

Ahmed Alwakeel

Explainable Transformer-Based Anomaly Detection for Large-Scale System Logs.docx

 Midocean University

Document Details

Submission ID

trn:oid:::2945:406764354

Submission Date

Aug 12, 2026, 12:10 AM GMT+3

Download Date

Aug 12, 2026, 12:12 AM GMT+3

File Name

Explainable Transformer-Based Anomaly Detection for Large-Scale System Logs.docx

File Size

3.2 MB

55 Pages

11,876 Words

78,329 Characters

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Match Groups

-  **75 Not Cited or Quoted 6%**
Matches with neither in-text citation nor quotation marks
-  **16 Missing Quotations 1%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 4%  Internet sources
- 3%  Publications
- 6%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- 75 Not Cited or Quoted 6%**
Matches with neither in-text citation nor quotation marks
- 16 Missing Quotations 1%**
Matches that are still very similar to source material
- 0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 4% Internet sources
- 3% Publications
- 6% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

- Submitted works**
The Robert Gordon University on 2026-08-03 <1%
- Internet**
arxiv.org <1%
- Submitted works**
University of East London on 2026-05-07 <1%
- Submitted works**
Midocean University on 2025-09-05 <1%
- Submitted works**
Thesis 2026 on 2026-02-11 <1%
- Submitted works**
Harrisburg University of Science and Technology on 2026-07-01 <1%
- Internet**
drepo.sdl.edu.sa <1%
- Submitted works**
American Public University System on 2023-12-19 <1%
- Internet**
www.mdpi.com <1%
- Submitted works**
Moodle2025 on 2026-08-10 <1%

11	Publication	Ocansey, Isaiah Thompson. "Log Anomaly Detection With Parameter-Efficient Tin...	<1%
12	Submitted works	University of Nottingham on 2026-05-14	<1%
13	Submitted works	Liverpool Hope on 2026-02-04	<1%
14	Submitted works	National Institute Of Technology, Tiruchirappalli on 2026-04-21	<1%
15	Internet	mdpi-res.com	<1%
16	Submitted works	The Robert Gordon University on 2026-04-26	<1%
17	Internet	publikasi.dinus.ac.id	<1%
18	Submitted works	Moodle2025 on 2026-08-06	<1%
19	Submitted works	Liverpool John Moores University on 2025-07-30	<1%
20	Internet	eprints.utm.my	<1%
21	Publication	Al-Marri, Noora Saeed M. J.. "Quantum Machine Learning for Intrusion Detection ...	<1%
22	Internet	digibug.ugr.es	<1%
23	Submitted works	INTI Universal Holdings SDM BHD on 2026-03-03	<1%
24	Internet	repository.embuni.ac.ke	<1%

25	Submitted works	University of Birmingham on 2008-05-01	<1%
26	Submitted works	Letterkenny Institute of Technology on 2025-08-29	<1%
27	Submitted works	University of Strathclyde on 2026-08-03	<1%
28	Submitted works	University of Warwick on 2026-08-09	<1%
29	Publication	V. Sekhar, P. Ajay Kumar Reddy, K. Logesh, T. Raghunadha Reddy, Dara Raju. "Ne...	<1%
30	Submitted works	College of Public Health on 2026-03-17	<1%
31	Publication	Keyuan Qiu, Zhejie Xu, Tao Luo, Meifang Yan, Ruru Liu, Pengjin Liu, Feng Chen. "F...	<1%
32	Submitted works	Math-Epi on 2026-08-06	<1%
33	Submitted works	Staffordshire University on 2026-06-30	<1%
34	Internet	ejournal.ubhara.ac.id	<1%
35	Internet	jutif.if.unsoed.ac.id	<1%
36	Internet	scikit-learn.org	<1%
37	Submitted works	Computer Science and Engineering on 2026-06-30	<1%
38	Publication	Kun Gong, Senlin Luo, Limin Pan, Linghao Zhang, Yifei Zhang, Haomiao Yu. "LogE...	<1%

39	Submitted works	University of Southampton on 2026-05-07	<1%
40	Internet	dspace.bracu.ac.bd	<1%
41	Internet	epubl.ktu.edu	<1%
42	Internet	repository.learn-portal.org	<1%
43	Publication	"Progress in Artificial Intelligence", Springer Science and Business Media LLC, 2026	<1%
44	Publication	Alam, Tanvir. "Bias Mitigation in Recommendation Systems: A Multi-Platform Ana...	<1%
45	Submitted works	Central Queensland University on 2025-11-11	<1%
46	Publication	Christensen, Erik R.. "A Framework for Characterising Performance in Multi-Class ...	<1%
47	Submitted works	FH Campus Wien on 2026-06-01	<1%
48	Submitted works	Glasgow Caledonian University on 2025-08-14	<1%
49	Publication	Hironori Uchida, Keitaro Tominaga, Hideki Itai, Yujie Li, Yoshihisa Nakatoh. "Impr...	<1%
50	Submitted works	Maastricht University on 2026-06-05	<1%
51	Submitted works	Malta College of Arts,Science and Technology on 2026-06-05	<1%
52	Submitted works	Moodle2025 on 2026-08-10	<1%

53	Submitted works	National Institute of Technology, Rourkela on 2026-05-15	<1%
54	Submitted works	New York Institute of Technology on 2026-05-05	<1%
55	Submitted works	Pioneer Academics on 2026-07-19	<1%
56	Submitted works	University of Leeds on 2022-08-10	<1%
57	Publication	Yucheng Xie, Ying Wang, Kai Feng, Le Chen, Jibin Wang. "STLAD: Sparse-Attention ...	<1%
58	Internet	mikenguyen13.github.io	<1%
59	Internet	repositorio.uach.mx	<1%
60	Publication	Junchen Ma, Yang Liu, Hongjie Wan, Guozi Sun. "Automatic Parsing and Utilizatio...	<1%
61	Submitted works	University of Westminster on 2026-04-17	<1%
62	Publication	Xiaowei Zhao, Kaiwei Guo, Mingting Huang, Shaojian Qiu, Lu Lu. "ELFA-Log: Cross...	<1%
63	Submitted works	Liverpool John Moores University on 2026-06-22	<1%
64	Submitted works	University of Surrey on 2010-09-22	<1%



Explainable Transformer-Based Anomaly Detection for Large-Scale System Logs

Prepared By

Name: Hesham Bashatah

ID NO.: 253001893

Name: Anas Barro

ID NO.: 253001086

Submitted in fulfillment of the requirements for the degree Master of
Science in Artificial Intelligence

At

College of Informatics

Midocean University

UAE

Supervisor Name:

Dr. Wesam Ahmed

Acknowledgements

We express our sincere appreciation to our supervisor, Dr. Wesam Ahmed, for his guidance, constructive criticism, and continued support throughout this research. We also acknowledge Midocean University and the College of Informatics for providing the academic environment required to complete this study. Finally, we thank our families and colleagues for their patience, encouragement, and support during the development, experimentation, and writing of this thesis.

Abstract

Large-scale computing systems continuously generate massive volumes of semi-structured system logs that provide critical information about operational events, warnings, failures, and component interactions. However, the increasing scale and complexity of modern computing environments make manual inspection and static rule-based monitoring inadequate, particularly when anomalies emerge from the semantic meaning and temporal relationships among multiple log events. Moreover, although deep learning approaches can improve anomaly detection, their opaque decision-making limits their transparency, auditability, and practical usefulness for operational investigation. This thesis proposes LogMiniLM-XAI, a lightweight, reproducible, and explainable Transformer-based framework for sequence-level anomaly detection in large-scale system logs. The proposed framework uses the BGL system-log dataset and applies deterministic log normalization followed by the construction of fixed, non-overlapping windows of ten consecutive events. The proposed framework achieves outstanding performance on the held-out chronological test partition, obtaining 99.8419% accuracy, 99.1973% precision, 98.1642% recall, 98.6780% F1-score, 99.9309% ROC-AUC, and 99.2511% PR-AUC. Beyond predictive performance, the thesis establishes an event-level explainability benchmark comparing SHAP and Event Occlusion under identical test sequences and replacement baselines. The two approaches demonstrate practically comparable faithfulness, while Event Occlusion achieves substantially higher repeated-run stability and baseline robustness and requires considerably less computational time. These findings demonstrate that effective log anomaly detection can be achieved using a compact Transformer architecture while simultaneously providing empirically evaluated and computationally practical event-level explanations. The proposed framework therefore provides a reproducible foundation for transparent anomaly monitoring in large-scale computing environments while maintaining a clear distinction between model-level explanation and operational root-cause analysis.

Keywords: system logs, anomaly detection, Transformer, MiniLM, explainable artificial intelligence, Event Occlusion, SHAP, BGL, faithfulness, stability.

Contents

Acknowledgements	2
Abstract	3
List of Figures	9
List of Tables	10
List of Abbreviations	11
Chapter 1: Introduction	12
1.1 Background and Motivation	12
1.2 Problem Statement	13
1.3 Research Questions	13
1.4 Research Objectives	13
1.5 Thesis Contributions	14
1.6 Scope and Boundaries	14
1.7 Thesis Organization	15
Chapter 2: Background and Related Work	16
2.1 Introduction	16
2.2 System Logs and Log Anomalies	16
2.2.1 Characteristics of System Logs	16
2.2.2 Types of Log Anomalies	16
2.3 Evolution of Log Anomaly Detection	16
2.4 Transformer-Based Sequence Modeling	17
2.4.1 Self-Attention	17
2.4.2 Positional Information	17
2.4.3 Advantages and Limitations	18
2.5 Semantic Representation of Log Messages	18
2.6 Window-Based Sequence Construction	18

2.7 Class Imbalance and Threshold Selection.....	19
2.8 Explainable Artificial Intelligence	19
2.8.1 Local Explanations	19
2.8.2 SHAP	19
2.8.3 Event Occlusion	20
2.8.4 Explanation Quality.....	20
2.9 Performance and Explainability Evaluation Metrics.....	20
2.9.1 Accuracy.....	21
2.9.2 Precision	21
2.9.3 Recall.....	21
2.9.4 F1-Score	21
2.9.5 True-Positive Rate.....	21
2.9.6 False-Positive Rate	21
2.9.7 ROC-AUC	22
2.9.8 Precision-Recall AUC	22
2.9.9 Explainability Evaluation Metrics.....	22
2.9.10 Comprehensiveness	22
2.9.11 Sufficiency	22
2.9.12 Decision-Flip Rate.....	22
2.9.13 Top-k Overlap	23
2.9.14 Jaccard Similarity	23
2.9.15 Spearman Rank Correlation	23
2.9.16 Repeated-Run Stability	23
2.9.17 Baseline Robustness.....	23
2.9.18 Attribution Concentration	23
2.9.19 Explanation Runtime.....	23

2.9.20 Cross-Method Agreement	24
2.9.21 Metric Interpretation in This Study	24
2.10 Related Work.....	24
2.10.1 Semantic and Transformer-Based Detection.....	24
2.10.2 Self-Supervised, Imbalance, and Uncertainty-Aware Methods	24
2.10.3 Transfer, Temporal, and Graph-Based Methods	25
2.10.4 Large Language Model and Data-Generation Methods	25
2.10.5 Operational and Explainable Detection.....	25
2.10.6 Comparative Synthesis	25
2.11 Research Gap.....	26
2.11.1 Lightweight Semantic Sequence Detection Gap	27
2.11.2 Reproducible Chronological Evaluation Gap	27
2.11.3 Objective Event-Level XAI Evaluation Gap.....	27
2.12 Chapter Summary.....	28
Chapter 3: Proposed Methodology.....	29
3.1 Methodological Overview	29
3.2 Dataset Description	30
3.3 Data Preprocessing	31
3.3.1 Loading and Label Extraction	31
3.3.2 Model-Input Construction and Normalization	31
3.3.3 Deterministic Event Mapping, Caching, and Reuse	32
3.4 Window Construction and Labeling	32
3.5 Temporal Dataset Split.....	34
3.6 MiniLM Semantic Encoding	35
3.7 Projection and Positional Representation.....	35
3.8 Lightweight Transformer Classifier	36

3.9 Class-Imbalance Handling	36
3.10 Training and Model Selection	36
3.11 Decision-Threshold Selection	37
3.12 SHAP Explanation Procedure	37
3.13 Event Occlusion Procedure	38
3.14 Unified Explanation Evaluation	38
3.15 Final Experimental Configuration.....	39
3.16 Algorithms.....	40
Algorithm 1: Detection Pipeline	40
Algorithm 2: Training and Selection Pipeline.....	40
Algorithm 3: XAI Evaluation Pipeline.....	40
3.17 Chapter Summary.....	41
Chapter 4: Experimental Results, Discussion, Conclusion, and Future Work.....	42
4.1 Introduction	42
4.2 Experimental Setup	42
4.3 Development Results.....	43
4.3.1 Preprocessing and Caching	43
4.3.2 Final Window Configuration	43
4.3.3 Imbalance Handling and Epoch Selection	44
4.3.4 Threshold Selection.....	44
4.4 Final Test Performance	44
4.5 Comparison with Baselines and Related Studies	45
4.6 Explainability Results	46
4.6.1 Common Benchmark Conditions	46
4.6.2 Faithfulness	47
4.6.3 Stability and Replacement-Baseline Robustness	47

4.6.4 Comparative XAI Decision	47
4.7 Computational Efficiency	48
4.8 Discussion	49
4.8.1 Contribution 1: Lightweight Semantic Transformer	49
4.8.2 Contribution 2: Reproducible Chronological Protocol	49
4.8.3 Contribution 3: Unified Event-Level XAI Benchmark	49
4.8.4 Answers to the Research Questions	49
4.9 Limitations and Threats to Validity	50
4.9.1 Internal Validity	50
4.9.2 Construct Validity	50
4.9.3 External Validity	50
4.9.4 Conclusion Validity	51
4.9.5 Operational Validity	51
4.10 Conclusion	51
4.11 Future Work	52
4.12 Chapter Summary	53
References	54

List of Figures

Figure 2.1 Conceptual development of log anomaly detection.

