Log-Based Anomaly Detection: Comparative Study of Real-World System Logs using Machine Learning And Deep Learning Approaches

Nadira Anjum Nipa

A Thesis

in

The Department

of

Concordia Institute for Information Systems Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Quality Systems Engineering) at

Concordia University

Montréal, Québec, Canada

July 2025

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Nadira Anjum Nipa**

Entitled: **Log-Based Anomaly Detection: Comparative Study of Real-World System Logs using Machine Learning And Deep Learning Approaches**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Quality Systems Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

______________________________ Chair & Examiner
Dr. Jeremy Clark

______________________________ Examiner
Dr. Honghao Fu

______________________________ Supervisor
Dr. Zachary Patterson

______________________________ Supervisor
Dr. Nizar Bouguila

Approved by ______________________________
Dr. Chun Wang, Chair
Department of Concordia Institute for Information Systems Engineering

____________ 2025 ______________________________
Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Log-Based Anomaly Detection: Comparative Study of Real-World System Logs using Machine Learning And Deep Learning Approaches

Nadira Anjum Nipa

The reliability and security of today's smart and autonomous systems increasingly rely on effective anomaly detection capabilities. Logs generated by intelligent devices during runtime offer valuable insights for system monitoring and troubleshooting. Nonetheless, the enormous quantity and complexity of these logs render manual anomaly inspection impractical and error-prone. To address this, various automated log-based anomaly detection methods have been developed. However, many of these approaches are evaluated in controlled environments with publicly available datasets, which differ significantly from the noisy, unstructured, and unlabeled logs encountered in industrial settings. This thesis explores and adapts existing machine learning and deep learning techniques for anomaly detection in real-world system logs produced by an intelligent autonomous display device. Initially, we conduct a comparative analysis of machine learning and deep learning methods using a small manually labeled dataset to evaluate the detection accuracy and computational efficiency. Our results highlight the most suitable approaches for enabling proactive maintenance and enhancing system reliability. Expanding on this, we evaluate advanced deep learning methods across weakly supervised, semi-supervised, and unsupervised learning paradigms, using heuristically labeled logs and benchmark them against fully supervised baselines to examine the trade-offs between label dependency, detection performance, and industrial applicability. Finally, we propose a systematic approach for managing unlabeled and noisy log data, providing practical guidelines for selecting suitable learning strategies based on label availability, data quality, and real-world constraints. The findings of this work provide valuable insights for the implementation of scalable, accurate, and robust log-based anomaly detection in industrial IoT environments.

Acknowledgments

First and foremost, I would like to express my heartfelt gratitude to my supervisors, Prof. Nizar Bouguila and Prof. Zachary Patterson, for their guidance, support, and encouragement throughout my studies. Their patience, insight, and thoughtful feedback have been a constant source of motivation and have played a crucial role in shaping this work.

I am also truly thankful to the Mitacs Accelerate Program for providing me with the opportunity to collaborate with Buspas Inc. It was a valuable experience that allowed me to apply my research in a real-world setting. I would especially like to thank Wissem Maazoun and his team for their support, cooperation, and the many helpful insights they shared along the way.

A special thanks goes to my lab colleagues, whose kindness and collaboration made the research journey more enjoyable. I have learned so much from our conversations and teamwork.

To my dear friends, especially Salim, thank you for always being there—your support, encouragement, and understanding meant more than words can say during both the good times and the more challenging ones.

I am deeply grateful to my mother, sisters, and brother for their endless love and belief in me. Your support has been the foundation of everything I have achieved.

And finally, I want to thank myself—for not giving up, for showing up every day, and for believing that I could make it through. This journey has taught me more than I could have imagined.

Contents

List of Figures

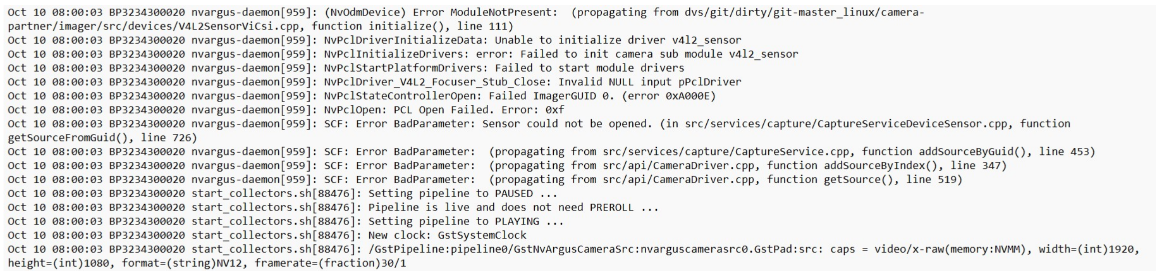
List of Tables

Chapter 1

Introduction

In the current technological environment, the Internet of Things (IoT) has become essential to various facets of everyday life, providing an extensive range of services. One example of an IoT device is the SCiNe (Smart City Network), a smart, autonomous display created by Buspas (Inc., n.d.), tailored specifically for the transportation sector. SCiNes operate autonomously, using a lithium battery and a solar panel and will eventually deliver real-time transit information at bus stops. This involves accurate bus wait times, bus occupancy information, and customer traffic insights to optimize vehicle dispatching based on demand (Inc., n.d.). For uninterrupted service and to guarantee customer satisfaction, this IoT device must operate continuously. Even small service disruptions could affect user experience, making dependable and continuous operation essential for such a large-scale and intricate system.

Anomaly detection is essential for promptly identifying unusual system behavior, which is vital for reducing system downtime and maintaining smooth operations (Epaillard & Bouguila, 2016; Fan, Bouguila, & Ziou, 2011; Sghaier, Amayri, & Bouguila, 2023). Anomaly detection involves identifying patterns, events, or observations in data that significantly differ from what is considered normal behavior. In the context of system logs, anomalies may indicate faults, failures, security threats, or other atypical conditions that require attention. Identifying these deviations is crucial to ensure system reliability, security, and operational efficiency. In industrial IoT settings, prompt detection of anomalies can prevent expensive downtime and interruptions in service. System logs



```
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: (NvOdmDevice) Error ModuleNotPresent:  (propagating from dvs/git/dirty/git-master_linux/camera-
partner/imager/src/devices/V4L2SensorViCsi.cpp, function initialize(), line 111)
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: NvPclDriverInitializeData: Unable to initialize driver v4l2_sensor
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: NvPclInitializeDrivers: error: Failed to init camera sub module v4l2_sensor
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: NvPclStartPlatformDrivers: Failed to start module drivers
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: NvPclDriver_V4L2_Focuser_Stub_Close: Invalid NULL input pPclDriver
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: NvPclStateControllerOpen: Failed ImagerGUID 0. (error 0xA000E)
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: NvPclOpen: PCL Open Failed. Error: 0xf
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: SCF: Error BadParameter: Sensor could not be opened. (in src/services/capture/CaptureServiceDeviceSensor.cpp, function
getSourceFromGuid(), line 726)
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: SCF: Error BadParameter:  (propagating from src/services/capture/CaptureService.cpp, function addSourceByGuid(), line 453)
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: SCF: Error BadParameter:  (propagating from src/api/CameraDriver.cpp, function addSourceByIndex(), line 347)
Oct 10 08:00:03 BP3234300020 nvargus-daemon[959]: SCF: Error BadParameter:  (propagating from src/api/CameraDriver.cpp, function getSource(), line 519)
Oct 10 08:00:03 BP3234300020 start_collectors.sh[88476]: Setting pipeline to PAUSED ...
Oct 10 08:00:03 BP3234300020 start_collectors.sh[88476]: Pipeline is live and does not need PREROLL ...
Oct 10 08:00:03 BP3234300020 start_collectors.sh[88476]: Setting pipeline to PLAYING ...
Oct 10 08:00:03 BP3234300020 start_collectors.sh[88476]: New clock: GstSystemClock
Oct 10 08:00:03 BP3234300020 start_collectors.sh[88476]: /GstPipeline:pipeline0/GstNvArgusCameraSrc:nvarguscamerasrc0.GstPad:src: caps = video/x-raw(memory:NVMM), width=(int)1920,
height=(int)1080, format=(string)NV12, framerate=(fraction)30/1
```

Figure 1.1: Example of system logs.

serve as one of the most valuable sources of data for detecting anomalies, as they document real-time events and activities that occur within a system. Generally, each log entry includes a timestamp, device identifier, log source, and a descriptive message. The datasets used in this work reflect the diverse and intricate characteristics of industrial system logs. For example, SCiNe device logs that record routine start and data updates are considered normal, while repeated 'Module Not Found' errors within seconds represent a clear deviation from expected behavior, potentially indicating a fault. As illustrated in Figure 1.1, logs from a SCiNe device are often flagged as anomalous when they contain terms such as "error" or "failed". Although such anomalies can sometimes be triggered by benign conditions, they frequently signal system malfunctions or performance degradation, positioning log-based anomaly detection as a significant field of study.

Historically, anomaly detection in logs has been based on manual inspection. However, the vast quantity and complexity of log events produced each second in contemporary systems make manual analysis impractical, prompting the development of automated log-analysis methods.

Many statistical and traditional machine learning algorithms, such as Decision Tree (M. Chen, Zheng, Lloyd, Jordan, & Brewer, 2004), Principal Component Analysis (Tra, Amayri, & Bouguila, 2024; Xu, Huang, Fox, Patterson, & Jordan, 2009), and Log Clustering (Lin, Zhang, Lou, Zhang, & Chen, 2016; Pawar, Zamzami, & Bouguila, 2020), have been used to automate the identification of significant incidents or anomalies in log data. Although these conventional methods have made notable contributions, they are hampered by drawbacks such as limited interpretability, inflexibility, and the requirement for manual feature engineering (Le & Zhang, 2022), (X. Zhang et al., 2019).

To address these challenges, deep learning techniques such as DeepLog (M. Du, Li, Zheng, &

Srikumar, 2017), LogRobust (X. Zhang et al., 2019) have been developed, demonstrating encouraging outcomes.

Although advances have been achieved in the literature, a significant gap remains in the use of these techniques for industrial datasets. The majority of research has focused on public datasets that come with predefined conditions, where data is already labeled, organized, and structured effectively. Conversely, real-world data introduces further difficulties, including noise, variability, and unstructured formats, which significantly complicate log-based anomaly detection in industrial systems. The following are some of the significant challenges faced in the application of real-world log-based anomaly detection.

- The absence of centralized aggregation mechanisms requires manual log collection from multiple sources, leading to significant time consumption and a higher risk of human error. The tedious and ineffective manual approach of collecting and identifying logs hinders data preparation.
- The structure of logs in industrial systems is highly heterogeneous and varies widely among systems, applications, and components, unlike public databases. The log file may include messages that have varying structures, complicating the application of standard parsing or analysis methods.
- The quality of industrial log data is often compromised by noise, redundancy, and extraneous information. Diverse applications generate logs at a rapid pace, resulting in substantial amounts of repetitive or irrelevant data. The lack of uniform logging standards across systems adds unnecessary tokens and metadata, making analysis more complex and hindering efficient anomaly detection.

Compounding these data challenges are the inherent difficulties associated with different learning paradigms. Log anomaly detection generally adheres to one of three learning paradigms based on the availability of labeled data: supervised, semi-supervised, or unsupervised learning, each of which encounters substantial challenges in industrial settings.

- Supervised learning models attain impressive accuracy when trained on accurately labeled normal and anomalous logs. Nevertheless, manual labeling incurs high costs, requires significant time, and is susceptible to errors, frequently resulting in mislabeled data where normal logs are misidentified as anomalies and the opposite occurs. Moreover, the disparity between normal and anomalous logs has a significant impact on model performance.

- Semi-supervised learning models rely on an extensive collection of labeled normal logs to understand typical behavior patterns. However, incorrectly labeled logs can greatly impair performance. Moreover, industrial logs exhibit a high degree of dynamism—log formats often evolve, leading to the emergence of unfamiliar log patterns that semi-supervised methods find challenging to capture.

- Unsupervised models don't need labeled data, but they often make more mistakes than correct ones because they don't know what to look for in abnormalities. These models face challenges with log instability, as frequent changes in log formats hinder the effectiveness of anomaly detection.

- To address these limitations, weakly supervised learning has emerged as a compelling alternative. Weakly supervised models use noisy, heuristically labeled data with minimal supervision, avoiding the need for high-quality manual annotations. Instead of relying on fully annotated ground truth, these models use domain expertise, predefined rules, and approximations for classifying data. They combine a small set of labeled abnormal logs with a large volume of unlabeled logs, improving generalization and making the models more resilient to label noise and contamination.

