
LightLog	1408	504	139	177.66	41.14
LS-TCN	852	426	116.75	60.45	11.105

Beyond detection accuracy, this Table 8 here compares the computational efficiency of this evaluated methods based on the number of parameters, the FLOPs, the model size, and testing time. The proposed approach achieves the lowest computational cost across all metrics, with fewer FLOPs and parameters than LightLog while also outperforming DeepLog and LogRobust in terms of scalability. particularly, LS-TCN reduces the total testing time to 60.45s on HDFS and 11.105s on BGL, representing clear speedups over all baseline methods. These results show the suitability of LS-TCN for deployment in resource-constrained environments, where both detection performance and computational efficiency are critical.

5. Conclusion

In this work, we address the challenge of log anomaly identification within large-scale and resource-constrained environments, where achieving accurate detection while maintaining a low computational cost remains a major challenge. We introduce a lightweight and context-aware solution that combines semantic log representations, a nonlinear dimensionality reduction strategy based on PCA-tSNE, and an enhanced Temporal Convolutional Network model for efficient temporal modeling. This design helps enable the capture of both semantic and temporal patterns in log sequences while preserving computational efficiency. Experimental evaluations on HDFS and BGL show that our method achieves strong and robust detection performance compared to existing methods, while also improving computational efficiency. In particular, this approach significantly reduces model complexity in terms of parameters and FLOPs compared to existing lightweight baselines, while decreasing total testing time by more than 60% on large-scale datasets such as HDFS. These gains are achieved without sacrificing detection effectiveness, showing that our method offers a good balance between accuracy and efficiency, and is well suited for real-world log monitoring systems operating under resource constraints. Future directions will focus on extending the method to online and streaming log analysis, as well as exploring adaptive semantic representations to further improve robustness across different system environments.

REFERENCES

1. S. He, J. Zhu, P. He, and M. R. Lyu, *Experience Report: System Log Analysis for Anomaly Detection*, In Proceedings of the IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 207–218, 2016.

2. J.-G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, *Mining invariants from console logs for system problem detection*, In Proceedings of the USENIX Annual Technical Conference (USENIX ATC), 2010.
3. D. Wu, Z. Jiang, X. Xie, X. Wei, W. Yu, and R. Li, *LSTM Learning With Bayesian and Gaussian Processing for Anomaly Detection in Industrial IoT*, IEEE Transactions on Industrial Informatics, vol. 16, no. 8, pp. 5244–5253, 2020.
4. Q. Cheng, D. Sahoo, A. Saha, W. Yang, C. Liu, G. Woo, M. Singh, S. Savarese, and S. C. H. Hoi, *AI for IT Operations (AIOps) on Cloud Platforms: Reviews, Opportunities and Challenges*, arXiv preprint arXiv:2304.04661, 2023. DOI: 10.48550/arXiv.2304.04661. arXiv preprint arXiv:2304.04661, 2023.
5. M. Du, F. Li, G. Zheng, and V. Srikumar, *DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning*, In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS), pp. 1285–1298, 2017.
6. W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, *LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs*, In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), pp. 4739–4745, 2019.
7. K. A. Nguyen, S. Schulte im Walde, and N. T. Vu, *Integrating Distributional Lexical Contrast into Word Embeddings for Antonym-Synonym Distinction*, arXiv preprint arXiv:1605.07766, 2016. DOI: 10.48550/arXiv.1605.07766.
8. X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, and D. Zhang, *Robust log-based anomaly detection on unstable log data*, In Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 807–817, 2019.
9. S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, *HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log*, IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2064–2076, 2020.
10. Z. Wang, J. Tian, H. Fang, L. Chen, and J. Qin, *LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge*, Computer Networks, vol. 203, p. 108616, 2022.
11. V. Raunak, V. Gupta, and F. Metze, *Effective Dimensionality Reduction for Word Embeddings*, In Proceedings of the 4th Workshop on Representation Learning for NLP (Repl4NLP), 2019, pp. 235–243. DOI: 10.18653/v1/W19-4328.
12. P. He, J. Zhu, Z. Zheng, and M. R. Lyu, *Drain: An Online Log Parsing Approach with Fixed Depth Tree*, In Proceedings of the IEEE International Conference on Web Services (ICWS), pp. 33–40, 2017.
13. J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, *Tools and Benchmarks for Automated Log Parsing*, In Proceedings of the IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2019.
14. 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, 2023. DOI: 10.1016/j.mlwa.2023.100470.
15. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171–4186, 2019.
16. Google Research, *BERT pretrained models*, Available at: <https://github.com/google-research/bert>, Accessed: Dec. 30, 2024, 2021.
17. W. Ling, C. Dyer, A. W. Black, and I. Trancoso, *Two/too simple adaptations of word2vec for syntax problems*, In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1299–1304, 2015.
18. L. van der Maaten and G. Hinton, *Visualizing data using t-SNE*, Journal of Machine Learning Research, vol. 9, no. Nov, pp. 2579–2605, 2008.
19. L. Bai, Y. Zhao, and X. Huang, *A CNN Accelerator on FPGA Using Depthwise Separable Convolution*, IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 65, no. 10, pp. 1415–1419, 2018.
20. M. Lin, Q. Chen, and S. Yan, *Network In Network*, arXiv preprint arXiv:1312.4400, 2013.
21. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, arXiv preprint arXiv:1704.04861, 2017.
22. 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, pp. 117–132, 2009.
23. A. Oliner and J. Stearley, *What supercomputers say: A study of five system logs*, In Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pp. 575–584, 2007.
24. B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*, MIT Press, 2002.
25. V. Chandola, A. Banerjee, and V. Kumar, *Anomaly detection: A survey*, ACM Computing Surveys, vol. 41, no. 3, pp. 1–58, 2009.
26. F. T. Liu, K. M. Ting, and Z.-H. Zhou, *Isolation Forest*, In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422, 2008.
27. S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
28. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*, arXiv preprint arXiv:1406.1078, 2014.
29. J. L. Elman, *Finding structure in time*, Cognitive Science, vol. 14, no. 2, pp. 179–211, 1990.
30. M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. W. Chang, *A novel anomaly detection scheme based on principal component classifier*, 2003.
31. A. Zizouan, I. Hafidi, and N. Aboutabit, “Enhancing LightLog with BERT-Based Contextual Embeddings,” in *Advances in Machine Intelligence and Computer Science Applications*, Springer Nature Switzerland, pp. 307–318, 2026.
32. S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding,” arXiv preprint arXiv:1510.00149, 2015.
33. B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, “Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,”

- in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2704–2713, 2018.
34. G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” *arXiv preprint arXiv:1503.02531*, 2015.
 35. T. Almansour, M. M. Almasri, and S. A. Nagro, *Contextual Detection of Cross-Site Scripting Attacks Using RoBERTa-Based Deep Learning Model*, *Statistics, Optimization & Information Computing*, vol. 16, no. 2, pp. 1521–1537, 2026.
 36. B. Bataineh and G. Shakah, *Information-Theoretic Feature Selection via Variational Autoencoders for Network Intrusion Detection: A Mutual Information Maximization Approach*, *Statistics, Optimization & Information Computing*, vol. 16, no. 4, pp. 3308–3333, 2026.