Figure 2.2 Window-based log sequence construction.

Figure 2.3 General event-level explainability methods for log anomaly detection.

Figure 3.1 Proposed LogMiniLM-XAI framework.

Figure 3.2 Log preprocessing and normalization flow.

Figure 3.3 Fixed non-overlapping event-window construction.

Figure 3.4 Chronological development, validation, and held-out test protocol.

Figure 3.5 Frozen MiniLM event encoding and reusable embedding cache.

Figure 3.6 LogMiniLM-XAI model architecture.

Figure 3.7 Unified SHAP and Event Occlusion evaluation workflow.

Figure 4.1 Evidence-based SHAP and Event Occlusion comparison.

List of Tables

Table 2.1. Modern related-work evidence core and comparison boundaries.

Table 2.2. Traceability from the research gaps to the proposed framework and evidence.

Table 3.1 BGL dataset and processed study counts.

Table 3.2 Final window-construction configuration.

Table 3.3 Final chronological dataset partitions.

Table 3.4 Unified XAI benchmark measures.

Table 3.5 Final experimental configuration.

Table 4.1 Final chronological dataset partitions.

Table 4.2 Final model and evaluation configuration.

Table 4.3 Validation threshold plateau and final selection.

Table 4.4 Confusion matrix on the held-out chronological BGL test partition.

Table 4.5 Final held-out detection results.

Table 4.6 Comparison tiers used in this thesis.

Table 4.7 Contextual comparison with representative published BGL studies.

Table 4.8 SHAP versus Event Occlusion benchmark results.

Table 4.9 Computational-efficiency evidence.

List of Abbreviations

ABBREVIATION MEANING

AI	Artificial Intelligence
AP	Average Precision
AUC	Area Under the Curve
BCE	Binary Cross-Entropy
BGL	Blue Gene/L
FN	False Negative
FP	False Positive
PR	Precision-Recall
ROC	Receiver Operating Characteristic
SHAP	SHapley Additive exPlanations
TN	True Negative
TP	True Positive
XAI	Explainable Artificial Intelligence

13

15

Chapter 1: Introduction

1.1 Background and Motivation

System logs are one of the most important sources of operational evidence in modern computing environments. They record events, warnings, errors, resource conditions, component states, and interactions among applications, services, operating systems, and infrastructure. In traditional systems, administrators could often inspect logs manually or construct rules around known failure signatures. That approach no longer scales to cloud, distributed, and service-oriented environments, where millions of events may be generated in a short period and failures may emerge from subtle changes in event order, frequency, or meaning. [1], [2]

The analytical challenge is not limited to identifying unusual individual messages. Many operational anomalies are sequence-level deviations: a message that is harmless in isolation may become suspicious when it follows a particular warning, appears at an unexpected frequency, or replaces an expected event. As log volume and complexity increased, detection methods consequently evolved from manual inspection and static rules toward learning-based approaches capable of modeling semantic and sequential behavior. [3], [4]

Deep learning has improved the capacity to model these patterns. Recurrent networks introduced trainable sequence modeling, while Transformer architectures enabled stronger contextual learning through self-attention. More recent work has used BERT-family models, semantic sentence encoders, generative Transformers, contrastive learning, graph reasoning, and large language models. Despite these advances, achieving strong detection performance together with computational efficiency, reproducible evaluation, and credible explanation remains a practical challenge. [5], [6], [7], [8]

This thesis addresses these limitations through LogMiniLM-XAI, a lightweight, reproducible, and explainable framework for practical log anomaly detection. Rather than increasing model size, the study investigates whether compact semantic representations combined with a lightweight Transformer can provide strong sequence-level performance. MiniLM encodes normalized events, a compact Transformer models cross-event context, and post-hoc methods identify the event positions that influence each anomaly decision. Model development follows a strict chronological training, validation, and testing protocol, while detection and explanation are evaluated as related but distinct stages. [9]

1.2 Problem Statement

Large-scale software systems generate log data that are too voluminous, heterogeneous, and context-dependent for reliable manual analysis. Rule-based and conventional statistical approaches struggle when anomalies depend on semantic similarity or relationships among events in a sequence. Deep neural models can improve detection, but their opaque decisions are difficult to audit and may provide insufficient evidence for operator investigation. A practical solution must therefore combine accurate and efficient sequence-level detection with leakage-resistant temporal evaluation and event-level explanations that are faithful, stable, robust, and computationally feasible. [1], [4]

The research problem is therefore defined as follows: how can a lightweight Transformer-based framework detect anomalous sequences in large-scale system logs under a reproducible chronological evaluation protocol while producing faithful, stable, robust, and computationally practical explanations of the event positions that influence its decisions?

1.3 Research Questions

1. How effectively can the proposed LogMiniLM-XAI classifier detect anomalous windows in the held-out chronological BGL test partition?
2. To what extent can compact MiniLM semantic representations support sequence-level log anomaly detection within the proposed framework?
3. How effectively does the combined preprocessing, chronological splitting, class-imbalance handling, and validation-based threshold-selection protocol support reproducible anomaly detection without test-set leakage?
4. Can SHAP and Event Occlusion identify the event positions that most strongly influence the model's anomaly decision?
5. How faithful, stable, robust, concentrated, and computationally efficient are the generated event-level explanations?

1.4 Research Objectives

- Prepare and normalize the BGL dataset while preserving chronological order and reproducible processing.
- Construct fixed, non-overlapping event windows and derive window-level anomaly labels.
- Investigate the effectiveness of frozen MiniLM embeddings as compact semantic representations for normalized log events.

- Design and evaluate a compact Transformer classifier for sequence-level anomaly detection.
- Evaluate a chronological training, validation, and testing protocol that prevents test-set leakage.
- Assess class-imbalance handling and validation-based threshold selection within the complete detection pipeline.
- Generate event-position explanations using SHAP and Event Occlusion.
- Compare the two XAI methods using faithfulness, stability, robustness, concentration, rank agreement, and runtime.
- Identify an evidence-based primary and complementary explanation strategy for the completed benchmark.
- Document the limits separating model explanation from causal root-cause analysis.

1.5 Thesis Contributions

- A lightweight semantic Transformer framework that combines frozen MiniLM event representations with a compact sequence classifier for large-scale log anomaly detection.
- A reproducible chronological evaluation protocol integrating deterministic preprocessing, fixed-window construction, temporal data partitioning, class-imbalance handling, and validation-only threshold selection.
- A unified event-level XAI benchmark that compares SHAP and Event Occlusion on identical cases using complementary faithfulness, stability, robustness, concentration, agreement, and runtime measures, leading to an evidence-based recommendation.

1.6 Scope and Boundaries

The empirical **scope of this study is limited to** supervised binary **classification of the BGL dataset** using fixed, non-overlapping windows of ten events. MiniLM is used as a frozen semantic encoder, while a separate compact Transformer performs sequence classification under a chronological training, validation, and testing protocol. The study does not evaluate alternative window sizes, overlapping or time-based windows, online or streaming detection, continual learning, or adaptation to concept drift. The explanation stage is applied after detection and attributes the model output to event positions using SHAP and Event Occlusion; these explanations describe model sensitivity rather than causal operational root cause. Cross-dataset generalization, production deployment, controlled operator-usability evaluation, cross-service root-cause analysis, and the integration of metrics, traces, topology, or dependency graphs are outside the current scope. Published

comparisons are interpreted contextually because preprocessing and evaluation protocols differ across studies.

1.7 Thesis Organization

This thesis is organized into four chapters. Chapter 2 introduces the research background, problem, questions, objectives, contributions, and scope. Chapter 3 establishes the theoretical and empirical background, defines the evaluation metrics, reviews representative related work, and identifies the research gap. Chapter 4 presents the proposed methodology, including preprocessing, window construction, chronological splitting, semantic encoding, classifier architecture, training, threshold selection, and XAI evaluation. Chapter 5 reports the experimental results and integrates their discussion with the study limitations, conclusions, and future work.

Chapter 2: Background and Related Work

2.1 Introduction

This chapter establishes the theoretical and empirical foundation for the proposed framework. It first describes the characteristics of system logs and the main types of log anomalies. It then reviews the evolution of learning-based detection, Transformer sequence modeling, semantic representation, window construction, class imbalance, threshold selection, and explainable artificial intelligence. The chapter also defines all detection and explainability metrics used in the study, reviews representative research, and identifies the gap addressed by LogMiniLM-XAI.

2.2 System Logs and Log Anomalies

2.2.1 Characteristics of System Logs

System logs are typically semi-structured: a stable message pattern is combined with variable elements such as timestamps, node identifiers, process IDs, addresses, counters, and resource values. Logs are also sequential because event order carries operational meaning. Their volume is often very high, and anomalous events usually represent only a small fraction of the stream. These properties make log anomaly detection a large-scale, context-dependent, and imbalanced learning problem. [1], [2]

2.2.2 Types of Log Anomalies

An anomaly is an individual event that differs from expected behavior. A sequence anomaly occurs when the order or co-occurrence of otherwise common events is unusual. A contextual anomaly depends on the surrounding operational state, while a frequency anomaly appears when an event occurs too often or too rarely. Semantic anomalies are messages whose meaning deviates from normal behavior even when their surface syntax resembles common logs. [3], [4]

2.3 Evolution of Log Anomaly Detection

Early approaches relied on expert rules, signatures, statistical thresholds, clustering, principal component analysis, support vector machines, and isolation-based techniques. These methods remain useful in constrained settings but usually require manual feature design and cannot fully represent long event dependencies. Recurrent models improved sequential learning, and Transformer architecture later enabled parallel contextual modeling through self-attention.

Explainable methods now extend the objective beyond detection accuracy to include transparency, stability, and operational usefulness. [3], [5], [6], [10]

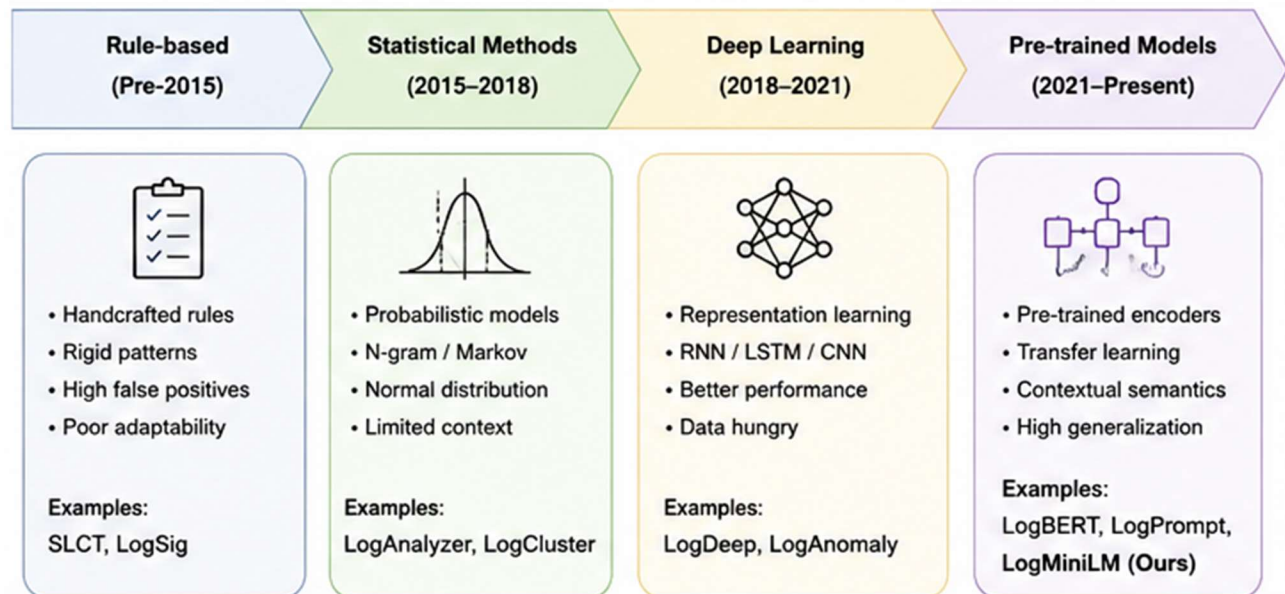


Figure 2.1 Conceptual development of log anomaly detection.