Despite significant progress in log anomaly detection, industrial logs vary substantially from publicly available datasets with respect to their structure, noise levels, and operational complexity. Furthermore, earlier research has not thoroughly examined the various learning paradigms in industrial settings, especially concerning heuristically labeled logs.

This thesis focuses on tackling the significant challenges associated with implementing log-based anomaly detection in practical industrial settings. Initially, we explore the difficulties presented by industrial system logs, including noise, redundancy, and inconsistent structure, by using various supervised and unsupervised machine learning (ML) and deep learning (DL) models on a small, expertly labeled dataset. This initial study offers important information on model performance under clean, yet limited labeling conditions, helping to identify the practical limitations and potential of traditional methods in real-world deployment scenarios. Next, we broaden our research to a larger, more realistic industrial dataset, where labels have been assigned heuristically. In this study, we evaluate the performance of advanced deep learning models within the weakly supervised, semi-supervised, and unsupervised learning approaches. These models are evaluated against fully supervised baselines to examine trade-offs in accuracy, label dependency, robustness, and scalability in noisy, partially labeled conditions. The findings of both studies provide valuable recommendations for choosing suitable learning paradigms and implementing anomaly detection systems in industrial IoT settings. This work ultimately aids in the creation of dependable and scalable solutions for proactive system monitoring and maintenance.

1.1 Contributions

The key contributions of this thesis are follows:

- **A Comparative Study of Log-Based Anomaly Detection Methods in Real-World System Logs:**

  We implemented various established ML and DL log anomaly detection methods on a real-world industrial dataset to evaluate their performance under different experimental conditions. This detailed analysis provides practical insights into the strengths and limitations of models, as well as strategies for employing log-based anomaly detection in industrial environments. This research was presented and published at the 10th$^{\text{th}}$ International Conference on Internet of Things, Big Data and Security (IoTBDS 2025) (Nipa, Bouguila, & Patterson, 2025a).

- **Log-Based Anomaly Detection Without Ground-Truth: Evaluating Weakly Supervised, Semi-Supervised, and Unsupervised Deep Learning Approaches:**

  We conducted an in-depth evaluation of deep learning models trained within weakly supervised, semi-supervised, and unsupervised paradigms utilizing heuristic-labeled industrial logs. These were evaluated against supervised baselines to investigate the trade-offs in detection performance, supervision levels, and computational efficiency. The analysis emphasizes the robustness of the model in the presence of noisy labels and offers guidelines for selecting models that are feasible and scalable in an industrial context. This work was accepted and presented at the 34th IEEE International Symposium on Industrial Electronics (ISIE 2025) (Nipa, Bouguila, & Patterson, 2025b).

- **Improving Transit System Reliability through Log-Based Anomaly Detection in a Smart Autonomous Device:**

  Expanding on the insights from the two conference papers, we integrated and enhanced the analysis to tackle wider issues concerning the reliability of the smart transit device. This study combines comparative assessment of machine learning and deep learning models at various supervision levels, offering practical suggestions for implementation in smart transportation settings. The study emphasizes how log-based anomaly detection can be leveraged to enhance operational reliability using existing IoT infrastructure, without incurring high costs for manual monitoring or large-scale log annotation. This paper has been submitted to the Transportation Research Board (TRB) Annual Meeting 2026.

1.2 Thesis Organization

The remainder of the thesis is structured as follows:

- Chapter 2 offers a comprehensive overview of the existing literature on log-based anomaly detection.
- Chapter 3 introduces the comparative study of log-based anomaly detection methods in real-world system logs.
- Chapter 4 presents log-based anomaly detection without ground truth: evaluation of weakly supervised, semi-supervised, and unsupervised deep learning approaches.
- Chapter 5 wraps up the thesis and presents some future directions.

Chapter 2

Literature Review

Log-based anomaly detection has received significant interest as an essential method to detect unusual behavior in intricate systems, including distributed applications, cloud environments, and industrial IoT devices. As systems produce logs in significant quantities and at a rapid pace, conventional rule-based or manual inspection techniques become unfeasible. As a result, numerous automated techniques have been developed, ranging from traditional machine learning (ML) to sophisticated deep learning (DL) to efficiently identify anomalies in system logs. These techniques include various learning paradigms, such as supervised, semi-supervised, unsupervised, and weakly supervised learning. Although numerous methods demonstrate encouraging results on benchmark datasets like Hadoop Distributed File System (HDFS) and BlueGene/L (BGL), their performance is frequently affected when used on noisy, unstructured, and partially labeled real-world industrial logs. This chapter describes the current state of the scientific literature relative to log-based anomaly detection, organized by methodological approach and learning paradigm.

2.1 Traditional Machine Learning-Based Approaches

Supervised machine learning techniques require labeled log data to train classification models capable of identifying anomalies. Techniques of supervised learning have frequently been used in the detection of log anomalies. For instance, Decision Trees (DT) (M. Chen et al., 2004) have been used to pinpoint anomalies in extensive internet-based settings, while Support Vector Machine

(SVM) (Liang, Zhang, Xiong, & Sahoo, 2007) classifiers have been employed to uncover failures within event logs. Regression-based methods have been investigated for analyzing cloud resource logs and identifying abnormalities (S. He et al., 2018). However, in real-world industrial contexts, acquiring labeled log data is often time-consuming, expensive, and error-prone, limiting the practical scalability of supervised ML methods.

Unsupervised ML methods address the challenge of label dependency by detecting anomalies through statistical properties or structural deviations within the data. Methods such as Isolation Forest (Liu, Ting, & Zhou, 2008) effectively isolate outliers without the need for prior labeling, whereas Principal Component Analysis (PCA) (Xu et al., 2009) minimizes dimensionality to reveal anomalies in log patterns. Invariant Mining (Lou, Fu, Yang, Xu, & Li, 2010) identifies linear correlations among event counts, marking violations as anomalies. Control flow graphs (Nandi, Mandal, Atreja, Dasgupta, & Bhattacharya, 2016) represent standard execution paths and identify any deviations. While adaptable, these methods might face challenges with log instability and may not provide the contextual depth required to analyze complex sequences.

2.2 Deep Learning-Based Approaches

The rise of deep learning has brought forth robust models that can learn intricate, high-dimensional representations from unprocessed log sequences. These models have been examined through various learning paradigms to find a balance between accuracy and label dependency.

Supervised methods using labeled logs achieve better results, but they require a significant amount of labeled data for training classification models. LogRobust (X. Zhang et al., 2019) uses an attention-based Bi-LSTM approach for log-based anomaly detection that can address the instability in log events. LogCNN (Lu, Wei, Li, & Wang, 2018) applies convolutional neural networks (CNNs) to learn log patterns, significantly enhancing the detection of anomalous events. LogTransfer (R. Chen et al., 2020), a transfer learning framework was proposed that improves log anomaly detection by minimizing the impact of noise in anomalous log sequences. The transformer architecture first applied to log-based anomaly detection via a method called HitAnomaly (Huang et al., 2020) uses a hierarchical transformer to represent both log events and their related parameters.

LogAttention (Q. Du, Zhao, Xu, Han, & Zhang, 2021) improves semantic representations by incorporating log patterns into semantic vectors and employing a self-attention neural network for anomaly detection.

Semi-supervised methods, which learn from normal log patterns, lessen the dependence on labeled data but frequently face challenges with high false positive rates. PLELog (Yang et al., 2021) leverages pseudo-labeling to improve the detection of anomalies in log sequences while reducing the need for large labeled datasets. LogBERT (Guo, Yuan, & Wu, 2021) employs the BERT architecture along with two self-supervised tasks to proficiently capture the structure of normal log sequences in the absence of labeled anomalies. An LSTM-based approach (Baril, Coustié, Mothe, & Teste, 2020) addresses temporal deviations by modeling normal behavior and detecting anomalies through the identification of deviations from established temporal patterns.

Unsupervised methods remove the necessity for labeled logs, greatly decreasing manual effort and annotation time; however, they may result in lower detection accuracy. DeepLog (M. Du et al., 2017) and LogAnomaly (Meng et al., 2019) are two major unsupervised deep learning models that learn normal log behaviors and detect anomalies based on deviations from these behaviors. Both models leverage recurrent neural networks (RNNs) to learn sequential patterns in log data and identify abnormal sequences effectively. LogAttn (L. Zhang et al., 2021) integrates an auto-encoder with an attention mechanism to reconstruct log sequences and identify anomalies when reconstruction errors exceed a certain threshold. A framework for transferable log anomaly detection, Log-TAD, was introduced in (X. Han & Yuan, 2021) which employs adversarial domain adaptation to facilitate cross-system anomaly detection with minimal labeled data.

Weakly supervised methods learn from noisy, partial, or inaccurately labeled logs, making them suitable for real-world industrial environments where high-quality labeling is limited or expensive. SpikeLog (Qi et al., 2023) uses spike events in log data to identify anomalies, combining a small amount of labeled abnormal logs with a large pool of unlabeled logs to improve generalization. LogLG (Guo et al., 2023) identifies log anomalies by creating a log-event graph, without requiring log parsing or labeled normal logs. A novel labeling strategy was proposed in (M. He et al., 2025), where system experts can label a collection of log entries within a specific time frame instead of labeling each individual log entry and introduced MIDLog approach to detect anomalies.

2.3 Dataset Usage and Real-World Challenges

A significant portion of the current literature evaluates models through the use of public datasets, including HDFS, BGL, and Thunderbird. Although valuable for benchmarking purposes, these datasets vary greatly from actual industrial logs, which tend to be unstructured, noisy, and inconsistent in formatting. Moreover, heuristic labeling is prevalent in industrial settings, which adds to the uncertainty. The identified discrepancies underscore the importance of assessing anomaly detection methods within the framework of realistic industrial constraints—an element thoroughly examined in this thesis.

2.4 Summary and Research Gap

In conclusion, log-based anomaly detection has progressed through various methodologies, each presenting trade-offs regarding performance, label dependency, and resilience to noise. Supervised methods demonstrate effectiveness; however, they necessitate a considerable amount of labeled data. Semi-supervised and unsupervised methods lessen the requirement for annotations, yet they might affect accuracy. Weakly supervised models present an intriguing opportunity for industrial settings, yet they have still not been fully analyzed. Moreover, a significant portion of previous research emphasizes clean public datasets, which restricts their relevance in practical scenarios. This thesis tackles these gaps by performing a comparative evaluation of machine learning and deep learning models across all significant learning paradigms on industrial datasets, providing practical guidance for deployment in intricate IoT environments.

Chapter 3

A Comparative Study of Log-Based Anomaly Detection Methods in Real-World System Logs

As previously discussed, existing log-based anomaly detection methods are generally developed and evaluated in controlled settings with clean, properly labeled public datasets. However, these methods face major challenges when applied with real-world industrial data, which are often noisy, unstructured, and sparsely labeled. This study presents a thorough comparative analysis of established machine learning and deep learning techniques applied to a real-world industrial log dataset to tackle these challenges. In this chapter, we describe the common framework of log-based anomaly detection and existing methods, outline the experimental setup, and present the evaluation results of various anomaly detection models. Additionally, we present a comparative analysis featuring results derived from a widely-used public dataset, emphasizing the distinctions in model behavior between controlled and real-world conditions. Finally, we evaluate the computational efficiency of each method, providing insights into their practical applicability and performance trade-offs in industrial contexts.

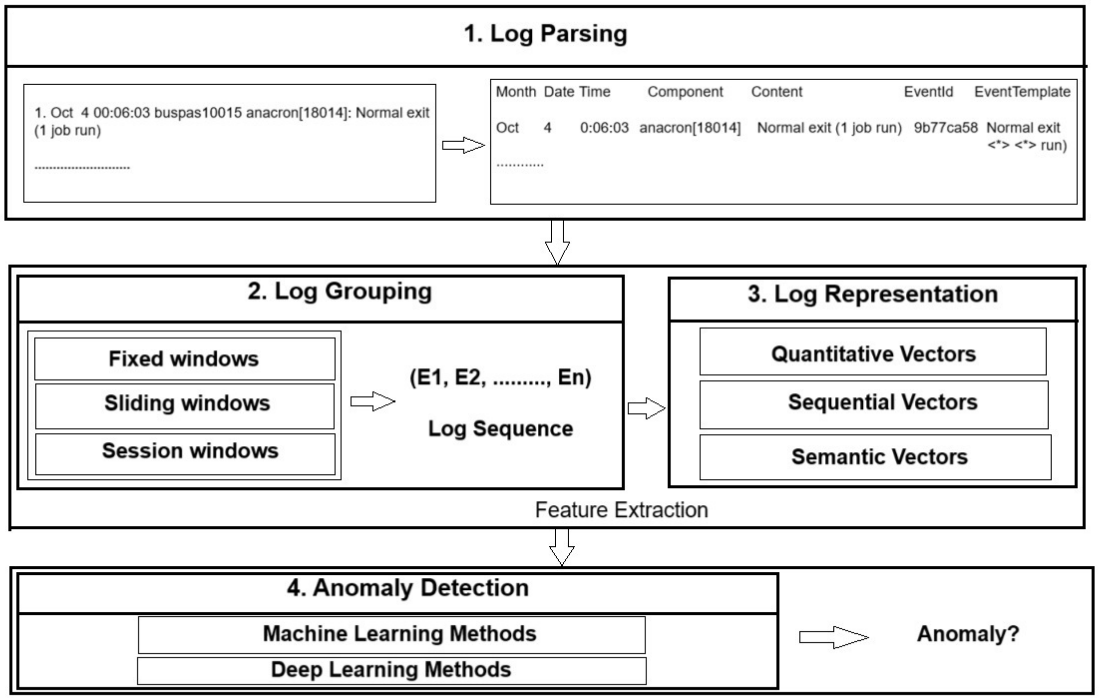


Figure 3.1: Anomaly detection framework (Le & Zhang, 2022).

3.1 Common Framework

The procedure for detecting log anomalies generally involves four essential steps: log parsing, log grouping, log representation, and anomaly detection (Le & Zhang, 2022). This framework is illustrated in Figure 3.1.

3.1.1 Log Parsing

The first step after collecting logs is log parsing, which transforms unrefined log messages into structured format. This entails the automatic segregation of the fixed, constant element (Log Key) of a log message from its variable parts. The consistent element stays unchanged throughout various log entries, whereas the variable component changes. The objective of parsing is to derive log templates by recognizing patterns and substituting variable segments with placeholders. For instance, in Figure 3.1, the * in EventTemplate signifies variable components.

A range of automated log parsing techniques have been developed, leveraging methods like clustering (Shima, 2016), (Hamooni et al., 2016), heuristic (Makanju, Zincir-Heywood, & Milios, 2009), (P. He, Zhu, Zheng, & Lyu, 2017) and longest common sub-sequence (M. Du & Li, 2016). A new log parsing technique, NuLog (Nedelkoski, Bogatinovski, Acker, Cardoso, & Kao, 2021) was introduced that employed a self-supervised learning model and showed enhanced accuracy and efficiency relative to other log parsing methods.

3.1.2 Log Grouping

Following the parsing process, the next step involves transforming the textual logs into numerical features suitable for use in anomaly detection methods. Before this conversion, it is essential to segment log data into distinct groups or sequences through various techniques. Every group represents a series of log events, and from these sequences, feature vectors (or event count vectors) are generated to construct a feature matrix. This matrix acts as the input for models designed to detect anomalies. Logs can be organized into groups through three main windowing techniques:

Fixed Window: In this approach, log events are categorized according to a specified time frame. The window size can fluctuate from seconds to minutes or even hours, depending on the specific issue being addressed. Logs that occur within the same window are considered a single sequence, ensuring there is no overlap between consecutive windows.

Sliding Window: In this approach, logs are organized in a manner akin to the fixed window, but it incorporates an extra parameter—step size. The step size, typically less than the window size, results in overlap between successive windows, producing additional sequences. For instance, a log sequence spanning an hour with a step size of five minutes will result in overlapping windows.

Session Window: In contrast to the earlier two methods, session windows categorize logs by using unique identifiers that monitor various execution paths, facilitating a more organized grouping of related events. For example, certain public datasets use node_id, block_id to identify and group related logs.

3.1.3 Log Representation

After logs are organized into sequences, they are transformed into feature vectors for additional analysis. There are three main types of feature representations:

Quantitative Vector: This is referred to as the log count vector, which records the frequency of each log event within a sequence. For instance, in the sequence [E1 E2 E3 E2 E1 E2], the resulting vector would be [2 3 1], with each number indicating the frequency of each event. This depiction is frequently used in ML methods.

Sequential vector: This vector represents the sequence of events as they unfold. For instance, the sequence [E1 E2 E3 E2 E1 E2] would yield the vector [1 2 3 2 1 2]. DL methods such as DeepLog (M. Du et al., 2017) utilize this method to understand event patterns according to the sequence of their occurrences.

Semantic Vector: In contrast to quantitative and sequential vectors, semantic vectors capture the meaning or context of log events through the use of language models. This method emphasizes the fundamental meaning of log messages instead of their frequency or sequence. For example, in the sequence of log events: [E1: ”Module Not Found”, E2: ”No Override File Found”, E3: ”Error Bad parameters”, E2: ”No Override File Found”, E1: ”Module Not Found”], the semantic vector for each event could look like this:
E1 (”Module Not Found”): [0.57, 0.35, 0.86, ...]
E2 (”No Override File Found”): [0.79, 0.63, 0.45, ...]
E3 (”Error Bad parameters”): [0.91, 0.37, 0.27, ...]

3.1.4 Anomaly Detection:

After extracting the feature vectors, they are input into ML and DL methods for the purpose of detecting anomalies. ML methods generally detect unusual log sequences by analyzing log event count vectors. Conversely, DL methods concentrate on identifying normal patterns within sequential logs and highlighting anomalies that diverge from these established patterns. While ML methods excel at detecting anomalies in static datasets, DL methods are more adept at recognizing intricate temporal patterns in logs. By integrating these techniques, we can efficiently identify anomalies in extensive and evolving systems.

3.2 Existing Methods

A range of ML and DL methods have been employed to identify anomalies in system logs, leveraging both supervised and unsupervised learning methods. In supervised learning, models are trained using labeled datasets, whereas unsupervised learning focuses on training with unlabeled data, with the goal of identifying anomalies based on patterns without any predefined labels. In this study, we have used both types of approaches. Here, we present a summary of the methods applied:

3.2.1 Supervised ML Methods

We used three supervised methods for anomaly detection: Logistic Regression (LR), Support Vector Machine (SVM), and Decision Tree (DT). The effectiveness of supervised methods is greatly affected by the quality and availability of the labeled dataset, as they rely on labeled data for training purposes. Increasing the amount of labeled data boosts a model's ability to learn both typical and atypical patterns, which in turn enhances their accuracy in identifying anomalies.

Logistic Regression: Logistic Regression (LR) (Bodik, Goldszmidt, Fox, Woodard, & Andersen, 2010) is a commonly used classification algorithm, particularly effective for binary tasks, especially in anomaly detection. Using a sigmoid function, it determines the likelihood of an instance being classified into a particular class. In the process of assessing new instances, if the probability exceeds a specified threshold (commonly set at 0.5), the instance is classified as anomalous; otherwise, it is considered normal.

Support Vector Machine: Support Vector Machine (SVM) (Liang et al., 2007) is a supervised classification method that aims to create an optimal hyperplane to separate classes in a high-dimensional space. In anomaly detection, the training data comprises event count vectors along with their corresponding labels. If a new instance is situated below the hyperplane, it is normal; if it is positioned above, then it is anomalous.

Decision Tree: Decision Tree (DT) (M. Chen et al., 2004) predicts results by using a sequence of nodes that divide data according to the most significant attribute, often using metrics such as information gain (J. Han, Pei, & Tong, 2022). Beginning with the root node, the data are partitioned until a stopping criterion is reached, like having uniform class instances. To classify a new instance, one navigates the decision tree from the root to a leaf node, which indicates the predicted class for that instance.

3.2.2 Unsupervised ML Methods

As previously mentioned, unsupervised methods are ideal for real world settings where labeling is frequently impractical. In this study, we used Principal Component Analysis (PCA), Isolation Forest (IF), and Log Clustering (LC) to detect anomalies without pre-labeled data, which allows greater scalability and flexibility in anomaly detection.

PCA: Principal Component Analysis (PCA) (Xu et al., 2009) is a technique for reducing dimensionality that selects key principal components to capture primary variations, reducing data to a lower-dimensional space. Early research on PCA for log-based anomaly detection used event count vectors to identify patterns. The data was divided into a normal space (Sn) with leading components and an anomaly space (Sa) with others. If the calculated projection length calculated on the anomaly space surpasses a specified threshold, the log sequence is marked as an anomaly.

Isolation Forest: Isolation Forest (IF) (Liu et al., 2008) identifies anomalies by leveraging their rarity, making them easier to isolate through random partitioning. This approach constructs a collection of Isolation Trees (iTrees) where anomalies are identified by their shorter average path lengths. In log-based anomaly detection, each Isolation Forest tree randomly selects an event count feature and value to split the data, isolating unique patterns. Instances with rare patterns show shorter average path lengths and are isolated faster. To identify anomalies, the isolation score of each instance is evaluated against a set threshold: instances with lower scores are marked as anomalies, while those with higher scores are considered normal.

Log Clustering: LogCluster (LC) (Lin et al., 2016) organizes logs for anomaly detection in two phases. First it converts log sequences into event count vectors, categorizing them as normal or abnormal, with each cluster represented by a centroid vector stored in a knowledge base. In the second phase, new vectors are compared to these centroids. If the nearest centroid is within a threshold distance, the vector joins that cluster; otherwise, a new cluster is created. Anomalies are identified by assessing the distance between a latest log sequence and the corresponding vectors stored in the knowledge base. If the closest distance surpasses the threshold, the log sequence is categorized as an anomaly.

3.2.3 Deep Learning Methods

To take advantage of neural networks for log anomaly detection, various deep learning techniques have been used that involve Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Transformers, etc. In this research, we employed two methods: DeepLog and LogRobust. DeepLog functions as an unsupervised model, identifying patterns in log data to uncover anomalies without requiring labeled inputs, whereas LogRobust is a supervised model that employs labeled data to enhance anomaly detection.

DeepLog: DeepLog (M. Du et al., 2017) is a complex deep learning model that detects log anomalies using Long short-term memory (LSTM) models and density clustering. The model captures sequential dependencies between log events by representing log messages by their log event indices. It functions through a predictive approach, acquiring knowledge of the typical patterns found in log sequences. When a deviation from the established normal pattern takes place, it signals the occurrence as a possible anomaly. This method successfully identifies anomalies by forecasting and recognizing deviations from anticipated log behaviors.

LogRobust: LogRobust (X. Zhang et al., 2019) is a supervised classification deep neural network model designed to address the challenges posed by the instability of logs resulting from noisy processing and logging systems. Unlike other models that primarily rely on log counting vectors for features, LogRobust converts log events into semantic vectors. In this method a pre-trained word2vec (Joulin, 2016) model was employed, integrated with TF-IDF weights to generate representation vectors for log templates. The semantic vectors are subsequently input into an Attention-based Bi-LSTM classification model designed to identify anomalies. This approach has shown promising results in successfully addressing log instability.

3.3 Evaluation Study

In this section, we discuss the dataset used, the experimental setup, and the evaluation result of the machine learning and deep learning methods. We also present a comparison to a public dataset

Table 3.1: Dataset overview.

Dataset	Time Span	# Logs	# Anomalies
Buspas	14 hours	30,730	184

and assess each model's efficiency, highlighting its comparative performance.

3.3.1 Experiment Design

Log Dataset

The dataset we used in this experiment is composed of system logs extracted from a SCiNe device. The logs document system activities, encompassing boot messages, kernel updates, and hardware events. A total of 30,730 log messages were gathered during a 14-hour timeframe, representing the complex pattern of real-world system logs. Every log entry contains details like the date, time, device name, and the content of the message.

Subsequently, the logs were manually labeled as either normal or anomalous in collaboration with domain experts. The manual labeling ensured that the labels accurately reflected the operational behavior of the system. The dataset, however, showed a notable class imbalance, containing only 184 anomalous messages (less than 1% of the dataset), which presents a fundamental challenge for anomaly detection models. Table 3.1 summarizes the dataset.

This work was specifically designed for the context of the Buspas system, providing a detailed perspective that is frequently lacking in large-scale studies. While based on a limited dataset, this study provides valuable insights into log-based anomaly detection methods, highlighting their applicability to more extensive datasets. Furthermore, the manual labeling process establishes a solid basis for handling proprietary and domain-specific logs, effectively addressing gaps often present in current large-scale studies.

Experimental Setup

In our experiment, we preprocess the log data and conduct anomaly detection method as follows:

Log Parsing: We made use of various log parsing techniques to transform the unstructured logs

Table 3.2: Log parser performance.