2.4 Transformer-Based Sequence Modeling

2.4.1 Self-Attention

Self-attention computes relationships among all event positions in a window. Each event is projected into query, key, and value vectors. Attention weights determine how strongly one event should use information from another. Multi-head attention allows the model to represent several relationship types simultaneously, including temporal dependency, repeated warning patterns, and interactions among components. [6], [11]

$$Attention(Q, K, V) = softmax(QK^T / \sqrt{d_k})V \quad (1)$$

2.4.2 Positional Information

Because self-attention does not inherently encode order, positional information must be added to the event representations. In log analysis this is essential because identical messages appearing in different positions may represent different operational states. [11]

2.4.3 Advantages and Limitations

Transformers support parallel training and capture long-range dependencies more directly than recurrent networks. Their limitations include memory cost, parameter count, sensitivity to preprocessing, and limited transparency. Compact architectures are therefore attractive when training resources are constrained or when repeated large-scale inference is required. [6], [11]

2.5 Semantic Representation of Log Messages

Event IDs and one-hot encodings represent identity but not meaning. Static embeddings introduce similarity but do not fully adapt to context. BERT-family encoders generate contextual representations, although full-size models can be computationally expensive. MiniLM is a compressed Transformer designed to preserve much of the representational capability of larger models at lower cost. In this study, MiniLM is not the final anomaly detector; it converts normalized events into dense semantic vectors that are consumed by a separate sequence classifier. [5], [7], [12], [13]

2.6 Window-Based Sequence Construction

A continuous log stream must be transformed into learning instances. A fixed window contains a constant number of consecutive events. If the step equals the window size, the windows do not overlap. A smaller step creates overlapping samples but increases computation and can produce leakage when neighboring windows are distributed across different partitions. The label of a window is commonly defined as anomalous when at least one event in the window is anomalous. This rule improves sensitivity but means that anomalous windows may still contain many normal events. [5], [10]

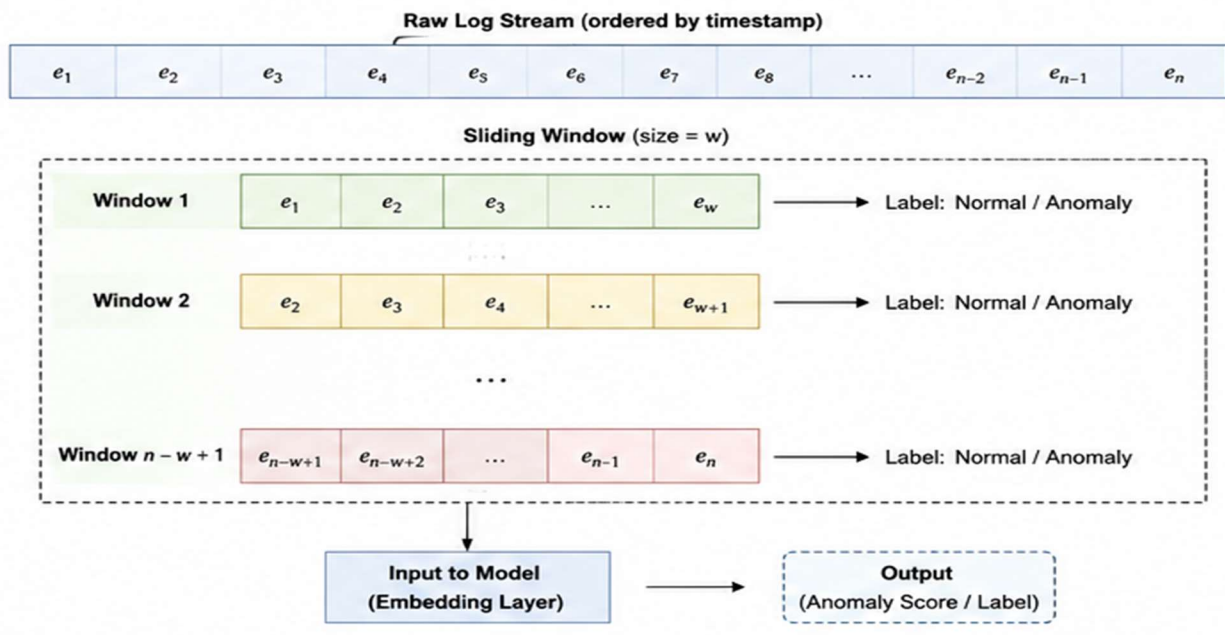


Figure 2.2 Window-based log sequence construction.

2.7 Class Imbalance and Threshold Selection

Normal windows dominate most system-log datasets. Accuracy can therefore appear high even when a model misses a large proportion of anomalies. Class-weighted loss, focal loss, over-sampling, under-sampling, and threshold tuning are common strategies. This study emphasizes class-weighted binary cross-entropy and validation-based threshold selection. The test set is reserved for final reporting and is not used to determine the operating threshold. [10], [14], [15]

2.8 Explainable Artificial Intelligence

2.8.1 Local Explanations

A local explanation describes why the model produced a particular output for one input. This is appropriate for anomaly detection because operators investigate individual alerts. Global explanations summarize general behavior but may obscure the details of a particular log sequence. [4], [16]

2.8.2 SHAP

SHAP assigns additive contribution values to features using Shapley-value principles. Positive values support the anomaly output and negative values oppose it. In event-level log explanation, each event position is treated as an interpretable feature. SHAP provides a distributional attribution

view but is approximate, computationally expensive, and sensitive to sampling and replacement baselines. [9]

2.8.3 Event Occlusion

Event Occlusion is a direct perturbation method. Each event is replaced by a defined baseline representation, and the model output is recomputed. The signed logit change indicates whether the event supports or opposes the anomaly decision, while the magnitude measures influence. Occlusion is deterministic for a fixed model, input, and baseline, although it requires repeated model inference. [4], [16]

2.8.4 Explanation Quality

Explanation quality should not be inferred from visual plausibility alone. It can be evaluated using comprehensiveness, sufficiency, decision-flip rate, repeated-run stability, robustness across replacement baselines, rank correlation, top-k concentration, and runtime. Cross-method agreement is useful as a descriptive measure, but disagreement does not determine which method is more faithful. [4], [8], [9]

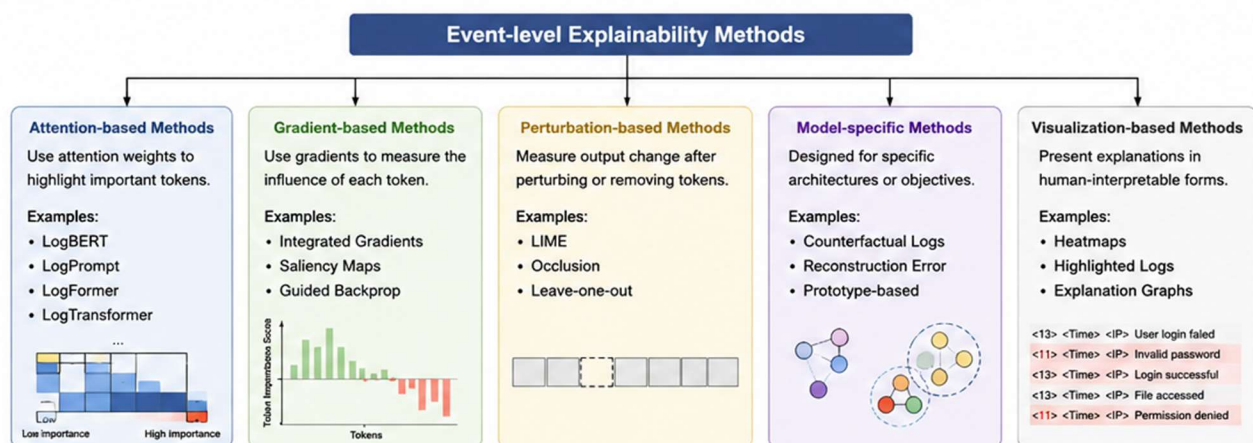


Figure 2.3 General event-level explainability methods for log anomaly detection.

2.9 Performance and Explainability Evaluation Metrics

The proposed framework is evaluated from two complementary perspectives. Detection performance is measured using classification and ranking metrics, while explanation quality is measured using faithfulness, stability, robustness, concentration, agreement, and computational cost. An anomalous window is treated as the positive class and a normal window as the negative class. [9], [14], [15]

2.9.1 Accuracy

Accuracy measures the proportion of correctly classified windows among all evaluated windows. It can be misleading under severe class imbalance and is therefore interpreted together with precision, recall, F1-score, and PR-AUC. [14], [15]

$$Accuracy = (TP + TN) / (TP + TN + FP + FN) \quad (2)$$

2.9.2 Precision

Precision measures the proportion of predicted anomalous windows that are actually anomalous. High precision indicates that the detector produces relatively few false alarms.

$$Precision = TP / (TP + FP) \quad (3)$$

2.9.3 Recall

Recalling measures the proportion of actual anomalous windows that are correctly detected. High recall indicates that relatively few anomalies are missing.

$$Recall = TP / (TP + FN) \quad (4)$$

2.9.4 F1-Score

The F1-score is the harmonic mean of precision and recall and is used as the primary model-selection metric in this study.

$$F1 = 2(Precision \times Recall) / (Precision + Recall) \quad (5)$$

2.9.5 True-Positive Rate

The true-positive rate is equivalent to recall and is used in constructing the ROC curve.

$$TPR = TP / (TP + FN) \quad (6)$$

2.9.6 False-Positive Rate

The false-positive rate measures the proportion of normal windows that are incorrectly classified as anomalous.

$$FPR = FP / (FP + TN) \quad (7)$$

2

2.9.7 ROC-AUC

ROC-AUC summarizes the relationship between the true-positive and false-positive rates across thresholds. A value near one indicates strong ranking ability, although ROC-AUC can appear optimistic under strong imbalance. [14], [15]

$$ROC - AUC = \int_0^1 TPR(FPR) d(FPR) \quad (8)$$

2.9.8 Precision-Recall AUC

PR-AUC focuses directly on the positive class and is especially informative for imbalanced datasets. Average precision is used as the numerical summary of the Precision-Recall curve. [14], [15]

27

$$AP = \sum_n (R_n - R_{n-1}) P_n \quad (9)$$

2.9.9 Explainability Evaluation Metrics

Let $z(X)$ denote the anomaly logit for log window X and $p(X) = \text{sigmoid}(z(X))$ the anomaly probability. Logit changes are emphasized because probabilities may become saturated near zero or one.

$$p(X) = 1 / (1 + \exp(-z(X))) \quad (10)$$

2.9.10 Comprehensiveness

Comprehensiveness evaluates whether the top-ranked events genuinely support the original anomaly decision. Let T_k be the top-k events. A larger positive value is better.

$$Comprehensiveness_k = z(X) - z(X \setminus T_k) \quad (11)$$

2.9.11 Sufficiency

Sufficiency evaluates whether the top-ranked events alone preserve the original decision. A smaller logit gap is better.

$$Sufficiency_k = |z(X) - z(X_{\setminus T_k})| \quad (12)$$

2.9.12 Decision-Flip Rate

The decision-flip rate measures how often removal of the top-k events changes the predicted class. It is treated as supporting evidence because highly confident predictions can remain unchanged despite a meaningful logit reduction.

$$DecisionFlip_k = (1/N) \sum_i I[y_i \neq y_i^{(-T_k)}] \quad (13)$$

58

2.9.13 Top-k Overlap

Top-k overlap measures the proportion of shared events between two top-k sets.

$$Overlap_k(A, B) = |T_k^A \cap T_k^B| / k \quad (14)$$

2.9.14 Jaccard Similarity

Jaccard similarity provides an alternative set-based agreement measure.

$$J(A, B) = |A \cap B| / |A \cup B| \quad (15)$$

2.9.15 Spearman Rank Correlation

Spearman correlation measures similarity between two complete event-importance rankings. Values range from -1 to 1.

$$\rho = 1 - [6 \sum_i d_i^2] / [n(n^2 - 1)] \quad (16)$$

2.9.16 Repeated-Run Stability

Repeated-run stability evaluates whether a method identifies the same events when applied repeatedly to the same input.

$$RepeatStability_k = [2 / R(R - 1)] \sum_{\{r < s\}} |T_k^r \cap T_k^s| / k \quad (17)$$

2.9.17 Baseline Robustness

Baseline robustness measures whether event rankings remain similar across different replacement baselines.

$$BaselineRobustness_k = [2 / B(B - 1)] \sum_{\{b < c\}} |T_k^b \cap T_k^c| / k \quad (18)$$

2.9.18 Attribution Concentration

Attribution concentration measures the proportion of total absolute attribution assigned to the top-k events. High concentration can improve readability but does not by itself prove faithfulness.

$$Concentration_k = \sum_{i \in T_k} |a_i| / \sum_j |a_j| \quad (19)$$

2.9.19 Explanation Runtime

Runtime measures the mean computational time required to explain one sequence.

$$MeanRuntime = (1/N) \sum_i t_i \quad (20)$$

2.9.20 Cross-Method Agreement

Cross-method agreement measures whether SHAP and Event Occlusion identify similar high-importance events. It is descriptive and is not used as the primary selection criterion.

$$Agreement_k = |T_k^{SHAP} \cap T_k^{Occ}| / k \quad (21)$$

2.9.21 Metric Interpretation in This Study

Detection is assessed primarily using F1-score, precision, recall, ROC-AUC, and PR-AUC; accuracy is reported as an additional overall measure. XAI methods are evaluated in the following order: faithfulness through comprehensiveness and sufficiency, repeated-run stability, robustness across replacement baselines, computational efficiency, attribution concentration, and descriptive cross-method agreement. When the central faithfulness difference between SHAP and Event Occlusion is below five percent, the methods are treated as practically comparable and the final choice is based on stability, robustness, and runtime. [9], [14], [15]

2.10 Related Work

2.10.1 Semantic and Transformer-Based Detection

Recent log anomaly detection research increasingly uses pretrained language models and Transformers to replace or complement symbolic event identifiers. LAnoBERT applies BERT masked-language modeling to parser-free anomaly detection, whereas LogFormer combines Transformer pretraining, adapters, and Log-Attention for cross-domain adaptation [5], [6]. LogELECTRA and ADALog also use self-supervised language-model objectives, but operate primarily at individual-log or token-reconstruction level rather than the supervised ten-event window classification used in this thesis [17], [18]. These studies support the value of semantic representations, but their anomaly units, supervision settings, and inference rules differ substantially from LogMiniLM-XAI.

2.10.2 Self-Supervised, Imbalance, and Uncertainty-Aware Methods

LogSD uses self-supervised dual-network learning and frequency-based masking to reduce the dominance of frequently occurring normal patterns [10]. LogEDL adds evidential deep learning to Transformer-based log modeling in order to quantify predictive uncertainty [19]. These approaches address important concerns—rare-event learning and uncertainty—but neither uncertainty estimates nor self-supervised reconstruction scores are equivalent to local event-position explanations. They are therefore treated as complementary research directions rather than direct XAI baselines.

35

2.10.3 Transfer, Temporal, and Graph-Based Methods

Cross-system studies extend anomaly detection beyond a single fixed environment. LogGT integrates BERT semantics, heterogeneous graph features, temporal intervals, and transfer learning, while LogETA emphasizes time-aware cross-system adaptation and inter-class boundary optimization [20], [21]. ELFA-Log applies enhanced pseudo-labeling and feature alignment to reduce the target-domain labeling requirement [22]. These studies demonstrate the importance of structural, temporal, and domain-shift information, but their cross-system objectives are outside the BGL-only scope validated in this thesis.

2.10.4 Large Language Model and Data-Generation Methods

Recent work has also explored substantially larger language models. LogLLM separates a semantic encoder, projection module, and Llama-based classifier, while LogLLaMA uses generative next-event modeling and reinforcement-learning-based anomaly identification [7], [23]. AnomalyGen uses large language models and program analysis to generate and label semantic log sequences for training and evaluation [24]. These studies expand the capabilities of log analysis, but also introduce different model scales, objectives, and training pipelines. Consequently, they motivate—rather than directly benchmark—the lightweight design adopted in LogMiniLM-XAI.

2.10.5 Operational and Explainable Detection

Operational and explanation-oriented studies reveal a further distinction between producing explanatory information and validating explanation quality. The ALICE O² framework combines BERT-based topic modeling, semi-supervised detection, and real-time visualization for operational monitoring [25]. TPLogAD separates template and dynamic-parameter representations, providing representation-level interpretability regarding different anomaly sources [26]. LLM-LADE jointly performs anomaly discrimination and natural-language explanation generation [8]. DeepEAD provides an earlier explicitly explainable system-log anomaly framework [4]. However, generated causes, topic keywords, representation decomposition, and post-hoc feature attributions answer different questions. None should be assumed faithful to a model decision without an explicit evaluation protocol.

2.10.6 Comparative Synthesis

Table 2.1 summarizes the fifteen-study modern evidence core. All studies are classified as published contextual comparisons because their preprocessing, anomaly unit, supervision, window construction, temporal partitioning, threshold selection, or metric implementation differs from the

protocol used in this thesis. Their published performance values are therefore not used to claim direct numerical superiority. The comparison instead identifies method families, evaluation boundaries, and the unresolved combination addressed by LogMiniLM-XAI. [5], [6], [7], [8], [10], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26].

Table 2.1. Modern related-work evidence core and comparison boundaries.

STUDY	YEAR / STATUS	METHOD	TASK / DATA	XAI	RELEVANCE AND BOUNDARY
lanobert [5]	2023 / Journal	BERT masked-language modeling	HDFS, BGL, Thunderbird; parser-free unsupervised	No objective event-level benchmark	Semantic BERT reference; objective and inference differ
logformer [6]	2024 / Conference	Pretraining, adapters, Log-Attention	Cross-domain log datasets; transfer learning	No event-level faithfulness benchmark	Compact adaptation reference; cross-domain protocol differs
logllm [7]	2024 / Preprint	BERT encoder, projector, Llama classifier	HDFS, BGL, Liberty, Thunderbird	No controlled event-level XAI	Related modular architecture, much larger scale
llm-lade [8]	2025 / Journal	LLM anomaly detection and explanation generation	Multiple benchmark datasets	Generated explanations	Closest modern detection-plus-explanation study; different evidence standard
logsd [10]	2024 / Research paper	Self-supervised dual networks and frequency masking	Three public datasets; semi-supervised	No matched SHAP-occlusion evaluation	Relevant to rare events and imbalance
logelectra [17]	2024 / Preprint	ELECTRA self-supervision	BGL, Spirit, Thunderbird; individual logs	No window-level XAI	Contrasts point anomalies with sequence classification
adalog [18]	2025 / Preprint	Adaptive MLM anomaly scoring	BGL, Thunderbird, Spirit	No matched event-window XAI	Relevant to semantics and adaptive thresholds
logedl [19]	2024 / Journal	Transformer with evidential head	HDFS, BGL, Thunderbird	Uncertainty, not local explanation	Distinguishes confidence from attribution
loggt [20]	2024 / Journal	BERT, heterogeneous graph, graph Transformer	Three systems; cross-system transfer	No matched attribution benchmark	Adds graph, component, and time information
logeta [21]	2024 / Journal	Time-aware attention and boundary optimization	Cross-system datasets	No event-level attribution	Relevant to temporal transfer outside current scope
elfa-log [22]	2025 / Journal	Pseudo-labeling and feature alignment	Cross-system benchmarks	No local XAI	Relevant to low-label adaptation and drift
logllama [23]	2025 / Preprint	LLaMA2 generation and reinforcement learning	BGL, Thunderbird, HDFS	No matched perturbation benchmark	Represents generative LLM direction
anomalygen [24]	2025 / Preprint	LLM-based semantic sequence generation	Hadoop/HDFS-oriented evaluation	Not an explanation method	Relevant to augmentation and data scarcity
alice o ² framework [25]	2025 / Journal	BERTopic, BERT embeddings, clustering	ALICE InfoLogger and BGL; real-time	Operational visualization	Adds deployment perspective; different task
tplogad [26]	2024 / Preprint	Template and parameter embeddings	Four public datasets; unsupervised	Representation decomposition	Highlights information in dynamic parameters

2.11 Research Gap

The reviewed literature demonstrates substantial progress in semantic representation, self-supervised learning, transfer learning, graph modeling, uncertainty estimation, large-language-model processing, deployment, and explanation generation. Nevertheless, the studies cannot be ranked directly because they differ in preprocessing, anomaly unit, supervision, sequence construction, temporal splitting, threshold selection, and metric implementation. The evidence therefore supports three bounded research gaps rather than a universal claim that prior work lacks accurate or explainable detection. [5], [6], [7], [8], [10], [20], [21]

2.11.1 Lightweight Semantic Sequence Detection Gap

Existing methods frequently emphasize either symbolic sequence modeling, self-supervised pretrained Transformers, cross-system graph architectures, or large language models. Event identifiers offer efficient sequence abstraction but do not directly represent semantic similarity between differently worded messages. Conversely, larger semantic, transfer, graph, and generative models can introduce additional encoding, adaptation, or inference complexity. There remains a practical need to evaluate a compact framework that separates frozen event-level semantic encoding from lightweight across-event sequence modeling. [5], [6], [7], [23]

2.11.2 Reproducible Chronological Evaluation Gap

The reviewed studies use heterogeneous definitions of events, sequences, labels, splits, thresholds, and performance measures. This heterogeneity makes published scores unsuitable for direct comparison and can make temporal leakage or test-set influence difficult to assess from incomplete reports. A transparent protocol is therefore needed in which preprocessing and fixed-window construction are deterministic, development and testing are separated chronologically, class weights are derived without test information, the operating threshold is selected using validation data only, and the held-out test set is used once for final reporting. [5], [6], [10], [20], [21]

2.11.3 Objective Event-Level XAI Evaluation Gap

Recent studies provide uncertainty estimates, graph or topic-level interpretation, representation decomposition, generated textual explanations, or post-hoc attribution. These outputs are useful but are not methodologically interchangeable. In particular, explanation plausibility does not establish faithfulness to the model output, and generated root-cause statements do not constitute causal validation. A controlled benchmark remains needed in which multiple event-level explanation methods are applied to the same detected cases and evaluated using explicit measures of faithfulness, repeated-run stability, replacement-baseline robustness, attribution concentration, cross-method agreement, and runtime. [4], [8], [19], [25], [26]

Accordingly, this thesis addresses three connected needs through LogMiniLM-XAI: (1) a lightweight semantic anomaly detector using frozen MiniLM event representations and a compact Transformer classifier; (2) a reproducible chronological BGL evaluation protocol with validation-only threshold selection and strict test isolation; and (3) an objective event-position XAI benchmark comparing SHAP and Event Occlusion under common cases, replacement baselines, and metrics.

The claims are limited to the BGL dataset, binary non-overlapping ten-event windows, and the completed experimental protocol.

Table 2.2. Traceability from the research gaps to the proposed framework and evidence.

RESEARCH GAP	RESPONSE IN LOGMINILM-XAI	EVIDENCE LOCATION
g1: lightweight semantic sequence detection	Frozen all-MiniLM-L6-v2 embeddings, 128-dimensional projection, one-layer compact Transformer	Chapter 3 architecture; Chapter 4 detection and model-size results
g2: reproducible chronological evaluation	Deterministic preprocessing, window size 10/step 10, chronological development/test split, validation-only threshold	Chapter 3 protocol; Chapter 4 threshold sweep and held-out test results
g3: objective event-level xai evaluation	SHAP and Event Occlusion on identical cases, three baselines, unified faithfulness, stability, robustness, agreement, and runtime metrics	Chapter 3 XAI design; Chapter 4 comparative XAI results

2.12 Chapter Summary

This chapter established the theoretical and empirical foundations for the proposed study. The reviewed literature shows that log anomaly detection has advanced from rule-based and statistical techniques to recurrent, Transformer-based, generative, transfer-learning, graph-based, uncertainty-aware, and semantically enriched methods. However, the evidence also reveals three connected needs: compact semantic sequence modeling, transparent chronological evaluation with validation-only threshold selection, and rigorous event-level assessment of post-hoc explanations. These findings motivate LogMiniLM-XAI, which combines frozen MiniLM event representations with a compact Transformer classifier and evaluates SHAP and Event Occlusion as model-decision explanation methods rather than causal root-cause mechanisms. Chapter 3 describes the methodology used to implement and evaluate this framework. [5], [6], [7], [8], [10], [20]

Chapter 3: Proposed Methodology

Introduction

3.1 Proposed Framework

The proposed LogMiniLM-XAI methodology is organized as a reproducible sequence of data preparation, fixed-window construction, semantic event encoding, lightweight across-event classification, validation-based model and threshold selection, held-out testing, and post-hoc explanation evaluation as shown in Figure 3.1. Chronologically ordered BGL records are normalized while retaining the original event messages and selected metadata for reporting. Unique normalized events are encoded once using the frozen all-MiniLM-L6-v2 model, after which a trainable projection and compact Transformer classifier model relationships across ten consecutive event positions.

Development data are used for parameter optimization, epoch selection, and operating-threshold selection, while the held-out test partition is reserved for final reporting. SHAP and Event Occlusion then explain the anomaly logit at event-position level on a common benchmark sample and are compared using faithfulness, repeated-run stability, replacement-baseline robustness, attribution concentration, cross-method agreement, and runtime. The explanation outputs describe model sensitivity and are not treated as causal operational root-cause evidence. Configurations, model checkpoints, predictions, figures, tables, and XAI artifacts are saved to support reproducibility.