Log Parser Name	**Time Taken (sec)**	**# of EventTemplates**
LenMa (Clustering)	40.881	15,646
Drain (Log Structured Heuristics)	**3.445**	**253**
AEL (Log Structured Heuristics)	4.151	252
Spell (Longest Common Subsequence)	4.263	347

into structured log templates. We used four parsers: LenMa, Drain, AEL, and Spell, from the toolkit LogParser (Zhu et al., 2019). Among these, Drain demonstrated the highest levels of accuracy and efficiency. The performance of each parser is illustrated in Table 3.2.

Log Grouping and Feature Extraction: We employed fixed and sliding window techniques for log grouping in our dataset, as the lack of identifiers excluded the session window approach. A log sequence, in this context, denotes a set of log templates that exist within a defined time frame. The window size varied from 10 minutes to 1 minute, with step sizes ranging from 5 to 0.5 minutes, based on the particular experiment.

After grouping the logs, we converted the sequences into numerical feature vectors. For each machine learning model, we generated quantitative vectors (event count vectors), marking a log sequence as anomalous if any log messages were classified as abnormal. For DeepLog, log sequences were transformed into sequential vectors by indexing each log event, while for LogRobust we used the established method to convert log sequences into semantic vectors.

Table 3.3 presents the number of sequences produced for each combination of window and step sizes. In the fixed window setting with a window size of 5 minutes, a total of 167 log sequences were produced, with 68 identified as anomalous and 99 classified as normal. In contrast, using the sliding window approach with a 5-minute window size and a step size of 2 minutes, a total of 414 sequences were produced, which included 77 anomalous sequences and 337 normal sequences. The increased number of sequences in the sliding window approach is due to the overlap between consecutive windows, leading to more comprehensive groupings.

Table 3.3: Log sequence summary.

Window Size				
10 min	7 min	5 min	3 min	1 min
Total: 84 instances, 53 anomaly, 31 normal	Total: 119 instances, 63 anomaly, 56 normal	Total: 167 instances, 68 anomaly, 99 normal	Total: 278 instances, 70 anomaly, 208 normal	Total: 833 instances, 89 anomaly, 744 normal
Step Size				
5 min	3 min	2 min	1 min	0.5 min
Total: 165 instances, 68 anomaly, 97 normal	Total: 276 instances, 71 anomaly, 205 normal	Total: 414 instances, 77 anomaly, 337 normal	Total: 830 instances, 91 anomaly, 739 normal	Total: 1663 instances, 111 anomaly, 1552 normal

Anomaly Detection: During this phase, various machine learning and deep learning techniques were trained using the features obtained in the prior step, each following its specific methodology. The dataset was split into 80% for training purposes and 20% for testing purposes. In the case of unsupervised methods, labels were omitted from the training data, since these methods do not need labeled data for the learning process.

All experiments were carried out on a machine featuring an 11th Gen Intel(R) Core(TM) i7-1185G7 Processor @ 3.00GHz and 16 GB of RAM. The parameters for each method were adjusted to guarantee peak performance. Each model underwent several iterations, with the most favorable outcomes being presented.

Evaluation Metrics

We evaluated the accuracy of the method through Precision, Recall, Specificity, and F-measure, given that log-based anomaly detection is a binary classification task. Precision measures the proportion of correctly identified anomalies compared to the total instances that the model categorizes as anomalies. Recall evaluates how accurately true anomalies are identified from the overall count of actual anomalies. Specificity denotes the proportion of correctly recognized normal sequences compared to the overall count of genuine normal sequences. F1-score serves as the harmonic mean of precision and recall, providing a well-rounded evaluation of model performance.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

$$F - measure = \frac{2 * Precison * Recall}{Precison + Recall}$$

True Positive (TP) denotes the count of anomalies that the model accurately detects, whereas True Negative (TN) indicate the normal log sequences that are accurately recognized as normal. A False Positive (FP) occurs when the model mistakenly identifies normal log sequences as anomalies, while False Negatives (FN) refers to the actual anomalies that the model fails to detect.

3.3.2 Performance of Anomaly Detection Methods

In this section, we discuss the performance of machine learning (ML) and deep learning (DL) models with respect to their accuracy. The findings are detailed for three supervised machine learning models, three unsupervised machine learning models, and two deep learning models, examined across different window configurations. Every set of models is examined thoroughly to emphasize its strengths, limitations, and adaptability to various experimental conditions. In conclusion, we present a brief overview of the performance trends noted for each category of models.

Accuracy of Supervised ML Methods

In supervised methods, while splitting the dataset we balanced positive and negative samples to minimize bias and enhance the models' ability to identify both normal and anomalous logs. All three methods — Logistic Regression, SVM, and Decision Tree — exhibited perfect precision and specificity across both fixed and sliding windows, with no false positives in any configuration. However, recall varied based on the type and size of the window. Results are shown in Table 3.4.

Table 3.4: Accuracy of supervised ML methods.

Fixed Window						
Model	WS	10 min	7 min	5 min	3 min	1 min
LR	P	1	1	1	1	1
	R	0.636	0.923	0.929	0.857	0.889
	S	1	1	1	1	1
	F1	0.778	0.96	0.963	0.923	0.941
SVM	P	1	1	1	1	1
	R	0.909	0.923	0.929	1	0.944
	S	1	1	1	1	1
	F1	0.952	0.96	0.963	**1**	0.971
DT	P	1	1	1	1	1
	R	0.909	0.923	0.929	1	0.889
	S	1	1	1	1	1
	F1	0.952	0.96	0.963	**1**	0.941
Sliding Window						
Model	WS, SS	10 min, 5 min	7 min, 3 min	5 min, 2 min	3 min, 1 min	1 min, 0.5 min
LR	P	1	1	1	1	1
	R	0.929	0.867	0.813	0.895	0.957
	S	1	1	1	1	1
	F1	0.963	0.929	0.897	0.944	0.978
SVM	P	1	1	1	1	1
	R	0.929	0.8	0.938	0.947	0.957
	S	1	1	1	1	1
	F1	0.963	0.889	0.968	**0.973**	0.978
DT	P	1	1	1	1	1
	R	0.929	0.8	0.875	0.947	0.913
	S	1	1	1	1	1
	F1	0.963	0.889	0.933	**0.973**	0.955

In fixed windows, Logistic Regression (LR) shows improved recall as the window size decreases, indicating that smaller windows capture more anomalies. SVM performs reliably but experiences a slight decline in recall with larger windows (e.g., 10 and 7 minutes). However, it improves with smaller windows, enhancing both recall and F1 scores. The Decision Tree (DT) behaves similarly to SVM but has slightly lower recall with larger windows. It excels at a 3-minute window, achieving perfect test results.

Sliding windows increase the number of instances by overlapping consecutive windows, enhancing model ability to detect patterns and anomalies. However, step size is critical for accurate detection. LR and SVM consistently perform well, particularly with smaller windows and step sizes in detailed and imbalanced data. In contrast, Decision Tree shows variability in recall and F1 scores, facing more challenges in generalization compared to LR and SVM. As window and step sizes decrease, all models improve, but LR and SVM exhibit greater robustness in handling imbalanced datasets and identifying anomalies.

In summary, LR and SVM demonstrated impressive adaptability across different experimental conditions, successfully handling imbalanced data and variations in window and step sizes with significant consistency. Although there are challenges with generalization, DT's remarkable performance in limited anomaly detection situations highlights its potential for targeted applications. The trends suggest that selecting the optimal windowing strategy and step size is crucial for improving the performance of supervised machine learning models in log-based anomaly detection.

Accuracy of Unsupervised ML Methods

For unsupervised methods fixed and sliding window techniques were used with the same parameters as those applied in supervised methods. Although unsupervised methods do not require labels for training, we employed our labeled dataset during testing to assess their performance.

PCA achieves perfect precision and specificity with fixed windows, avoiding false positives, but missing many true anomalies due to low recall. Isolation Forest (IF) maintains high precision and specificity across most window sizes but struggles with smaller windows, leading to more false positives and reduced recall. In contrast, LogClustering (LC) offers a better balance between precision and recall, particularly excelling with smaller window sizes. Although its precision and specificity are slightly lower than PCA and Isolation Forest, LogClustering achieves the highest F1 scores with shorter windows, demonstrating better adaptability in a detailed fixed window context.

In sliding windows, PCA shows improved recall with smaller window and step sizes, achieving a recall of 0.9 with a 5-minute window and 2-minute step, though precision drops. Isolation Forest maintains high precision, particularly in larger windows, but its recall is limited, while performance declines in smaller settings. LogClustering displays more variability; it achieves high recall in some configurations (e.g., W=7, S=3) but had lower precision. This suggests LogClustering is more effective at detecting anomalies, but with increased false positives in detailed configurations. Table 3.5 represents the results.

In summary, PCA demonstrates strong precision and specificity, while it faces challenges in

Table 3.5: Accuracy of unsupervised ML methods.

Fixed Window						
Model	WS	10 min	7 min	5 min	3 min	1 min
PCA	P	1	1	1	1	1
	R	0.286	0.25	0.222	0.222	0.25
	S	1	1	1	1	1
	F1	0.444	0.4	0.364	0.364	0.4
IF	P	1	1	1	1	0.25
	R	0.286	0.25	0.222	0.222	0.083
	S	1	1	1	1	0.981
	F1	0.444	0.4	0.364	0.364	0.125
LC	P	0.333	0.364	0.333	0.25	0.083
	R	0.571	0.5	0.667	0.778	0.583
	S	0.2	0.563	0.52	0.553	0.503
	F1	0.421	0.421	**0.444**	0.378	0.145
Sliding Window						
Model	WS, SS	10 min, 5 min	7 min, 3 min	5 min, 2 min	3 min, 1 min	1 min, 0.5 min
PCA	P	1	1	0.161	1	0.333
	R	0.222	0.182	0.909	0.25	0.308
	S	1	1	0.278	1	0.975
	F1	0.364	0.308	0.274	0.4	0.32
IF	P	1	1	1	0.333	0.2
	R	0.222	0.182	0.182	0.083	0.23
	S	1	1	1	0.987	0.966
	F1	0.364	0.308	0.308	0.133	0.222
LC	P	0.375	0.32	0.2	0.071	0.073
	R	0.667	0.727	0.636	0.5	0.462
	S	0.583	0.622	0.625	0.494	0.763
	F1	**0.48**	0.444	0.311	0.125	0.126

identifying true anomalies. Isolation Forest excels with larger windows but struggles with intricate configurations. LogClustering strikes an impressive balance, demonstrating excellence in recall and F1 scores, although it requires meticulous tuning to reduce false positives. These trends highlight the importance of choosing the right method according to the particular needs of the anomaly detection task, including an appropriate balance between precision and recall.

Accuracy of DL Methods

DeepLog shows reliable recall and F1-scores in larger fixed windows (10-minute and 7-minute), emphasizing its ability to capture long-term dependencies. Nonetheless, its performance diminishes in smaller windows, revealing constraints in managing intricate patterns, although it attains greater

Table 3.6: Accuracy of deep learning methods.

Fixed Window					
Model	WS	10 min	7 min	5 min	3 min
DeepLog	P	0.75	0.727	0.6	0.692
	R	0.857	1	0.75	1
	S	0.8	0.812	0.84	0.913
	F1	0.8	0.842	0.667	0.818
LogRobust	P	0.778	0.889	1	0.75
	R	1	1	1	1
	S	0.8	0.938	1	0.935
	F1	0.875	0.941	**1**	0.857
Sliding Window					
Model	WS, SS	10 min, 5 min	7 min, 3 min	5 min, 2 min	3 min, 1 min
DeepLog	P	0.667	0.75	0.75	0.355
	R	0.923	1	0.273	0.393
	S	0.684	0.833	0.966	0.851
	F1	0.774	0.857	0.4	0.373
LogRobust	P	0.867	0.9	1	1
	R	1	1	1	0.964
	S	0.895	0.944	1	1
	F1	0.927	0.947	**1**	0.982

specificity in these instances. This indicates that DeepLog is more appropriate for situations that demand a wider contextual comprehension instead of detailed anomaly detection.

In comparison, LogRobust consistently surpasses DeepLog in all fixed window settings, attaining flawless recall, precision, and specificity at the 5-minute window. This emphasizes LogRobust's flexibility with different window sizes and its ability to manage imbalanced datasets efficiently.

In sliding windows, DeepLog enhances specificity and F1-score with a 7-minute window and a 5-minute step size, yet its effectiveness declines with smaller window configurations, highlighting difficulties in detecting overlapping anomalies. Conversely, LogRobust demonstrates outstanding performance in sliding window configurations, exceeding its fixed window capabilities and exhibiting remarkable recall and precision across various step sizes. The adaptability of LogRobust establishes it as a dependable approach for identifying anomalies in both fixed and sliding configurations, especially in thorough analyses.