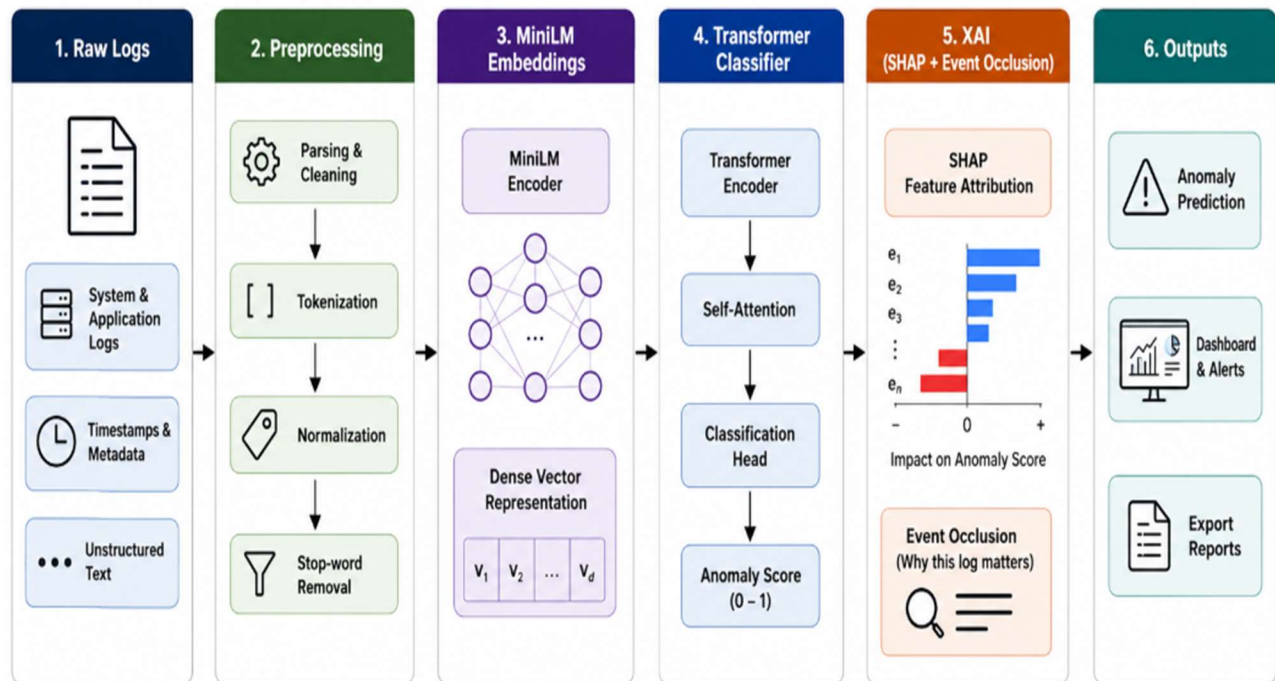


Figure 3.1 Proposed LogMiniLM-XAI framework.

3.2 Dataset Description

The Blue Gene/L (BGL) dataset contains chronologically ordered system logs collected from a large supercomputer environment. Each record includes a raw anomaly label and a structured sequence of operational fields followed by a textual event message. In the final implementation, the parser extracts the raw label, timestamp information, node, source, component, severity, and message text when these fields are available. A raw label of “-” is mapped to the normal class, while any other label is mapped to the anomalous class. [2]

The final preprocessing pipeline produced 4,713,493 usable log records, comprising 4,365,033 normal events and 348,460 anomalous events. After normalization, these records corresponded to 47,012 unique normalized messages. Preserving the original order, the records were grouped into 471,349 non-overlapping windows of ten consecutive events, of which 432,326 were normal and 39,023 were anomalous. BGL is appropriate for this study because it is large, chronologically ordered, strongly imbalanced, line-labeled, and widely used in log-anomaly research. Nevertheless, all detection and explanation conclusions in this thesis are bounded to BGL and the implemented binary ten-event window task.

يوضع في الجزء الذي بعد كده

Table 3.1 BGL dataset and processed study counts.

PROPERTY	VALUE
dataset	Blue Gene/L
processed usable lines	4,713,493
normal lines	4,365,033
anomalous lines	348,460
unique normalized messages	47,012
window size	10 events
step size	10 events
total windows	471,349
normal windows	432,326
anomalous windows	39,023

3.3 Data Preprocessing

3.3.1 Loading and Label Extraction

The BGL file is read sequentially so that the original temporal order is preserved. Each line is split into whitespace-separated fields, and records containing fewer than ten fields are excluded using a deterministic parser rule. File decoding uses an ignore-on-error policy, which prevents interruption by undecodable characters but may omit unsupported byte sequences. A raw label of “-” is mapped to the normal class, while every other raw label is mapped to the anomalous class.

3.3.2 Model-Input Construction and Normalization

The model input is constructed from the source, component, severity, and event-message fields. Structured timestamps, dates, node identifiers, Unix timestamps, and labels are excluded from the model representation to reduce direct shortcut learning. The resulting text is converted to lowercase and normalized using an ordered set of regular expressions. IPv4 addresses, date strings, time strings, hexadecimal values, BGL-like node identifiers occurring inside the text, and standalone integers are replaced by the typed placeholders <ip>, <date>, <time>, <hex>, <node>, and <num>, respectively. Remaining punctuation is converted to spaces and repeated whitespace is collapsed.

The original event message and selected parsed metadata are retained for alert and XAI reporting, while the complete raw-line string is not duplicated in the lightweight metadata artifact. The implementation does not contain a dedicated session-identifier rule, and no such normalization is claimed.

3.3.3 Deterministic Event Mapping, Caching, and Reuse

After normalization, unique normalized texts are sorted and assigned deterministic integer event IDs. Every parsed log line is mapped to its corresponding event ID, and line-level labels are stored separately. This procedure produces 47,012 unique normalized messages, the unique-text vocabulary, line-level event-ID arrays, and line-level label arrays.

Semantic embeddings are generated once for the unique normalized messages and cached. The unique-text table, event IDs, labels, lightweight metadata, and embeddings are saved as reusable artifacts. The implemented cache check validates the number of stored rows against the current vocabulary size; it does not constitute full hash-based validation of the text content, encoder version, or normalization configuration.

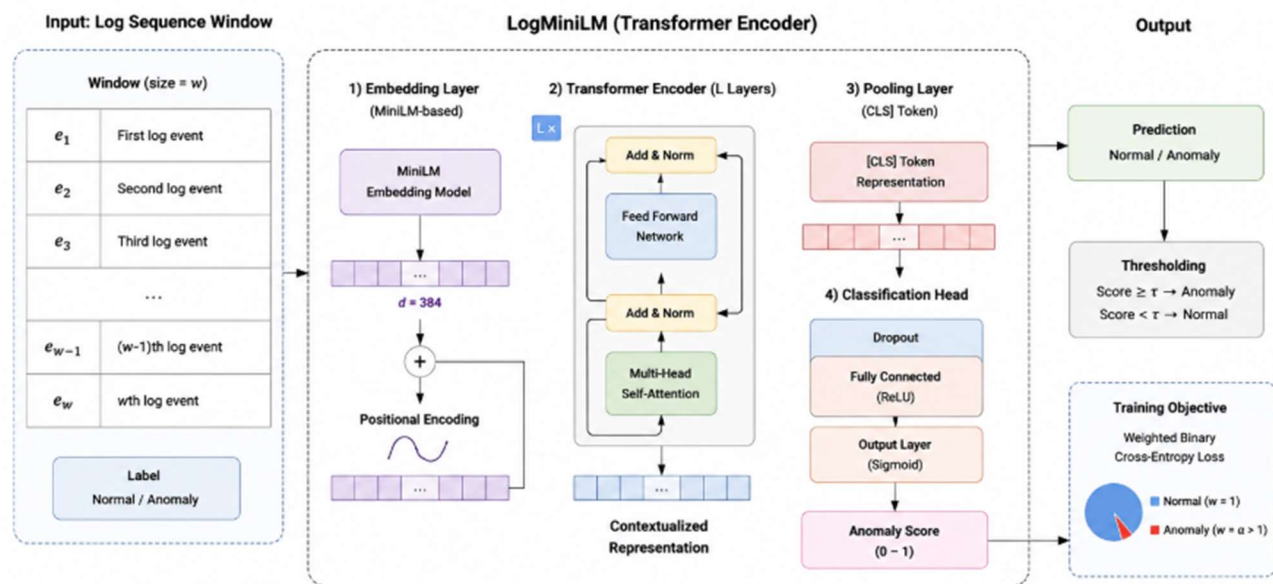


Figure 3.2 Log preprocessing and normalization flow.

3.4 Window Construction and Labeling

The normalized event stream is converted into fixed non-overlapping windows of ten consecutive log events. Let e_i denote the deterministic event ID for the i -th parsed record and $l_i \in \{0,1\}$ its binary line label. Starting at indices 0, 10, 20, and so forth, each complete window is defined as $X_s = [e_s, \dots, e_{s+9}]$. The window label is computed as $y_s = \max(l_s, \dots, l_{s+9})$; therefore, a window is labeled anomalous when at least one of its ten events is anomalous and normal only when all events are normal.

The step size equals the window size, so adjacent windows do not overlap and no log line is shared between them. Partial trailing windows are discarded rather than padded. From 4,713,493 parsed records, this procedure produced 471,349 complete windows, comprising 432,326 normal windows and 39,023 anomalous windows; the final three records were not included because they could not form a complete window. Each sequence is stored together with its inclusive start and end line indices for traceability.

Table 3.2 Final window-construction configuration

PROPERTY	CONFIGURATION
input unit	Parsed log event
sequence representation	Deterministic event IDs
window size	10 events
step size	10 events
overlap	0
padding	None
partial final window	Discarded
window-label rule	Anomalous if any event is anomalous
total complete windows	471,349
unused trailing records	3

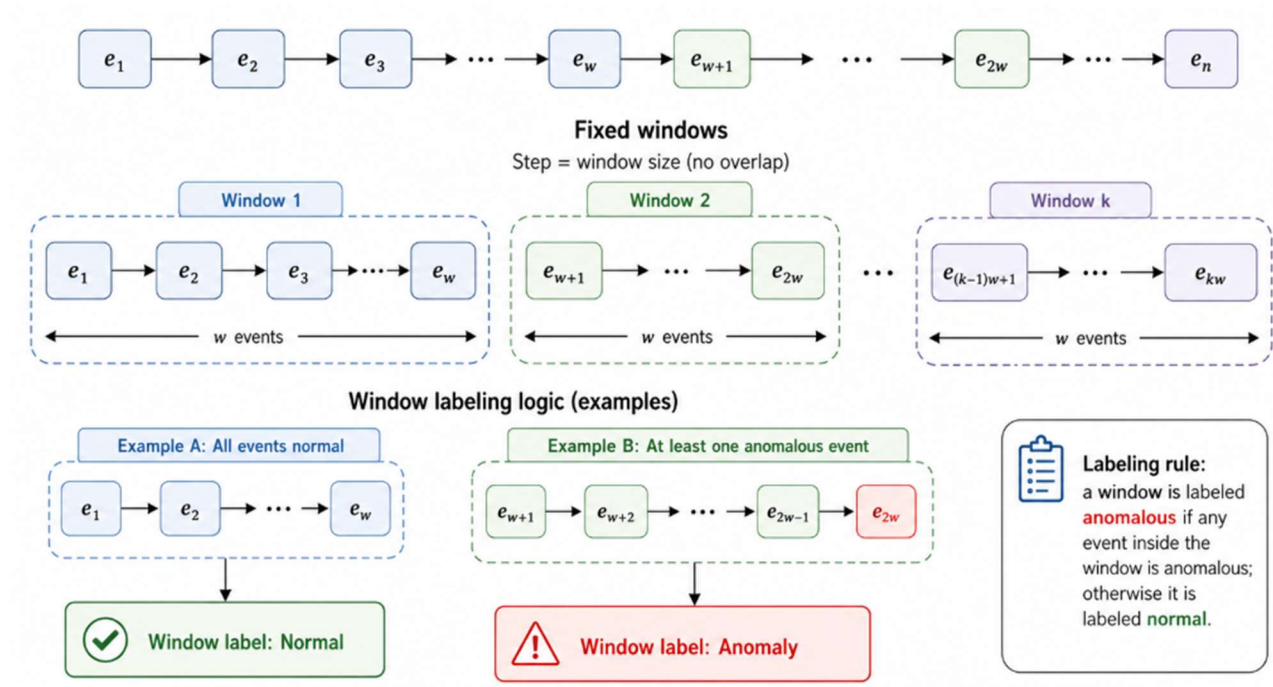


Figure 3.3 Fixed non-overlapping event-window construction.

3.5 Temporal Dataset Split

The complete chronological window stream is split only after window construction. The earliest 80% forms the development partition and the latest 20% forms the held-out test partition. Within the development partition, the earliest 90% is used for classifier training and the latest 10% for validation. No random shuffling is applied before partitioning.

The resulting development partition contains 377,079 windows, of which 339,372 are used during development training and 37,707 for validation. The held-out test partition contains 94,270 windows. Validation data are used for epoch selection and operating-threshold selection. After the best epoch is identified, the classifier is retrained on the full development partition, while the threshold selected from validation is frozen before test evaluation. The test partition remains untouched during model development and is used for final detection reporting and the XAI benchmark.

Class weights are derived without using test information. MiniLM embeddings may be computed and cached for unique normalized messages before classifier training because the encoder is frozen, does not use BGL labels, and is not optimized on the study dataset. Nevertheless, partition boundaries and all supervised model-selection decisions remain strictly chronological.

Table 3.3 Final chronological dataset partitions

PARTITION	WINDOWS	PURPOSE
development training	339,372	Optimize classifier parameters and monitor development
validation	37,707	Select epoch and decision threshold
final retraining partition	377,079	Retrain the selected architecture for the chosen number of epochs
held-out test	94,270	Final detection and XAI evaluation only

Chronological Development, Validation, and Held-Out Test Protocol

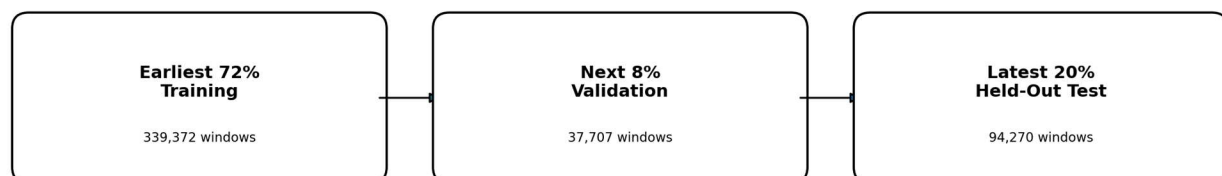


Figure 3.4 Chronological development, validation, and held-out test protocol.

3.6 MiniLM Semantic Encoding

Each unique normalized log message is encoded once using the frozen all-MiniLM-L6-v2 sentence-transformer model. Mean pooling is applied to produce a fixed 384-dimensional semantic embedding for each message. Because identical normalized messages reuse the same vector, only the 47,012 unique normalized messages are encoded, rather than all 4,713,493 parsed lines. [12], [13]

The MiniLM encoder is not fine-tuned on BGL and does not perform anomaly detection. Its role is to produce within-event semantic representations. Window-level anomaly discrimination is learned later by the trainable projection and compact Transformer classifier, which model relationships across event positions.

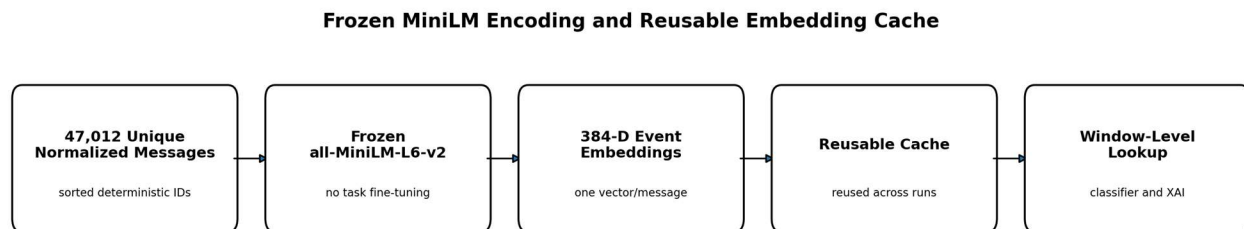


Figure 3.5 Frozen MiniLM event encoding and reusable embedding cache.

3.7 Projection and Positional Representation

A trainable linear projection maps every 384-dimensional MiniLM embedding to a 128-dimensional internal representation. This layer acts as a representation adapter between the frozen semantic encoder and the trainable sequence classifier. It reduces the parameter and memory cost of the subsequent attention and feed-forward operations without claiming to preserve all semantic information.

Learnable positional embeddings are added to the projected event representations so that the classifier can distinguish the ten positions in each window. This is necessary because self-attention alone is permutation-invariant.

3.8 Lightweight Transformer Classifier

The sequence classifier receives ten projected event vectors rather than raw text tokens. It consists of one Transformer encoder layer with two attention heads and a 256-dimensional feed-forward sublayer. Residual connections, layer normalization, and dropout support stable optimization. The architecture models contextual relationships across the ten-event window and does not claim unrestricted long-range sequence modeling.

Mean pooling aggregates the ten event-level outputs into one window representation. A linear classifier then produces the anomaly logit $z(X)$, and the sigmoid function converts this value to the anomaly probability $p(X) = 1 / (1 + \exp(-z(X)))$. The complete system contains 18,236,033 parameters, of which 183,425 are trainable because the MiniLM encoder remains frozen.

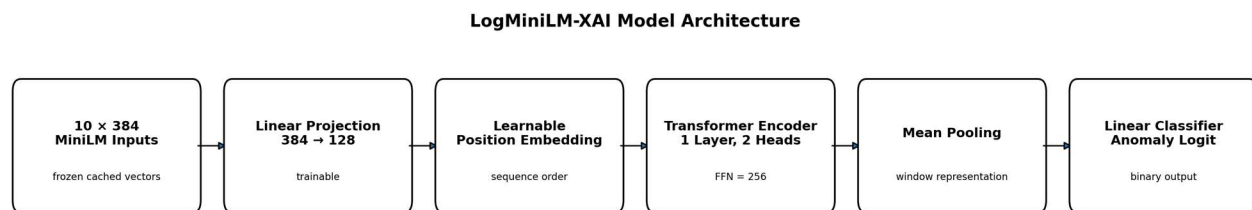


Figure 3.6 LogMiniLM-XAI model architecture.

3.9 Class-Imbalance Handling

Class imbalance is addressed using weighted binary cross-entropy. The anomaly-class weight is derived from the training distribution and multiplied by an anomaly-weight multiplier of 2.0. The resulting loss increases the penalty assigned to misclassified anomalous windows while preserving the original chronological sample order.

3.10 Training and Model Selection

The trainable projection and Transformer classifier are optimized using AdamW with mini-batches, dropout, fixed random seeds, and class-weighted binary cross-entropy. The frozen MiniLM encoder is excluded from gradient-based updates. Validation F1-score is the primary criterion for selecting the training epoch, with validation loss used only as a tie-breaker. [27]

Epoch 3 yields the highest validation F1-score and is therefore selected. A fresh classifier is then trained for exactly three epochs on the complete development partition. This procedure distinguishes development-time epoch selection from final retraining and avoids selecting the last epoch by default. Fixed seeds improve reproducibility, although exact bitwise determinism is not claimed for all GPU operations.

3.11 Decision-Threshold Selection

The decision threshold is treated as an inference-time parameter rather than a training hyperparameter. A threshold sweep is conducted exclusively on the validation partition after epoch selection. Thresholds from 0.10 to 0.30 produce the same maximum validation F1-score of 0.965767 and identical validation predictions. The selected threshold is 0.20, the tested value closest to the midpoint of this stable plateau.

The threshold is frozen before the held-out test set is accessed and is not modified after observing test results. This maintains test isolation and prevents post-hoc threshold tuning.

کل ده خاص بجزء واحد

مراجعة الاجزاء ودمجها بطريق صحيحه

3.12 SHAP Explanation Procedure

SHAP is applied after detection as a complementary event-position attribution method. Each of the ten event positions is treated as an interpretable feature, and the explained scalar output is the anomaly logit. Positive attribution values support the anomaly output, whereas negative values oppose it. [9]

The benchmark uses the same 50 held-out test sequences, three replacement baselines, three SHAP repeats, and 300 samples per explanation configuration. Repeated-run comparisons quantify stochastic stability, while cross-baseline comparisons quantify dependence on the replacement reference. SHAP explanations describe model sensitivity and do not identify the true operational cause of an anomaly.

3.13 Event Occlusion Procedure

Event Occlusion is evaluated on the same 50 test sequences and the same three replacement baselines. For event position j , the original embedding is replaced by the selected baseline representation, the anomaly logit is recomputed, and the signed attribution is calculated as $a_j = z(X) - z(X_{\text{masked}}(j))$. A positive value means that replacing the event reduced the anomaly evidence, while a negative value means that replacement increased it. [4], [16]

Each event position is perturbed separately without retraining the model. Event Occlusion is deterministic for a fixed model, input, inference mode, and baseline. It therefore provides a direct model-sensitivity measure, but it is not interpreted as causal root-cause evidence.

3.14 Unified Explanation Evaluation

The objective of the unified benchmark is not to determine which explanation method is universally superior, but to compare SHAP and Event Occlusion under an identical protocol. Both methods use the same classifier, held-out cases, replacement baselines, top-k values, and evaluation measures.

Faithfulness is assessed using comprehensiveness, sufficiency gap, and decision-flip evidence. Repeated-run top-k overlap measures stochastic stability. Cross-baseline top-k overlap and Spearman rank correlation measure replacement-baseline robustness. Attribution concentration summarizes how strongly importance is focused on the highest-ranked event positions. Cross-method agreement is descriptive and does not establish correctness. Runtime measures the mean time required to explain one sequence and excludes the one-time cached MiniLM embedding stage.

A five-percent practical-comparability rule is applied to the central faithfulness measures. When the relative difference between SHAP and Event Occlusion remains below this threshold, the methods are treated as practically comparable in faithfulness, and the final operational recommendation is based on stability, robustness, and computational efficiency.

Table 3.4 Unified XAI benchmark measures

MEASURE	DEFINITION	INTERPRETATION
comprehensiveness	Logit reduction after replacing top-k events	Higher positive values indicate stronger support for the original anomaly output
sufficiency gap	Absolute logit gap when only top-k events are retained	Lower values indicate that the selected events preserve more of the original decision
decision flip	Whether top-k replacement changes the predicted class	Supporting perturbation evidence
repeat stability	Top-k overlap across repeated explanations	Higher is better

baseline robustness	Top-k overlap and Spearman agreement across baselines	Higher is better
concentration	Top-k absolute attribution mass divided by total mass	Descriptive focus measure
cross-method agreement	Similarity between SHAP and Occlusion rankings	Descriptive only
runtime	Mean explanation time per sequence	Lower is better

Unified SHAP and Event Occlusion Evaluation Workflow

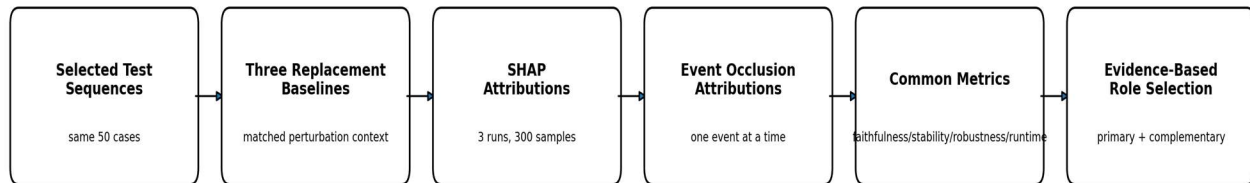


Figure 3.7 Unified SHAP and Event Occlusion evaluation workflow.

3.15 Final Experimental Configuration

Table 3.5 Final experimental configuration.

COMPONENT	FINAL CONFIGURATION
dataset	BGL system logs
processed usable lines	4,713,493
unique normalized messages	47,012
window method	Fixed, non-overlapping
window size / step	10 / 10
semantic encoder	sentence-transformers/all-MiniLM-L6-v2
encoder status	Frozen
semantic embedding dimension	384
projection dimension	128
positional representation	Learnable positional embeddings
transformer encoder layers	1
attention heads	2
feed-forward dimension	256
pooling	Mean pooling
trainable parameters	183,425
optimizer	AdamW
loss	Weighted binary cross-entropy
anomaly-weight multiplier	2.0
selected epoch	3
threshold selection	Validation sweep only
selected threshold	0.20
xai methods	SHAP and Event Occlusion
xai sample	50 held-out test sequences
replacement baselines	3
shap repeats	3

shap samples

| 300 per explanation configuration

3.16 Algorithms

Algorithm 1: Detection Pipeline

6. Read BGL records sequentially and extract structured fields and binary labels.
7. Construct the model text from source, component, severity, and event message.
8. Apply the ordered normalization rules and retain original messages and selected metadata.
9. Assign deterministic event IDs to sorted unique normalized messages.
10. Construct fixed non-overlapping ten-event windows and derive any-anomaly labels.
11. Split the window stream chronologically into development and held-out test partitions.
12. Encode unique normalized messages once using frozen MiniLM and cache the embeddings.
13. Project the embeddings, add positional representations, and apply the compact Transformer classifier.
14. Apply the frozen operating threshold to generate final predictions.

Algorithm 2: Training and Selection Pipeline

15. Divide the development partition chronologically into training and validation subsets.
16. Calculate class weighting from development training data only.
17. Train the projection and Transformer classifier with AdamW and weighted binary cross-entropy.
18. Select the best epoch using validation F1-score, with validation loss as a tie-breaker.
19. Retrain a fresh classifier for the selected number of epochs on the complete development partition.
20. Sweep thresholds on validation outputs only and select the representative midpoint of the best-F1 plateau.
21. Freeze the model configuration, selected epoch, and decision threshold before test evaluation.

Algorithm 3: XAI Evaluation Pipeline

22. Select the common benchmark sequences from the held-out test partition.
23. Compute each sequence's original anomaly logit.
24. Generate SHAP event-position attributions using the defined repeats, baselines, and sample budget.