In summary, although DeepLog demonstrates advantages in extensive windows and wider contexts, LogRobust stands out as the more resilient and flexible model, achieving better outcomes

Table 3.7: Comparison with public dataset.

Model	**HDFS Dataset**			**Private Dataset**			
	Precision	Recall	F1	Precision	Recall	F1	Window Type
LR	0.95	1	0.98	1	0.96	0.98	Sliding (WS: 1 min, SS: 0.5 min)
Decision Tree	1	0.99	1	1	0.95	0.97	Sliding (WS: 3 min, SS: 1 min)
SVM	0.95	1	0.98	1	1	1	Fixed (WS: 3 min)
PCA	0.98	0.67	0.79	1	0.29	0.44	Fixed (WS: 10 min)
Isolation Forest	0.83	0.78	0.8	1	0.29	0.44	Fixed (Window Size: 10 min)
LogClustering	0.87	0.74	0.8	0.32	0.73	0.44	Sliding (WS: 7 min, SS: 3 min)
DeepLog	0.95	0.96	0.96	**0.9**	**1**	**0.95**	Sliding (WS: 7 min, SS: 3 min)
LogRobust	0.98	1	0.99	**1**	**1**	**1**	Fixed (WS: 5 min)

across diverse experimental scenarios. The results highlight the significance of choosing deep learning methods tailored to the particular needs of anomaly detection tasks. The results are presented in Table 3.6.

3.3.3 Comparison with Public Dataset

We evaluated the accuracy of each method used on our small-scale dataset against their performance on the publicly available HDFS dataset Xu et al. (2009). The HDFS dataset is composed of 575,061 log blocks, with 16,838 blocks (2.9%) identified as anomalous M. Du et al. (2017). The benchmark results for these methods were obtained using the HDFS dataset, employing a session windowing approach for log grouping. To facilitate a meaningful comparison, we showcased the optimal results of each method on our small dataset, using specific window types and sizes, as illustrated in Table 3.7. This comparison uncovers several important insights:

The outcomes of supervised methods applied to our limited dataset are remarkably close to, and in some cases better than, the benchmark results on the HDFS dataset. This emphasizes the capability of a properly labeled dataset, regardless of its size, to achieve similar accuracy.

In contrast, the performance of unsupervised models was notably lower when compared to the HDFS dataset. This indicates that unsupervised methods need a more extensive dataset to successfully categorize data and detect anomalies, as their effectiveness is significantly dependent on the presence of varied patterns and an adequate amount of data.

Both deep learning methods excelled on our small dataset, showcasing impressive accuracy even

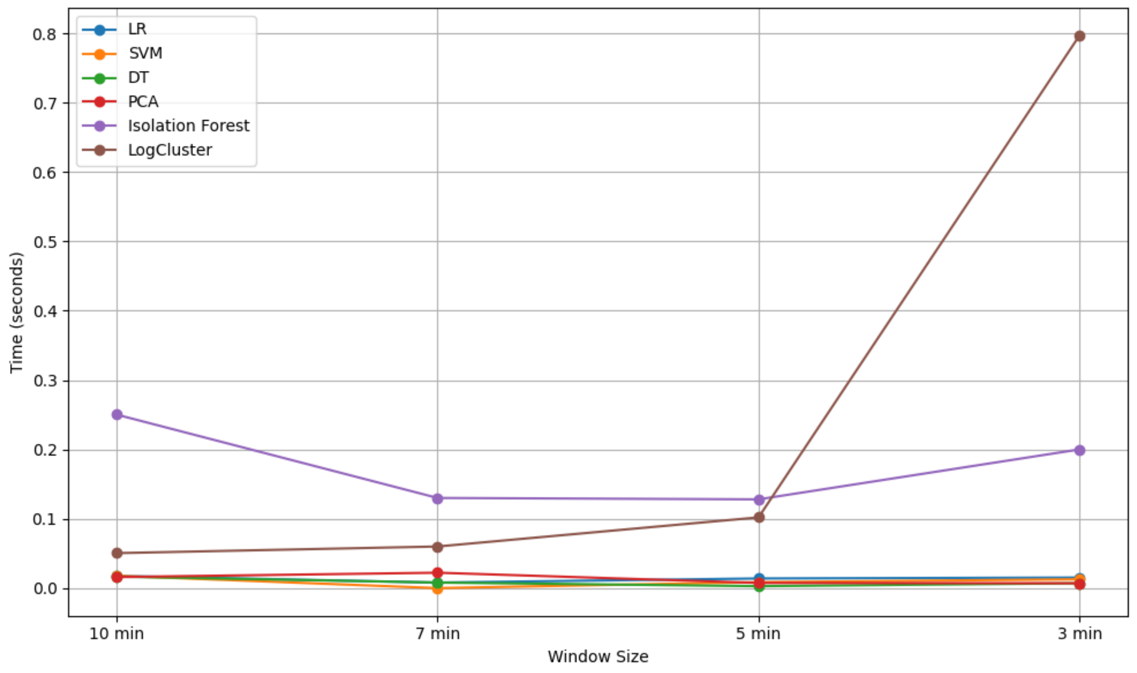


Figure 3.2: Running time of ML methods with varying window size.

with the restricted data size. This suggests that deep learning models, especially those using sequential and semantic patterns, can adjust well to small-scale datasets while maintaining performance levels.

This study further confirms that log-based anomaly detection methods can be adapted to datasets of different sizes, opening up possibilities for their use in both small-scale and large-scale systems.

3.3.4 Efficiency of Anomaly Detection Methods

Efficiency measures how quickly a model can perform anomaly detection. We assess this efficiency by tracking the running time required for the anomaly detector during both the training and testing phases.

Figure 3.2 shows that supervised machine learning methods maintain low processing times across different window sizes, with Decision Tree (DT) being the fastest. This indicates that supervised methods are efficient and stable despite changes in window size. In contrast, most unsupervised methods have longer processing times, although PCA performs comparably to the supervised

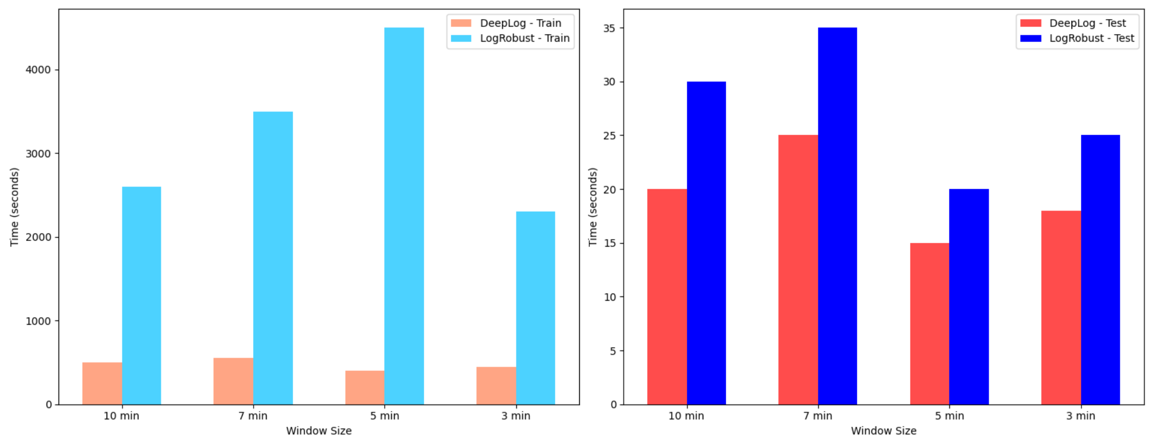


Figure 3.3: Efficiency of Deep Learning methods

methods. Isolation Forest has the longest processing time overall, while LogClustering slows down significantly with smaller window sizes, highlighting the greater computational demands of these methods, especially with larger log sequences.

In deep learning, DeepLog demonstrates impressive training times, achieving the shortest duration in the 5-minute window and maintaining efficient testing capabilities. In contrast, LogRobust has significantly longer training times, especially in the 5-minute window, but performs better in testing speed. While DeepLog is more efficient in training, LogRobust may offer advantages for specific performance needs, particularly for faster inference during testing. Figure 3.3 illustrates these results.

3.4 Discussion

This study provides a comparative evaluation of supervised, unsupervised, and deep learning techniques for log-based anomaly detection using real-world data. Although the findings offer valuable insights, certain aspects require further discussion, particularly regarding dataset limitations, comparative evaluation, and model efficiency. This section discusses the limitations of our study and suggests possible directions for future research.

Addressing Dataset Constraints: One primary limitation of this study is the relatively small dataset, which could affect the generalizability of the results to larger systems. Our comparison with the publicly available datasets shows that the performance trends in our dataset are consistent with those found in larger datasets. This indicates that our findings, despite being derived from a restricted dataset, continue to be relevant and insightful. Moreover, our dataset captures real-world limitations such as the lack of labeled data and unstructured log formats, making it relevant for practical applications. In the future, we will focus on broadening the scope of our study by integrating more extensive datasets that feature a larger volume and diversity of log messages.

Comparative Evaluation of Methods: Our analysis shows the key trade-offs between different anomaly detection methods. Supervised methods such as logistic regression, SVM, and Decision Tree demonstrated impressive accuracy when there was access to labeled data; however, they tend to be less effective in real-world scenarios where labeled anomalies are limited. Unsupervised methods such as PCA and Isolation Forest identified anomalies without the need for labeled data, yet they exhibited variability in both precision and recall, especially when employing various windowing techniques. Deep learning models (LogRobust) achieved the optimal precision-recall balance, though they demanded significant computational resources and extended training time.

The findings indicate that the selection of method is contingent upon the particular needs of the system. For scenarios that require real-time detection, conventional machine learning models, such as SVM and Decision Tree, provide quick and reliable options. In situations where there are limited labeled data, unsupervised methods such as LogClustering can be employed, although they necessitate careful parameter tuning to minimize false positives. Deep learning models like LogRobust provide exceptional performance, yet they might be better suited for batch processing or environments with ample computational resources.

Timeliness and Efficiency of Approaches: Efficiency plays a vital role in anomaly detection, especially in the context of real-time applications. Our evaluation indicates that Decision Tree and SVM demand considerably less processing time, making them appropriate for real-time anomaly detection. Conversely, deep learning models, especially LogRobust, requires extended training periods yet, provide enhanced accuracy. The findings indicate that in industrial environments, lightweight ML models might be more suitable for real-time monitoring, while deep learning techniques offer enhanced accuracy for detecting anomalies in historical log analysis.

Future Work: Future research will aim to broaden the dataset and integrate semi-supervised learning methods to lessen the reliance on manual labeling, thus enhancing efficiency. Furthermore, the integration of various feature representations, such as the combination of sequential and semantic vectors, could enhance detection performance even more by capturing more complex data patterns. We also plan to investigate additional techniques like invariant mining (Lou et al., 2010), LogAnomaly (Meng et al., 2019), and CNN (Lu et al., 2018) to assess model effectiveness across a broader range of methods. Additionally, we aim to deploy the most effective method for real-world anomaly detection.

In summary, this study provides a foundation for understanding the trade-offs among various anomaly detection methods and offers valuable insights into their relevance for real-world log analysis. The findings highlight the significance of choosing models that align with system constraints, computational efficiency, and the availability of data.

Chapter 4

Log-Based Anomaly Detection Without Ground-Truth: Evaluating Weakly Supervised, Semi-Supervised, and Unsupervised Deep Learning Approaches

In this chapter, we present weakly supervised learning as a practical and scalable alternative for log-based anomaly detection, aimed at addressing the limitations found in traditional supervised, semi-supervised, and unsupervised approaches. These limitations include the high cost of manual labeling, susceptibility to format inconsistencies, and reduced robustness in the presence of noise. We evaluate the effectiveness of four deep learning models under weakly supervised, semi-supervised, and unsupervised learning paradigms and benchmark their performance against two fully supervised baselines. We also present our findings, that offer valuable insights into model selection by considering label quality, industrial constraints, and deployment scalability, thereby providing actionable recommendations for effective log-based anomaly detection in real-world industrial environments.

4.1 Methodology

4.1.1 Dataset Collection and Annotation

In this experiment, we collect system logs from the SCiNe device to examine its operational behavior. We specifically choose system logs from the various log categories because they record essential system activities, such as kernel events, boot messages, and hardware operations. The data set initially consisted of five months of log messages, resulting in a substantial volume of data. However, excessive redundancy in log messages makes many logs unnecessary in detecting anomalies. To create a data set that is representative and manageable, we select a subset of system logs comprising 598,237 log messages chronologically appearing in 24 hour time span, offering a real depiction of industrial logs while ensuring a variety of data.