25. Generate Event Occlusion attributions by replacing one event position at a time.
26. Compute faithfulness, stability, baseline robustness, concentration, agreement, and runtime.
27. Apply the practical-comparability rule to the central faithfulness measures.
28. Store operator-readable reports using the original event messages and selected metadata.

3.17 Chapter Summary

This chapter presented the complete LogMiniLM-XAI methodology. The first contribution is operationalized through frozen MiniLM event representations, a trainable 384-to-128 projection, and a compact one-layer Transformer classifier. The second contribution is implemented through deterministic preprocessing, fixed non-overlapping ten-event windows, chronological development and test separation, validation-based epoch and threshold selection, and strict held-out testing. The third contribution is implemented through a unified event-position benchmark that compares SHAP and Event Occlusion on identical cases and baselines using faithfulness, stability, robustness, concentration, agreement, and runtime.

The methodology therefore defines a single reproducible path from raw BGL logs to anomaly predictions, model-decision explanations, and saved experimental artifacts. Chapter 4 reports the resulting detection performance and XAI benchmark outcomes, evaluates the evidence supporting the three contributions, and discusses the limitations and implications of the completed study.

Chapter 4: Experimental Results and Discussion

4.1 Introduction

This chapter presents the empirical evidence generated by the completed LogMiniLM-XAI study. It first reports the finalized data partitions and implementation configuration, then documents the validation evidence used to select the training epoch and operating threshold. The held-out chronological BGL test results are subsequently presented using the confusion matrix, threshold-dependent classification metrics, and ranking metrics.

The chapter then distinguishes direct, reimplemented contextual, and published contextual comparisons, evaluates SHAP and Event Occlusion under a common event-level benchmark, and interprets the findings through the three thesis contributions. The discussion answers the research questions, identifies threats to validity, and closes the four-chapter thesis with a bounded conclusion and future-work agenda. Published results are treated contextually unless the full experimental protocol is equivalent, and explanation results are interpreted as evidence about model sensitivity rather than causal operational root cause.

4.2 Experimental Setup

The final experiment uses 471,349 fixed non-overlapping windows created from the chronologically ordered BGL stream. The earliest 80% forms the development partition and the latest 20% forms the held-out test partition. The development partition contains 377,079 windows; its earliest 339,372 windows are used for development training and its latest 37,707 windows for validation. The held-out test partition contains 94,270 windows.

Table 4.1 Final chronological dataset partitions

PARTITION	TOTAL WINDOWS	NORMAL	ANOMALOUS	PURPOSE
development training	339,372	Not separately exported	Not separately exported	Parameter optimization
validation	37,707	Not separately exported	Not separately exported	Epoch and threshold selection
final retraining partition	377,079	343,721	33,358	Fresh retraining for selected epoch count
held-out test	94,270	88,605	5,665	Final evaluation only

The final architecture uses frozen 384-dimensional all-MiniLM-L6-v2 event embeddings, a trainable 128-dimensional projection, one Transformer encoder layer, two attention heads, a 256-dimensional feed-forward sublayer, learnable positional embeddings, mean pooling, and a linear binary classifier. The full system contains 18,236,033 parameters, of which 183,425 are trainable.

Table 4.2 Final model and evaluation configuration.

COMPONENT	FINAL VALUE
semantic encoder	all-MiniLM-L6-v2, frozen
semantic embedding dimension	384
projection dimension	128
transformer encoder layers	1
attention heads	2
feed-forward dimension	256
pooling	Mean pooling
Trainable parameters	183,425
total parameters	18,236,033
loss	Weighted binary cross-entropy
anomaly-weight multiplier	2.0
selected epoch	3
selected threshold	0.20

Only settings verified by the final notebook or saved artifacts are reported. The thesis does not infer unrecorded hardware, package versions, learning-rate values, batch size, or other optimizer details.

4.3 Development Results

4.3.1 Preprocessing and Caching

Preprocessing produced 47,012 unique normalized messages from 4,713,493 usable log records. These unique messages were encoded once using frozen MiniLM and stored in a reusable cache. This reduced repeated semantic-encoder inference during classifier retraining and the explanation experiments. The cache check in the implementation validates the stored row count against the current vocabulary size; it does not provide full content- or configuration-hash validation.

4.3.2 Final Window Configuration

The final configuration uses windows of ten events with a step of ten. Adjacent samples therefore do not share raw log lines. This preserves local event order, reduces duplicated neighboring samples, lowers overlap-related leakage risk, and reduces computational cost. The ten-event configuration is the final implemented design; it is not claimed to be universally optimal because alternative window sizes and time-based windows were not included in the final controlled study.

4.3.3 Imbalance Handling and Epoch Selection

Class imbalance is handled during training using weighted binary cross-entropy. The anomaly-class weight is derived without test information and is multiplied by an anomaly-weight multiplier of 2.0. Validation F1-score is the primary epoch-selection criterion. Epoch 3 achieved the strongest validation result and was therefore selected. A fresh classifier was then retrained for exactly three epochs on the complete 377,079-window development partition.

4.3.4 Threshold Selection

After epoch selection, a decision-threshold sweep was conducted using validation outputs only. Thresholds from 0.10 to 0.30 produced identical validation predictions and the same maximum validation F1-score of 0.965767. Threshold 0.20 was selected as the tested value closest to the midpoint of this stable plateau. It was frozen before held-out test evaluation and was not adjusted after observing test results. This procedure is decision-threshold selection rather than probability calibration.

Table 4.3 Validation threshold plateau and final selection.

THRESHOLD RANGE	PRECISION	RECALL	F1-SCORE	DECISION
0.10-0.30	1.000000	0.9338	0.965767	Equal maximum plateau
0.20	1.000000	0.9338	0.965767	Selected midpoint representative

4.4 Final Test Performance

The frozen operating configuration was evaluated once on the latest 94,270 chronological BGL windows. The confusion matrix contains 88,560 true negatives, 45 false positives, 104 false negatives, and 5,561 true positives. These counts are the authoritative threshold-dependent test evidence.

Table 4.4 Confusion matrix on the held-out chronological BGL test partition.

ACTUAL / PREDICTED	NORMAL	ANOMALOUS	TOTAL
normal	88,560 (TN)	45 (FP)	88,605
anomalous	104 (FN)	5,561 (TP)	5,665
total	88,664	5,606	94,270

Table 4.5 Final held-out detection results.

METRIC	FINAL VALUE
accuracy	0.998419
precision	0.991973
recall	0.981642
f1-score	0.986780
roc-auc	0.999309
average precision (pr summary)	0.992511
selected threshold	0.20

The precision of 0.991973 indicates that most predicted anomaly windows were correct, while the recall of 0.981642 shows that most anomalous windows were detected. The model nevertheless produced 45 false positives and missed 104 anomalous windows. These errors remain important limitations despite the strong aggregate score. Accuracy is reported for completeness, while F1-score and Average Precision provide more informative positive-class summaries under the imbalanced test distribution. ROC-AUC and Average Precision are ranking metrics derived from saved probability outputs and cannot be reconstructed from the thresholded confusion matrix alone.

4.5 Comparison with Baselines and Related Studies

Comparison claims are divided into three evidence tiers. Tier 1 contains direct comparisons executed under the same preprocessing, windowing, label rule, chronological split, threshold policy, and metric definitions. Tier 2 contains locally reimplemented methods with material protocol differences. Tier 3 contains published results reported under study-specific protocols. Numerical superiority claims are permitted only for Tier 1.

Table 4.6 Comparison tiers used in this thesis.

TIER	DEFINITION	ALLOWED INTERPRETATION
tier 1 - direct	Same final BGL protocol and metrics	Numerical comparison allowed
tier 2 - reimplemented	Local implementation with material protocol differences	Contextual interpretation only
tier 3 - published	Original study values under study-specific protocols	No direct superiority claim
contextual		

The final Chapter 4 evidence does not contain a verified direct comparison table for the earlier LSTM, Transformer-classifier, or LogBERT-style experiments under the complete final protocol. Accordingly, unsupported placeholder rows are omitted from this reviewed version. LAnoBERT, LogGPT, LogFormer, and other published methods remain contextual references because their

preprocessing, objectives, sequence construction, supervision, thresholds, or test protocols differ. [5], [6], [10]

The strongest current empirical statement is that LogMiniLM-XAI achieved $F1 = 0.986780$ on the held-out chronological BGL test partition under the documented protocol. The result supports the integrated framework and does not isolate the causal contribution of MiniLM, the projection layer, the compact Transformer, class weighting, or the ten-event window. Matched ablation evidence would be required for such component-level conclusions.

Table 4.7 positions the proposed framework against representative BGL results reported in the original studies. The values are reproduced for contextual positioning only because the studies use materially different anomaly units, supervision settings, sequence lengths, preprocessing procedures, split policies, and threshold-selection rules. [5], [6], [10].

Table 4.7 Contextual comparison with representative published BGL studies.

STUDY	YEAR	BGL SETTING	PRECISION	RECALL	F1	OTHER METRIC	COMPARISON STATUS
lanobert [5]	2023	Parser-free, unsupervised MLM; test-optimized F1 threshold	Not reported	Not reported	0.8749	AUROC 0.9721	Tier 3 contextual
logformer [6]	2024	20-log non-overlapping windows; supervised source-domain model	0.9600	0.9700	0.9700	Not reported	Tier 3 contextual
logsd [10]	2024	100-log windows; self-supervised/semi-supervised protocol	0.9600	0.9664	0.9627	PRC 0.9716	Tier 3 contextual
logminilm-xai	Current	10-log non-overlapping windows; chronological held-out test	0.9920	0.9816	0.9868	AP 0.9925	Current study

Note: Published values are shown for contextual positioning and do not establish direct numerical superiority. LAnoBERT reports its best test-set F1 threshold under an unsupervised protocol; LogFormer uses 20-log sequences and source-domain supervised training; LogSD uses 100-log windows and a different self-supervised evaluation design. These settings are not equivalent to the validation-selected, chronological ten-event protocol used by LogMiniLM-XAI.

The reported values were verified against the original publications: LAnoBERT Table 4, LogFormer Table 2, and LogSD Table 2. The comparison is therefore traceable to primary sources while remaining explicitly contextual.

4.6 Explainability Results

4.6.1 Common Benchmark Conditions

SHAP and Event Occlusion were evaluated using the same trained classifier, the same 50 held-out test sequences, the same three replacement baselines, and the same event-position interpretation

unit. SHAP used three repetitions and 300 samples per explanation configuration. Event Occlusion replaced one event position at a time and recomputed the anomaly logit.

4.6.2 Faithfulness

SHAP achieved mean top-3 comprehensiveness of 0.465982 and mean top-3 sufficiency gap of 0.419759. Event Occlusion achieved corresponding values of 0.458191 and 0.422976. The symmetric relative differences are 1.686% for comprehensiveness and 0.763% for sufficiency gap, both below the predefined 5% practical-comparability boundary.

The approved interpretation is therefore that SHAP and Event Occlusion demonstrated practically comparable central faithfulness under the completed benchmark. SHAP is marginally stronger numerically on both central measures, but the differences are not treated as practically meaningful.

4.6.3 Stability and Replacement-Baseline Robustness

Event Occlusion achieved repeated-run top-3 stability of 1.000000, compared with 0.482222 for SHAP. Perfect repeated-run stability is expected for deterministic Occlusion under fixed model, input, inference, and baseline conditions. Cross-baseline robustness provides the stronger discriminator: Event Occlusion achieved top-3 robustness of 0.796667 and Spearman robustness of 0.720812, compared with 0.466667 and 0.234099 for SHAP.

SHAP's lower repeated-run stability is specific to the implemented Kernel SHAP sampling budget and is not generalized to all SHAP variants or parameter settings.

4.6.4 Comparative XAI Decision

Table 4.8 SHAP versus Event Occlusion benchmark results.

CRITERION	EVENT OCCLUSION	SHAP	INTERPRETATION
comprehensiveness (k=3)	0.458191	0.465982	Practically comparable
sufficiency gap (k=3)	0.422976	0.419759	Practically comparable
repeat top-3 stability	1.000000	0.482222	Occlusion stronger
baseline top-3 robustness	0.796667	0.466667	Occlusion stronger
baseline spearman robustness	0.720812	0.234099	Occlusion stronger
runtime per sequence	0.036889 s	0.766334 s	Occlusion faster
selected role	Primary	Complementary	Evidence-based assignment

Evidence-Based SHAP and Event Occlusion Comparison		
Faithfulness	Practically comparable	Δ comp. 1.686%; Δ suff. 0.763%
Repeat Stability	Occlusion stronger	1.000 vs 0.482
Baseline Robustness	Occlusion stronger	Top-3: 0.797 vs 0.467
Rank Robustness	Occlusion stronger	Spearman: 0.721 vs 0.234
Runtime	Occlusion faster	0.0369 s vs 0.7663 s

Final role assignment: Event Occlusion = Primary; SHAP = Complementary

Figure 4.1 Evidence-based SHAP and Event Occlusion comparison.