For labeling, we used a heuristic-based method, classifying log messages as anomalous if they included terms linked to system errors (e.g., ”error”, ”err”, ”fail”, ”warn”), whereas all other messages were designated as normal. Using this method, 46,526 logs (7.78% of the total dataset) were recognized as anomalous. The labeling was done using simple python coding.

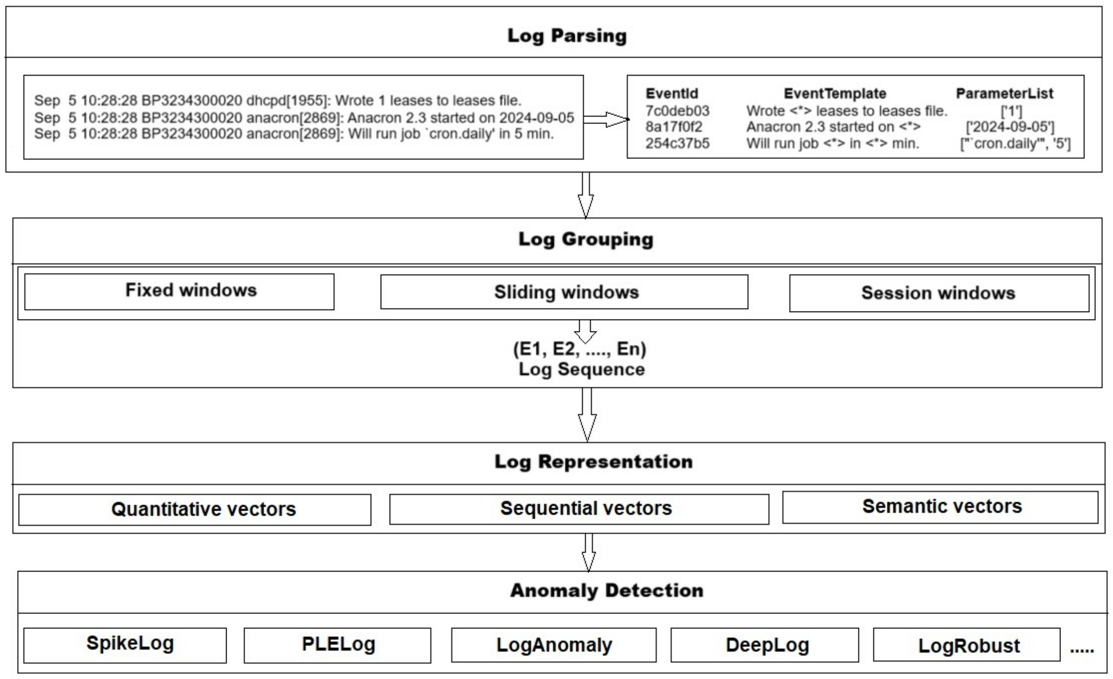
This labeling method, however, has its drawbacks — not all logs that include these terms are genuinely anomalous, and some real anomalies may not contain these keywords. Yet, heuristic labeling remains a popular, scalable, and versatile approach that facilitates quick and effective annotation while requiring minimal human input.

Furthermore, this weakly labeled dataset holds significant value for training weakly supervised and semi-supervised models, as these methods are specifically tailored to manage noisy and unlabeled data. This corresponds to our main objective of creating scalable log anomaly detection techniques that require little manual involvement, making them suitable for industrial use.

4.1.2 Common Workflow for Log-based Anomaly Detection

Once log data have been collected, the majority log-based anomaly detection techniques follow a four-stage process: log parsing, log grouping, log representation, and anomaly detection. Figure 4.1 presents this common workflow.

- Log Parsing: Log parsing converts unstructured raw logs into a semi-structured format by



```
Sep  5 10:28:28 BP3234300020 dhcpd[1955]: Wrote 1 leases to leases file.
Sep  5 10:28:28 BP3234300020 anacron[2869]: Anacron 2.3 started on 2024-09-05
Sep  5 10:28:28 BP3234300020 anacron[2869]: Will run job `cron.daily' in 5 min.
```

EventId	EventTemplate	ParameterList
7c0deb03	Wrote <*> leases to leases file.	['1']
8a17f0f2	Anacron 2.3 started on <*>	['2024-09-05']
254c37b5	Will run job <*> in <*> min.	["`cron.daily'", '5']

Figure 4.1: Common workflow for log-based anomaly detection.

separating the constant components (event templates) from the variable parts (parameters). During parsing, log templates are extracted by recognizing patterns and substituting variable elements with placeholders . Several effective log parsers have been developed for this purpose, such as Spell (M. Du & Li, 2016), Drain (P. He et al., 2017), NuLog (Nedelkoski et al., 2021).

- Log Grouping: Following the parsing stage, logs are divided into sequences of events, a procedure referred to as log grouping. Every group or sequence serves as a unit of analysis to identify anomalies. Three commonly used grouping strategies are: i) Fixed window: Logs are organized according to a set time period or a specific count of log entries. All logs in the same window are considered as a single sequence, and there is no overlap between windows. ii) Sliding window: This approach resembles fixed windows, yet it incorporates a step size parameter that determines the distance the window shifts. The step size is usually less than the window size, resulting in overlapping sequences and increasing the number of log sequences generated. iii) Session window: Logs are grouped according to unique identifiers (e.g., session_id) that monitor individual execution paths. This approach is frequently applied in datasets like HDFS, where logs can be easily divided by session context.

- Log Representation: After grouping, log sequences need to be transformed into feature vectors that are suitable for anomaly detection models. Three main representation techniques are commonly applied: i) Quantitative vectors: These vectors count how often each log event occurs within a sequence. ii) Sequential Vectors: Sequential representations maintain the order of events in the sequence, allowing models to understand patterns of event transitions. iii) Semantic Vectors: Semantic representations encapsulate the context and meaning of log events through the use of language models or word embeddings. This method prioritizes mapping each event to a dense vector, which captures the semantic relationships between events, rather than focusing on frequency or order.

- Anomaly Detection: After extracting feature vectors from log sequences, the last step is to implement anomaly detection models to identify irregularities in the log data. A variety of deep learning techniques have been developed to address key challenges inherent to log-based anomaly detection, including handling noisy data, adapting to evolving log formats, managing log instability, and mitigating class imbalance. The selection of a model frequently relies on the nature of the dataset, the representation of features, and the intended balance between accuracy and computational efficiency.

4.1.3 Log-based Anomaly Detection Approaches

SpikeLog (Qi et al., 2023) employs a weakly supervised learning method for anomaly detection, leveraging a limited amount of labeled data combined with a substantial quantity of unlabeled data. The design uses a Spiking Neural Network (SNN) architecture that demonstrates exceptional resilience to label noise and anomaly contamination, effectively managing mislabeled data while maintaining high anomaly detection performance.

PLELog (Yang et al., 2021) uses a semi-supervised technique to prevent manual labeling by using historical anomalies and probabilistic label estimation. It helps model sequential dependencies and focus on significant log events by representing log sequences with semantic vectors, clustering with HDBSCAN, and detecting anomalies with an attention-based GRU network.

DeepLog (M. Du et al., 2017) is an LSTM based anomaly detection method that uses sequential vectors as input for analyzing normal log event sequences. The approach employs forecasting, allowing the model to predict the subsequent log event in a sequence. If the actual event is not found in the top predicted events, it is identified as an anomaly.

LogAnomaly (Meng et al., 2019) is an unsupervised deep learning method that uses sequential and quantitative features to identify anomalies in unstructured logs. It aligns new log events with templates via semantic integration and employs an LSTM-based network to capture patterns, enhancing anomaly detection across diverse datasets.

LogRobust (X. Zhang et al., 2019) is a supervised deep learning model that uses semantic representations of log events to detect anomalies in logs. It merges pre-trained Word2Vec embeddings with TF-IDF weighting to build semantic vectors that reflect contextual significance and relevance of words in log messages. The semantic vectors are supplied into an Attention-based Bi-LSTM classifier for anomaly detection, that can effectively handle log instability and variability.

LogCNN (Lu et al., 2018) is a supervised model that uses CNN to detect abnormalities in logs. Initially, logs are organized into sessions and then transformed into a matrix format suitable for model training. The matrix representation is used to train the model to identify log sequences as normal or anomalous, detecting abnormal behaviors through spatial patterns in log data.

4.2 Experimental Setup

4.2.1 Dataset Overview

We gathered system logs from the industrial device SCiNe and employed a heuristic-based labeling methodology to annotate the data, as outlined in Section 4.1.1. Typically, each log entry contains a date, time, device name, log source, and a descriptive message. In order to maintain the diverse and complex nature of industrial logs, we refrain from any removal or filtering of log messages, thereby keeping the data set relevant for practical use in real-world scenarios.

Table 4.1: Log sequence summary

Window Size	Training Logs		Testing Logs	
	# Log seq	**# Anomaly**	**# Log Seq**	**# Anomaly**
20 logs	20939	12169	8974	3590
50 logs	8376	6743	3590	2956
100 logs	4188	3731	1795	1632
200 logs	2094	1966	898	870

4.2.2 Data Preparation

Log Parsing: We use the log parser Drain (P. He et al., 2017) with its default parameter settings to convert unstructured raw logs into log templates and parameters. This parsing process organizes logs into semi-structured formats, enabling easier analysis downstream. Figure 4.1 shows the process that transforms raw log messages into template-based representation.

Log Grouping: We then segmented the parsed logs into sequences using the fixed window technique, applying window sizes of 20, 50, 100, and 200 log messages. Logs are organized in chronological order according to timestamps, ensuring that sequences accurately represent real-time system behavior. No random shuffling is applied, as a randomized order can lead to data leakage, allowing future data to appear in training sets, which in turn diminishes the validity of evaluation results (Le & Zhang, 2022). In each window, if any log message is marked as anomalous, the whole log sequence is classified as anomalous to ensure consistency in evaluating the sequence level. Table 4.1 displays the total number of sequences generated for each combination of window sizes.

Log Representation: The grouped log sequences are converted into feature vectors, which act as input for the anomaly detection models. The representation of logs differs depending on the method used. SpikeLog and PLELog use semantic vectors produced by a pre-trained Word2Vec model, enhanced with TF-IDF weighting, to effectively capture the contextual meaning of log messages. DeepLog employs sequential vectors, mapping each log event to a distinct index while maintaining the chronological order of events. LogAnomaly integrates sequential vectors with quantitative vectors (log count) to effectively capture the frequency and order of events. LogRobust and LogCNN use semantic vectors, which are derived from Word2Vec embeddings combined with TF-IDF weights to enhance contextual representation.

Anomaly Detection: In this stage, every deep learning model undergoes training using the generated feature vectors. The model architectures are designed specifically for the corresponding methods. SpikeLog uses a dual Spiking Neural Network (SNN) architecture, incorporating semantic vectors as inputs into a Potential-Assisted Spiking Neuron Model (PASNM). The network models membrane potentials and adaptive thresholds across discrete time intervals to produce anomaly scores. PLELog employs a single-layer GRU (Gated Recurrent Unit) network for the purpose of sequence modeling. DeepLog is composed of two LSTM layers, each containing 128 neurons, and is designed to forecast the subsequent log event. In a similar fashion, LogAnomaly incorporates two LSTM layers, each containing 128 neurons, to effectively model both sequence and frequency patterns. LogRobust employs a two-layer Bi-directional LSTM architecture featuring 128 neurons in each layer, augmented by an attention mechanism that highlights significant events within a sequence. LogCNN consists of three convolutional layers featuring different filter sizes, along with a max-pooling layer designed for extracting features from log sequences arranged as matrices.

4.2.3 Implementations and Environments

We applied established log-based anomaly detection methods to our private dataset, ensuring a fair assessment among various models. The dataset was divided into 70% for training and 30% for testing. We meticulously adjusted the parameters for each method to ensure optimal performance. Furthermore, all models underwent training with a consistent number of iterations and learning rate, ensuring unbiased comparisons among various learning paradigms.

All experiments were performed on a Linux-based server using a 13th Gen Intel® Core™ i9-13900 CPU, 64 GB of RAM, and an NVIDIA RTX A2000 GPU with 12 GB of memory. The operating system used is Ubuntu 24.04.1.

4.2.4 Evaluation Metrics

Log anomaly detection is approached as a binary classification task, and we assess the effectiveness of each method through Precision, Recall, Specificity, and F1-score.

- **Precision** $\frac{TP}{TP+FP}$ quantifies the ratio of accurately identified anomalous sequences to the total number of sequences predicted as anomalous. A high precision value means that the model produces minimal false alarms, correctly identifying true anomalies.

- **Recall** $\frac{TP}{TP+FN}$ measures the percentage of true anomalies accurately identified by the model. A higher recall indicates that the model is proficient in identifying the majority of true anomalies, thus reducing the chances of missed detections.

- **Specificity** $\frac{TN}{TN+FP}$ indicates how well the model can accurately recognize normal log sequences among all genuine normal sequences. A high specificity value shows that the model effectively identifies normal behavior, with minimal normal logs incorrectly labeled as anomalies.

- **F1-score** $\frac{2*(Precision*Recall)}{Precision+Recall}$ represents the harmonic mean of precision and recall, offering a balanced assessment of accuracy in identifying anomalies while also reflecting the model's effectiveness in reducing false negatives and false positives.

True Positives (TP) are abnormal log sequences that have been appropriately detected as anomalies, while True Negatives (TN) are normal sequences. False Positives (FP) are normal sequences misidentified as anomalies, and False Negatives (FN) are abnormal sequences the model missed.

4.3 Results and Analysis

We evaluate weakly supervised (SpikeLog), semi-supervised (PLELog), and unsupervised (DeepLog, LogAnomaly) models for anomaly detection in industrial logs with heuristic labels, where noise is a key challenge. Supervised models LogRobust and LogCNN serve as baselines. We assessed model's performance using precision, recall, specificity, and F1-score across log sequence sizes (20, 50, 100, 200). Given heuristic labeling limitations, we also analyze model robustness to label noise, stability across window sizes, and computational efficiency for industrial applicability.

4.3.1 Accuracy of Anomaly Detection Methods

The data presented in the Table 4.2 illustrates that both weakly supervised SpikeLog and semi-supervised PLELog achieve high levels of precision, recall, and specificity across various log sizes.

Table 4.2: Anomaly detection results

Method	Metric	20 logs	50 logs	100 logs	200 logs
SpikeLog	Precision	0.998	0.999	0.998	**1**
	Recall	0.987	0.991	0.987	0.996
	Specificity	0.997	0.996	0.981	**0.999**
	F1-score	**0.993**	0.995	0.992	**0.998**
PLELog	Precision	0.998	0.995	0.999	0.998
	Recall	0.868	0.993	0.988	0.997
	Specificity	0.998	0.979	0.993	0.964
	F1-score	0.929	0.996	0.993	**0.998**
DeepLog	Precision	0.77	0.915	0.946	0.968
	Recall	0.7	0.888	0.788	**1**
	Specificity	0.697	0.617	0.552	0
	F1-score	0.738	0.901	0.86	0.984
LogAnomaly	Precision	0.654	0.896	0.946	0.984
	Recall	0.841	0.81	0.788	0.978
	Specificity	0.364	0.565	0.552	0.536
	F1-score	0.736	0.851	0.86	0.981
*LogRobust	Precision	0.589	1	0.999	1
	Recall	1	0.997	0.998	1
	Specificity	0	1	0.988	1
	F1-score	0.741	0.998	0.998	1
*LogCNN	Precision	1	1	1	1
	Recall	0.996	1	1	1
	Specificity	1	1	1	1
	F1-score	**0.998**	1	1	**1**

This suggests their effectiveness in predicting positive events, detecting true positives and true negatives. The high F1-score across various window sizes demonstrates that these models achieve balance between precision and recall, making them reliable for log anomaly detection.

In contrast, DeepLog and LogAnomaly exhibit low precision with smaller window sizes, which gradually enhances as the dataset grows. This suggests these models need larger datasets to perform well. DeepLog's recall fluctuates, demonstrating difficulties recognizing abnormalities in smaller datasets but improves with additional data. However, DeepLog's low specificity across all window size suggests it faces considerable challenges to detect normal instances. In contrast, LogAnomaly excels at detecting anomalies across various dataset sizes; however, it exhibits low specificity with smaller datasets, which improves considerably with larger datasets. The improvement in F1-score

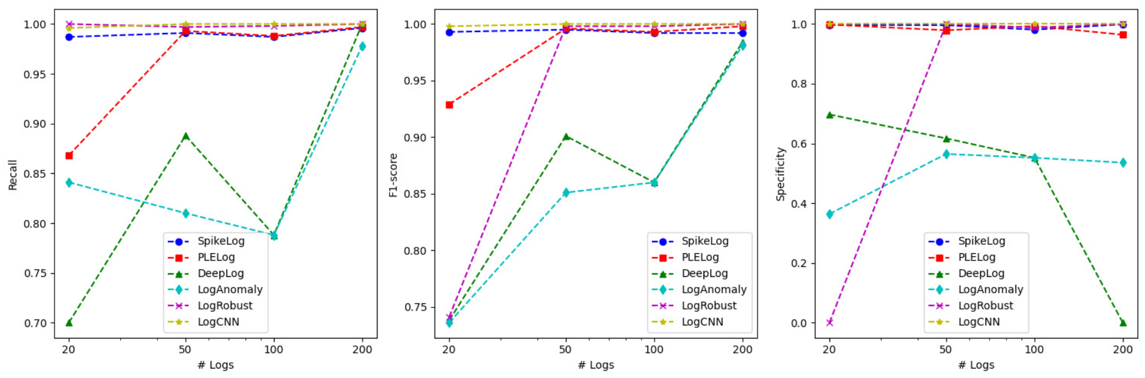


Figure 4.2: Robustness and stability of anomaly detection methods.

from smaller to larger datasets indicates that both models benefit from larger training data, enhancing their ability to detect anomalies as the dataset expands.

The precision of LogRobust starts off low with smaller window sizes but shows improvement and consistency as more data are introduced; meanwhile, its recall stays high across various window sizes. The specificity starts at a low point of 0 for 20 logs but shows a remarkable improvement in identifying true negatives as the dataset increases. Similarly, F1-score demonstrates a significant enhancement with increased data, indicating that LogRobust becomes highly proficient at identifying anomalies as more information is accessible. Conversely, LogCNN demonstrates flawless performance across all dataset sizes, with precision, specificity, and F1-score consistently achieving 1.0. This shows that LogCNN stands out as the most stable and effective model under all conditions, demonstrating excellence in identifying both normal and anomalous instances. Even with smaller data sets, it achieves nearly flawless recall, indicating that it excels at detecting anomalies, even when the data is constrained.

4.3.2 Robustness and Stability of Anomaly Detection Methods

Considering the heuristic nature of our dataset labels, it is essential to ensure the robustness and stability of the models. Robustness under label noise refers to how well a model performs when labels are unreliable, whereas stability across varying window sizes reflects the model's ability to maintain consistent performance despite varying log sizes. We assess these factors by examining recall, F1-score, and specificity across different window sizes, as shown in Figure 4.2.

The results demonstrate that SpikeLog and PLELog consistently attain high recall, F1-score, and specificity across various log sizes, reflecting robust generalization and resilience to label noise.

In contrast, DeepLog and LogAnomaly demonstrate low specificity, indicating an elevated false-positive rate. This indicates that these unsupervised methods have difficulty in effectively distinguishing anomalies, resulting in reduced reliability. The fluctuations in their F1-score across different window sizes further suggest sensitivity to log size variations.

LogRobust, as a supervised baseline, exhibits inadequate recall at the minimal window size of 20 logs. Nonetheless, performance exhibits considerable enhancement with increased log sizes, underscoring its reliance on a substantial volume of labeled data. The abrupt changes in specificity indicate instability at reduced window sizes, highlighting its sensitivity to variations in log sequences. However, LogCNN exhibits consistent performance, showing minimal variation across various window sizes, thus establishing itself as a robust and reliable model.

4.3.3 Efficiency of Anomaly Detection Methods

Efficiency is crucial in log-based anomaly detection, as it determines how quickly a method can identify anomalies. We evaluate this by measuring each method's training and testing run-time. Figure 4.3 presents these times on a logarithmic scale for better readability. SpikeLog exhibits the longest training time compared to the other models, with times varying significantly based on the size of the logs. However, it compensates with faster inference, as it processes events in an event-driven fashion instead of analyzing complete sequences. To find a balance between efficiency and accuracy, we also experimented with fewer training epochs (20 training epochs instead of 100), reducing both the training and testing times considerably. Although accuracy decreased slightly, this trade-off could be beneficial for real-time applications (results shown in Table 4.3). PLELog stands out as the most computationally efficient method, with training times of just a few seconds and testing times well under one second across all window sizes. The efficiency comes from its direct log embedding approach, which eliminates the need for iterative backpropagation, enabling it to be both memory efficient and scalable.

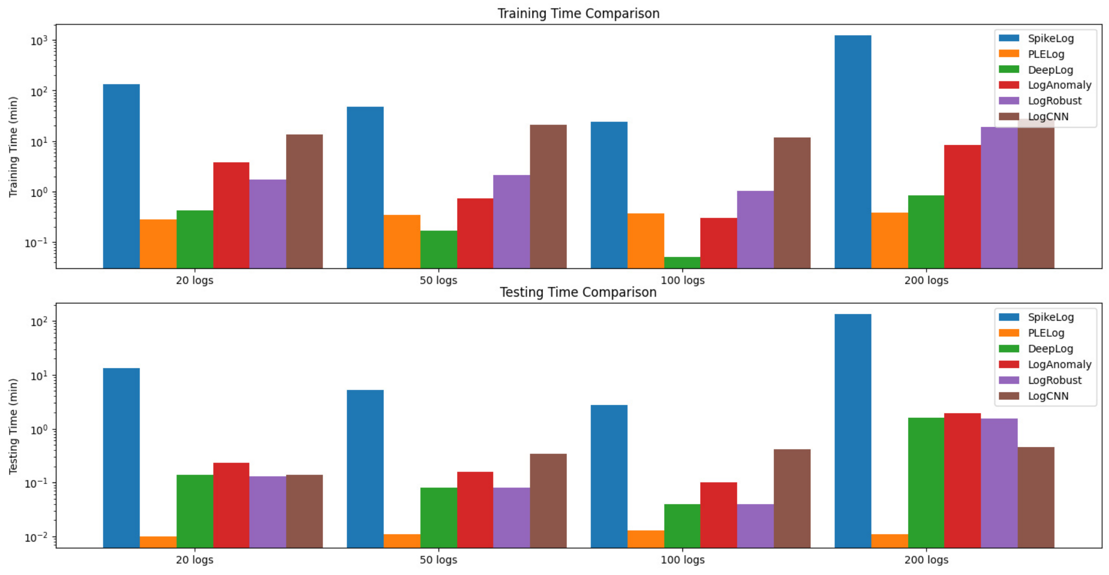


Figure 4.3: Running time of log-based anomaly detection methods.

DeepLog demonstrates a good balance between efficiency and performance, with training durations varying across datasets but generally remaining within a reasonable range. Testing times also scale with the size of the logs, though they remain relatively efficient for larger datasets.

LogAnomaly requires a bit more time due to its capability to analyze both sequential and quantitative patterns, resulting in longer training and testing durations compared to some of the other models.

The baseline model, LogRobust, shows an increase in computational complexity as dataset sizes grow, with training and testing durations rising as well. LogCNN processes log sequences in parallel, leading to longer training times, but its testing times remain relatively short, slightly higher than LogRobust's.

4.4 Discussion

This study presents a comparative evaluation of weakly supervised, semi-supervised, unsupervised, and supervised deep learning techniques for log-based anomaly detection using real-world industrial data. This section discusses the scalability and suitability of these approaches, highlighting their strengths and limitations. Additionally, we address the constraints of our study and propose

Table 4.3: Accuracy and running time of SpikeLog in different epochs

Metric	**20 logs**	**50 logs**	**100 logs**	**200 logs**
Precision	0.998 / 0.991	0.999 / 0.997	0.998 / 0.999	1.0 / 0.999
Recall	0.987 / 0.996	0.991 / 0.985	0.987 / 0.982	0.996 / 0.995
Specificity	0.997 / 0.988	0.996 / 0.987	0.981 / 0.993	0.999 / 0.995
F1-score	0.993 / 0.994	0.995 / 0.991	0.992 / 0.990	0.998 / 0.997
Training Time (min)	136 / 20.98	47.47 / 8.52	24.07 / 4.27	1257 / 226.77
Testing Time (min)	13.37 / 13.20	5.21 / 5.26	2.78 / 2.73	133.81 / 133.13

*First value for epoch 100 and second value for epoch 20.

directions for future research.

4.4.1 Model Scalability and Industrial Suitability

Our experiments reveal how different log-based anomaly detection methods perform on heuristically labeled data. Since our dataset is derived from a real-world industrial setting, evaluating model scalability and industrial applicability is crucial. Different methods exhibit varying levels of robustness and efficiency, making them suitable for different operational scenarios.