4.7 Computational Efficiency

MiniLM embedding extraction is a one-time cacheable preprocessing cost over 47,012 unique normalized messages. During classifier training and explanation benchmarking, the frozen semantic representations are reused. Only 183,425 classifier parameters are trainable.

Event Occlusion requires an average of 0.036889 seconds per explained sequence, whereas SHAP requires 0.766334 seconds. SHAP is therefore approximately 20.77 times slower in the completed benchmark. This ratio applies only to the implemented explanation configurations and excludes the cached MiniLM embedding stage.

Table 4.9 Computational-efficiency evidence.

STAGE OR METHOD	EXECUTION MODE	MEASURED VALUE	INTERPRETATION
minilm embedding extraction	One-time, cacheable	47,012 unique messages	Front-loaded cost
trainable classifier	Development training	183,425 parameters	Compact trainable head
event occlusion	Per explained sequence	0.036889 s	Primary method
shap	Per explained sequence	0.766334 s	Complementary method

4.8 Discussion

4.8.1 Contribution 1: Lightweight Semantic Transformer

The held-out results support the effectiveness of the integrated lightweight semantic detector. Frozen MiniLM provides reusable event-level semantic representations, while the trainable projection and compact Transformer model relationships across the ten-event window. The final classifier achieved $F1 = 0.986780$ with only 183,425 trainable parameters. This supports the practical value of the integrated design, but no matched ablation isolates the independent causal contribution of MiniLM, the projection layer, or the Transformer.

4.8.2 Contribution 2: Reproducible Chronological Protocol

The study preserved chronological order, constructed deterministic non-overlapping windows, separated development from the latest held-out test period, selected Epoch 3 and threshold 0.20 using validation evidence only, and froze all supervised decisions before test reporting. This provides a transparent leakage-resistant evaluation chain. The selected threshold should be understood as a stable validation operating point rather than a universally optimal probability cutoff.

4.8.3 Contribution 3: Unified Event-Level XAI Benchmark

The benchmark produced a mixed but actionable result. SHAP was marginally stronger on the two central raw faithfulness measures, but the differences remained below the 5% practical boundary. Event Occlusion was substantially more stable, more robust across replacement baselines, and faster. Event Occlusion is therefore selected as the primary explanation method and SHAP is retained as a complementary attribution perspective.

This recommendation is stronger than selecting one method by convention because it is tied to an explicit common protocol. Nevertheless, both methods explain sensitivity of the trained anomaly logit to event-position perturbations and do not establish operational root cause.

4.8.4 Answers to the Research Questions

RQ1: LogMiniLM-XAI achieved 0.991973 precision, 0.981642 recall, 0.986780 F1-score, 0.999309 ROC-AUC, and 0.992511 Average Precision on the held-out chronological BGL test partition, with 45 false positives and 104 false negatives.

RQ2: The integrated classifier using frozen 384-dimensional MiniLM event embeddings, a 128-dimensional projection, and a compact Transformer achieved strong held-out performance with

63

17

1

1

only 183,425 trainable parameters. This supports MiniLM's usefulness within the pipeline but does not isolate its independent causal contribution.

RQ3: The protocol preserved chronology, used deterministic non-overlapping windows, derived model-development decisions from development data only, selected Epoch 3 and threshold 0.20 on validation evidence, and reserved the latest 20% for held-out testing.

RQ4: Both SHAP and Event Occlusion generated event-position attribution rankings tied to the anomaly logit. They identify influential event positions under the defined perturbation procedures but do not establish the true operational cause.

RQ5: The two methods were practically comparable on central faithfulness. Event Occlusion was more stable, more robust across replacement baselines, and approximately 20.8 times faster, supporting its role as the primary explanation method.

4.9 Limitations and Threats to Validity

4.9.1 Internal Validity

- The final score reflects an integrated pipeline; no matched ablation isolates each component.
- Threshold selection depends on the validation partition.
- Event replacement may create uncommon or semantically artificial sequences.
- Kernel SHAP results depend on the implemented sampling budget.
- Cache validation checks row count rather than a complete text/configuration hash.

4.9.2 Construct Validity

- A window is labeled anomalous when at least one event is anomalous, so many events in an anomalous window may still be normal.
- Comprehensiveness and sufficiency measure faithfulness to model output, not operational causality.
- Event-position attribution is a proxy for explanation quality.
- Average Precision is the numerical Precision-Recall summary and should not be confused with trapezoidal PR-AUC.

4.9.3 External Validity

- The empirical evaluation uses BGL only.

- Results may not generalize to HDFS, Thunderbird, enterprise logs, streaming systems, or changing environments.
- The fixed ten-event, non-overlapping window may not suit other systems.
- No cross-domain adaptation or concept-drift experiment was completed.
- The XAI benchmark uses 50 selected test sequences and three replacement baselines.

4.9.4 Conclusion Validity

- Published-study values remain contextual because protocols differ.
- No universal superiority claim is supported.
- Repeated end-to-end training runs and confidence intervals were not reported.
- Runtime values are environment-specific and apply only to the completed benchmark.

4.9.5 Operational Validity

- No controlled operator study was performed.
- Explanations were not tested for investigation-time reduction, trust, or decision quality.
- The framework does not perform multi-source root-cause analysis using metrics, traces, topology, or service dependencies.

4.10 Conclusion

This thesis addressed the problem of combining accurate system-log anomaly detection with computationally practical and objectively evaluated explanations. LogMiniLM-XAI integrates deterministic BGL preprocessing, fixed non-overlapping ten-event windows, frozen MiniLM event representations, a trainable 384-to-128 projection, and a compact one-layer Transformer classifier under a chronological development and held-out testing protocol.

On the held-out chronological BGL test partition, the framework achieved 0.991973 precision, 0.981642 recall, 0.986780 F1-score, 0.999309 ROC-AUC, and 0.992511 Average Precision, with 45 false positives and 104 false negatives. The classifier contained only 183,425 trainable parameters because MiniLM remained frozen and its semantic representations were cached.

The explainability contribution was evaluated through a controlled comparison rather than the untested adoption of a single explainer. SHAP and Event Occlusion demonstrated practically comparable central faithfulness. Event Occlusion was more stable, more robust across replacement baselines, and approximately 20.8 times faster in the completed benchmark. It was therefore

selected as the primary explanation method, while SHAP was retained as a complementary attribution method.

The study contributes: (1) a lightweight semantic Transformer framework for sequence-level log anomaly detection; (2) a reproducible chronological evaluation protocol with validation-only epoch and threshold selection; and (3) a unified event-level XAI benchmark using common cases, baselines, and evaluation measures. The conclusions are limited to BGL, the implemented binary ten-event task, the final chronological protocol, and the 50-sequence explanation benchmark. The study demonstrates that strong held-out anomaly-detection performance can be combined with compact trainable modeling and explicit evaluation of explanation quality while preserving a clear boundary between model-decision explanation and operational root-cause analysis.

4.11 Future Work

- Validate the complete protocol on HDFS, Thunderbird, enterprise application logs, and multiple systems.
- Repeat end-to-end training with multiple seeds and report confidence intervals.
- Conduct matched ablations for MiniLM, the projection layer, Transformer versus LSTM, class weighting, and window size.
- Evaluate alternative event-count windows, time-based windows, adaptive segmentation, and controlled overlap.
- Adapt or fine-tune the semantic encoder using log-domain or contrastive objectives.
- Expand the XAI benchmark to larger stratified samples, additional replacement baselines, grouped-event Occlusion, and faster SHAP approximations.
- Evaluate uncertainty-aware explanations and interaction effects between event positions.
- Conduct controlled operator studies measuring trust, investigation time, and alert-triage usefulness.
- Integrate logs with metrics, traces, topology, and service dependencies to support multi-source root-cause analysis.
- Investigate concept-drift monitoring, cross-domain transfer, and deployment-time threshold adaptation.

4.12 Chapter Summary

This chapter reported the completed LogMiniLM-XAI evidence and closed the four-chapter thesis. The framework achieved $F1 = 0.986780$ on the held-out chronological BGL test partition. SHAP and Event Occlusion were practically comparable in central faithfulness, while Event Occlusion was more stable, more robust across replacement baselines, and substantially more efficient; it was therefore selected as the primary explanation method. The findings support the three bounded contributions of a lightweight semantic detector, a reproducible chronological protocol, and a unified event-level XAI benchmark, while remaining limited to BGL, the implemented final protocol, and the completed explanation sample.

References

- [1] X. Ma et al., “Practitioners’ expectations on log anomaly detection,” *IEEE Transactions on Software Engineering*, 2025.
- [2] A. Oliner and J. Stearley, “What supercomputers say: A study of five system logs,” in *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, pp. 575–584, 2007.
- [3] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Computing Surveys*, vol. 41, no. 3, Art. no. 15, 2009, doi: 10.1145/1541880.1541882.
- [4] X. Wang, K. J. Kim, Y. Wang, T. Koike-Akino, and K. Parsons, “DeepEAD: Explainable anomaly detection from system logs,” *Mitsubishi Electric Research Laboratories, Tech. Rep.*, 2023.
- [5] Y. Lee, J. Kim, and P. Kang, “LAnoBERT: System log anomaly detection based on BERT masked language model,” *Applied Soft Computing*, vol. 146, Art. no. 110689, 2023, doi: 10.1016/j.asoc.2023.110689.
- [6] H. Guo et al., “LogFormer: A pre-train and tuning pipeline for log anomaly detection,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 1, pp. 135–143, 2024, doi: 10.1609/aaai.v38i1.27764.
- [7] W. Guan et al., “LogLLM: Log-based anomaly detection using large language models,” *arXiv:2411.08561*, 2024.
- [8] Z. Zhang, S. Li, L. Zhang, J. Ye, C. Hu, and L. Yan, “LLM-LADE: Large language model-based log anomaly detection with explanation,” *Knowledge-Based Systems*, vol. 326, Art. no. 114064, 2025, doi: 10.1016/j.knosys.2025.114064.
- [9] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [10] Y. Xie, H. Zhang, and M. A. Babar, “LogSD: Detecting anomalies from system logs through self-supervised learning and frequency-based masking,” *ACM Transactions on Software Engineering and Methodology*, 2024.
- [11] A. Vaswani et al., “Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [12] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained Transformers,” *arXiv:2002.10957*, 2020.
- [13] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proceedings of EMNLP-IJCNLP*, pp. 3982–3992, 2019.
- [14] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves,” in *Proceedings of ICML*, pp. 233–240, 2006.
- [15] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers,” *PLOS ONE*, vol. 10, no. 3, Art. no. e0118432, 2015.
- [16] M. T. Ribeiro, S. Singh, and C. Guestrin, “Why should I trust you? Explaining the predictions of any classifier,” in *Proceedings of ACM SIGKDD*, pp. 1135–1144, 2016.

- [17] Y. Yamanaka, T. Takahashi, T. Minami, and Y. Nakajima, “LogELECTRA: Self-supervised anomaly detection for unstructured logs,” arXiv:2402.10397, 2024.
- [18] P. Pospieszny, W. Mormul, K. Szyndler, and S. Kumar, “ADALog: Adaptive unsupervised anomaly detection in logs with self-attention masked language model,” arXiv:2505.13496, 2025.
- [19] Y. Duan et al., “LogEDL: Log anomaly detection via evidential deep learning,” *Applied Sciences*, vol. 14, no. 16, Art. no. 7055, 2024, doi: 10.3390/app14167055.
- [20] P. Wang, X. Zhang, Z. Cao, W. Xu, and W. Li, “LogGT: Cross-system log anomaly detection via heterogeneous graph feature and transfer learning,” *Expert Systems with Applications*, vol. 251, Art. no. 124082, 2024, doi: 10.1016/j.eswa.2024.124082.
- [21] K. Gong et al., “LogETA: Time-aware cross-system log-based anomaly detection with inter-class boundary optimization,” *Future Generation Computer Systems*, vol. 157, pp. 16–28, 2024, doi: 10.1016/j.future.2024.03.028.
- [22] “ELFA-Log: Cross-system log anomaly detection via enhanced pseudo-labeling and feature alignment,” *Computers*, vol. 14, no. 7, Art. no. 272, 2025, doi: 10.3390/computers14070272.
- [23] Z. Yang and I. G. Harris, “LogLLaMA: Transformer-based log anomaly detection with LLaMA,” arXiv:2503.14849, 2025.
- [24] X. Li et al., “AnomalyGen: An automated semantic log sequence generation framework with LLM for anomaly detection,” arXiv:2504.12250, 2025, doi: 10.48550/arXiv.2504.12250.
- [25] A. Techaviseschai, S. Tarnpradab, V. C. Barroso, and P. Phunchongharn, “A real-time semi-supervised log anomaly detection framework for ALICE O² facilities,” *Applied Sciences*, vol. 15, no. 11, Art. no. 5901, 2025, doi: 10.3390/app15115901.
- [26] J. Lu and C. Wu, “TPLogAD: Unsupervised log anomaly detection based on event templates and key parameters,” arXiv:2411.15250, 2024.
- [27] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *International Conference on Learning Representations*, 2019.