Weakly supervised SpikeLog and semi-supervised PLELog demonstrate strong and consistent performance across all log sizes, maintaining high accuracy even with noisy labels. Their ability to adapt to heuristic labels makes them well-suited for industrial applications. Notably, PLELog is the most computationally efficient, requiring minimal training and testing time while maintaining high detection accuracy.

Unsupervised models like DeepLog and LogAnomaly eliminate the need for labeled anomalies, making them attractive for industrial use. However, DeepLog struggles with small datasets and requires large log sequences to perform effectively, limiting its suitability for real-time detection. Similarly, LogAnomaly is highly sensitive to noisy labels and computationally expensive, making it less practical for industrial deployment.

We also evaluated supervised learning models LogRobust and LogCNN as benchmarks for comparison with other approaches. LogRobust outperforms all models when trained on extensive labeled datasets; however, its reliance on clean and abundant labels limits its practicality for industrial settings where labeled data is often scarce or heuristically assigned. LogCNN also requires labeled data, which limits its utility in industrial contexts with sparse or heuristic labels. Despite longer training times, its manageable testing speed makes it a viable option for industrial applications, though label dependency remains a challenge.

4.4.2 Limitations and Future Work

While our study provides valuable insights, it has certain limitations. This section outlines these constraints and suggests future research directions.

Dataset Scope: Our analysis is based on a single industrial dataset sourced from SCiNe, which is heuristically labeled. This labeling process may introduce bias, affecting model evaluation. Additionally, studying only one industrial device limits generalizability. Future work will validate our findings using human-verified anomaly labels and extend the study to logs from diverse industrial devices.

Log Segmentation Approach: Currently, we use a fixed windowing technique to segment logs. Future research will explore alternative segmentation strategies and evaluate their impact on anomaly detection performance.

Expanding Model Exploration: While we assessed multiple deep learning models, further research will explore additional state-of-the-art methods, including hybrid and self-supervised techniques, to improve anomaly detection in unlabeled logs.

Real-Time Deployment: Future efforts will focus on optimizing models for real-time anomaly detection and deploying them on SCiNe to evaluate their effectiveness in live industrial environments.

By addressing these challenges, we aim to enhance the robustness, efficiency, and real-world applicability of log-based anomaly detection methods for industrial settings.

Chapter 5

Summary and Future Research Directions

Log-based anomaly detection is essential to maintain the reliability and security of modern intelligent and industrial systems. However, the complex nature, substantial volume, and often noisy characteristics of real-world logs pose significant challenges in developing reliable and scalable detection systems. In this thesis, we explored a comprehensive set of machine learning and deep learning approaches for detecting anomalies in system logs, focusing specifically on the constraints and variability found in industrial environments.

First, we performed a comparative evaluation of supervised and unsupervised models using a small, manually labeled real-world dataset. We evaluated their effectiveness through extensive experiments, focusing on a range of performance metrics and computational efficiency. Our findings highlighted that while supervised methods like SVM and LogRobust achieved high accuracy, they were significantly reliant on clean, labeled data—making them less feasible for real-world situations with labeling limitations.

Second, we broadened our research to include deep learning models within weakly supervised, semi-supervised, and unsupervised frameworks, utilizing a more extensive dataset that was labeled heuristically. We evaluated models including SpikeLog, PLELog, and LogAnomaly in comparison to fully supervised baselines. The results showed that weakly supervised approaches, particularly

PLELog, struck a commendable balance between performance, robustness to label noise, and computational efficiency—making them more suitable for industrial deployment. Unsupervised methods, despite being label-free, showed limited adaptability to log variability and noise.

In summary, our experimental findings indicated that the effectiveness of anomaly detection models is significantly affected by the learning paradigm, the quality of labels, and preprocessing decisions like the windowing strategy. The suggested comparative framework provides valuable guidance for choosing appropriate models for industrial applications. As potential future directions, we aim to expand the dataset to include multiple industrial systems with human-verified labels, investigate alternative log segmentation techniques, and optimize specific models for real-time deployment in real-world industrial environments.

Bibliography

Baril, X., Coustié, O., Mothe, J., & Teste, O. (2020). Application performance anomaly detection with lstm on temporal irregularities in logs. In *Proceedings of the 29th acm international conference on information & knowledge management* (pp. 1961–1964).

Bodik, P., Goldszmidt, M., Fox, A., Woodard, D. B., & Andersen, H. (2010). Fingerprinting the datacenter: automated classification of performance crises. In *Proceedings of the 5th european conference on computer systems* (pp. 111–124).

Chen, M., Zheng, A. X., Lloyd, J., Jordan, M. I., & Brewer, E. (2004). Failure diagnosis using decision trees. In *International conference on autonomic computing, 2004. proceedings.* (pp. 36–43).

Chen, R., Zhang, S., Li, D., Zhang, Y., Guo, F., Meng, W., … Liu, Y. (2020). Logtransfer: Cross-system log anomaly detection for software systems with transfer learning. In *2020 ieee 31st international symposium on software reliability engineering (issre)* (pp. 37–47).

Du, M., & Li, F. (2016). Spell: Streaming parsing of system event logs. In *2016 ieee 16th international conference on data mining (icdm)* (pp. 859–864).

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 acm sigsac conference on computer and communications security* (pp. 1285–1298).

Du, Q., Zhao, L., Xu, J., Han, Y., & Zhang, S. (2021). Log-based anomaly detection with multi-head scaled dot-product attention mechanism. In *Database and expert systems applications: 32nd international conference, dexa 2021, virtual event, september 27–30, 2021, proceedings, part i 32* (pp. 335–347).

Epaillard, E., & Bouguila, N. (2016). Proportional data modeling with hidden markov models based on generalized dirichlet and beta-liouville mixtures applied to anomaly detection in public areas. *Pattern Recognit.*, *55*, 125–136.

Fan, W., Bouguila, N., & Ziou, D. (2011). Unsupervised anomaly intrusion detection via localized bayesian feature selection. In *2011 ieee 11th international conference on data mining* (p. 1032-1037).

Guo, H., Guo, Y., Yang, J., Liu, J., Li, Z., Zheng, T., . . . Zhang, B. (2023). Loglg: Weakly supervised log anomaly detection via log-event graph construction. In *International conference on database systems for advanced applications* (pp. 490–501).

Guo, H., Yuan, S., & Wu, X. (2021). Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Hamooni, H., Debnath, B., Xu, J., Zhang, H., Jiang, G., & Mueen, A. (2016). Logmine: Fast pattern recognition for log analytics. In *Proceedings of the 25th acm international on conference on information and knowledge management* (pp. 1573–1582).

Han, J., Pei, J., & Tong, H. (2022). *Data mining: concepts and techniques*. Morgan kaufmann.

Han, X., & Yuan, S. (2021). Unsupervised cross-system log anomaly detection via domain adaptation. In *Proceedings of the 30th acm international conference on information & knowledge management* (pp. 3068–3072).

He, M., Jia, T., Duan, C., Cai, H., Li, Y., & Huang, G. (2025). Weakly-supervised log-based anomaly detection with inexact labels via multi-instance learning. In *2025 ieee/acm 47th international conference on software engineering (icse)* (pp. 726–726).

He, P., Zhu, J., Zheng, Z., & Lyu, M. R. (2017). Drain: An online log parsing approach with fixed depth tree. In *2017 ieee international conference on web services (icws)* (pp. 33–40).

He, S., Lin, Q., Lou, J.-G., Zhang, H., Lyu, M. R., & Zhang, D. (2018). Identifying impactful service system problems via log analysis. In *Proceedings of the 2018 26th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 60–70).

Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H., & Luan, Z. (2020). Hitanomaly: Hierarchical transformers for anomaly detection in system log. *IEEE transactions on network and* *service management*, *17*(4), 2064–2076.

Inc., B. (n.d.). *Real-world applications for remarkable innovations [use cases].* Retrieved 2024-10-28, from https://buspas.com/

Joulin, A. (2016). Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*.

Le, V.-H., & Zhang, H. (2022). Log-based anomaly detection with deep learning: How far are we? In *Proceedings of the 44th international conference on software engineering* (pp. 1356–1367).

Liang, Y., Zhang, Y., Xiong, H., & Sahoo, R. (2007). Failure prediction in ibm bluegene/l event logs. In *Seventh ieee international conference on data mining (icdm 2007)* (pp. 583–588).

Lin, Q., Zhang, H., Lou, J.-G., Zhang, Y., & Chen, X. (2016). Log clustering based problem identification for online service systems. In *Proceedings of the 38th international conference on software engineering companion* (pp. 102–111).

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413–422).

Lou, J.-G., Fu, Q., Yang, S., Xu, Y., & Li, J. (2010). Mining invariants from console logs for system problem detection. In *2010 usenix annual technical conference (usenix atc 10).*

Lu, S., Wei, X., Li, Y., & Wang, L. (2018). Detecting anomaly in big data system logs using convolutional neural network. In *2018 ieee 16th intl conf on dependable, autonomic and secure computing, 16th intl conf on pervasive intelligence and computing, 4th intl conf on big data intelligence and computing and cyber science and technology congress (dasc/picom/datacom/cyberscitech)* (pp. 151–158).

Makanju, A. A., Zincir-Heywood, A. N., & Milios, E. E. (2009). Clustering event logs using iterative partitioning. In *Proceedings of the 15th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1255–1264).

Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., ... others (2019). Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Ijcai* (Vol. 19, pp. 4739–4745).

Nandi, A., Mandal, A., Atreja, S., Dasgupta, G. B., & Bhattacharya, S. (2016). Anomaly detection using program control flow graph mining from execution logs. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 215–224).

Nedelkoski, S., Bogatinovski, J., Acker, A., Cardoso, J., & Kao, O. (2021). Self-supervised log parsing. In *Machine learning and knowledge discovery in databases: Applied data science track: European conference, ecml pkdd 2020, ghent, belgium, september 14–18, 2020, proceedings, part iv* (pp. 122–138).

Nipa, N. A., Bouguila, N., & Patterson, Z. (2025a). A comparative study of log-based anomaly detection methods in real-world system logs. In *Proceedings of the 10th international conference on internet of things, big data and security - iotbds* (p. 141-152). SciTePress. doi: 10.5220/0013367000003944

Nipa, N. A., Bouguila, N., & Patterson, Z. (2025b). Log-based anomaly detection without ground truth: Evaluating weakly supervised, semi-supervised, and unsupervised deep learning approaches. In *Accepted for presentation at the 34th ieee international symposium on industrial electronics (isie), 2025.*

Pawar, Y., Zamzami, N., & Bouguila, N. (2020). An effective hybrid anomaly detection system based on mixture models. In *2020 international symposium on networks, computers and communications (isncc)* (p. 1-6).

Qi, J., Luan, Z., Huang, S., Fung, C., Yang, H., & Qian, D. (2023). Spikelog: log-based anomaly detection via potential-assisted spiking neuron network. *IEEE Transactions on Knowledge and Data Engineering*, *36*(12), 9322–9335.

Sghaier, O., Amayri, M., & Bouguila, N. (2023). Multivariate beta normality scores approach for deep anomaly detection in images using transformations. In *2023 ieee international conference on systems, man, and cybernetics (smc)* (p. 3428-3433).

Shima, K. (2016). Length matters: Clustering system log messages using length of words. *arXiv preprint arXiv:1611.03213*.

Tra, V., Amayri, M., & Bouguila, N. (2024). Unsupervised fault detection for building air handling unit systems using deep variational mixture of principal component analyzers. *IEEE Transactions on Automation Science and Engineering*, *21*(4), 6787-6803.

Xu, W., Huang, L., Fox, A., Patterson, D., & Jordan, M. I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the acm sigops 22nd symposium on operating systems principles* (pp. 117–132).

Yang, L., Chen, J., Wang, Z., Wang, W., Jiang, J., Dong, X., & Zhang, W. (2021). Semi-supervised log-based anomaly detection via probabilistic label estimation. In *2021 ieee/acm 43rd international conference on software engineering (icse)* (pp. 1448–1460).

Zhang, L., Li, W., Zhang, Z., Lu, Q., Hou, C., Hu, P., ... Lu, S. (2021). Logattn: Unsupervised log anomaly detection with an autoencoder based attention mechanism. In *International conference on knowledge science, engineering and management* (pp. 222–235).

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., ... others (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 807–817).

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019). Tools and benchmarks for automated log parsing. In *2019 ieee/acm 41st international conference on software engineering: Software engineering in practice (icse-seip)* (pp. 121–130).