Intrusion Detection System using LAD

Bhaumikaditya Guleria and Maroti Deshmukh

Department of Computer Science and Engineering, NIT Uttarakahand, Srinagar India.

*Corresponding author(s). E-mail(s): marotideshmukh@nituk.ac.in;
Contributing authors: mt23cse003@nituk.ac.in;

Abstract

Intrusion Detection Systems (IDS) will continue to play a critical role in safeguarding networks from malicious activities in an increasingly connected world. Traditional IDS approaches are expected to face persistent challenges, including high false positive rates, inefficiency in handling evolving threats, and limited adaptability to diverse data distributions. This research will investigate the potential of Logical Analysis of Data (LAD) as a robust mathematical framework for enhancing intrusion detection. LAD will be employed to identify critical patterns in network traffic, leveraging its ability to handle heterogeneous and imbalanced datasets effectively. The study will design a LAD-based IDS model and evaluate its performance on widely recognized datasets, such as NSL-KDD and KDD-Cup99. Key performance metrics, including accuracy, precision, recall, F2-score, and data coverage, will be assessed to measure the effectiveness of the proposed model. It is anticipated that the LAD-based approach will demonstrate superior adaptability and accuracy compared to traditional methods, while also addressing issues of dataset coverage and computational efficiency. The findings of this study will provide valuable insights into the application of LAD in cybersecurity and will pave the way for the development of more robust, scalable, and adaptive IDS models. Future research will focus on refining the LAD methodology to address its limitations and further enhance its applicability to real-world intrusion detection scenarios.

Keywords: IDS, LAD, HybridIDS

1 Introduction

In recent years, the exponential growth of interconnected devices and complex network systems has increased the vulnerability of critical infrastructure and sensitive data to cyberattacks. This alarming trend has necessitated the development of robust cybersecurity mechanisms to ensure the safety and integrity of digital systems. Intrusion Detection Systems (IDS) have emerged as essential tools in this context, providing the capability to monitor network activities, identify potential threats, and mitigate risks by detecting malicious activities in real-time. IDS technologies are particularly significant in addressing the ever-evolving nature of cyber threats, which exploit sophisticated techniques to bypass traditional security measures [1, 2].

IDS can be broadly categorized into three main types on the basis of detection technique: signature-based, anomaly-based, and hybrid approaches. Signature-based systems operate by identifying known attack patterns and are highly efficient against previously documented threats. However, their limitation lies in their inability to detect novel or zero-day attacks. In contrast, anomaly-based IDS use statistical or machine learning models to detect deviations from normal behavior, making them more effective at identifying previously unseen attack patterns. Hybrid systems combine these two approaches to leverage their respective strengths and provide a more comprehensive intrusion detection solution [3, 4]. Recent advancements in IDS methodologies have increasingly relied on machine learning (ML) and deep learning (DL) techniques, enabling systems to enhance their detection accuracy, scalability, and adaptability to complex network environments [5–7].

The growing prevalence of the Internet of Things (IoT) further exacerbates the need for advanced IDS solutions. IoT environments are characterized by heterogeneous devices, constrained resources, and diverse communication protocols, which pose unique challenges for intrusion detection. Traditional IDS frameworks often fall short in meeting these demands, necessitating the development of IoT-specific IDS tailored to the distinctive requirements of these systems. Research in this domain has explored various innovative approaches, including the use of optimization algorithms, hybrid methods, and lightweight ML models, to improve the scalability, efficiency, and performance of IDS in IoT environments [7, 8].

Logical Analysis of Data (LAD) has emerged as a promising data mining methodology with applications in intrusion detection, classification, and pattern recognition. LAD leverages Boolean functions and combinatorial optimization techniques to generate interpretable patterns that distinguish between normal and malicious activities. While LAD has demonstrated significant potential in addressing the challenges of IDS, its application is often constrained by computational complexity and incomplete dataset coverage, especially when dealing with high-dimensional or large-scale datasets. Nonetheless, LAD's ability to generate interpretable and explainable rules positions it as a valuable tool for intrusion detection, particularly when combined with complementary machine learning models [9, 10].

In this study, we aim to investigate the effectiveness of LAD in the context of intrusion detection by applying it to two benchmark datasets: NSL-KDD and KDD-Cup99. Through rigorous experimentation and analysis, we evaluate LAD's performance

across various metrics, including accuracy, precision, recall, and F2-score. Additionally, we address the limitations of LAD, such as long training times and incomplete coverage, by proposing potential enhancements to its methodology. Furthermore, we highlight the trade-offs between rule interpretability and detection performance, with a focus on improving the scalability and applicability of LAD in real-world intrusion detection scenarios.

The contributions of this work are threefold: 1. A comprehensive evaluation of LAD's performance on intrusion detection datasets with varying characteristics. 2. Insights into the trade-offs and challenges associated with LAD-based IDS, including computational complexity and coverage limitations. 3. A proposed framework for integrating secondary models as fallback mechanisms to enhance detection accuracy and address coverage gaps.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work in IDS and LAD. Section 3 describes the experimental setup and LAD-based rule generation process. Section 4 presents the results of our experiments, followed by an analysis of their implications. Finally, Section 5 concludes the study with a discussion of future research directions.

2 Literature Analysis

The field of Intrusion Detection Systems (IDS) has witnessed significant advancements with the incorporation of modern computational techniques, including machine learning, deep learning, and Logical Analysis of Data (LAD). These innovations have improved the ability to detect and mitigate cyber threats in increasingly complex network environments, including the Internet of Things (IoT). This section reviews the key developments in IDS and LAD, highlighting state-of-the-art methodologies, their strengths, limitations, and areas of application. The aim is to provide a comprehensive understanding of the existing literature while identifying gaps and opportunities for further research.

2.1 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) have evolved significantly with the integration of machine learning and deep learning techniques. Signature-based IDS identify malicious activities by recognizing known patterns or signatures. Alhomoud et al. [5] conducted a performance evaluation of IDS techniques, highlighting the strengths and limitations of various methodologies. Kaur and Singh [6] employed deep recurrent neural networks for signature generation, achieving improved detection accuracy. On the other hand, anomaly-based IDS detect deviations from normal behavior to identify potential threats. Abbasi et al. [1] proposed a deep learning-based feature extraction mechanism optimized with a finite state machine, enhancing anomaly detection. Liu et al. [2] introduced a GA-GOGMM-based adaptive pattern learning technique, integrating fuzzy rough set-based attribute selection to improve IDS effectiveness. Chauhan et al. [7] explored logical analysis of data (LAD) and information gain ratio for intrusion detection in IoT systems.

Hybrid IDS combine signature and anomaly detection approaches, leveraging their streng-ths to improve effectiveness. Ugtakhbayar et al. [4] developed a hybrid model that enhances detection capabilities through this integration. Similarly, Kim et al. [11] proposed a hybrid method to reduce false positive rates by merging anomaly detection with misuse detection. Comprehensive surveys, such as those by Santhosh Kumar et al. [3], provide insights into machine learning-based IDS tailored for IoT environments, addressing unique challenges in secure communication. Khraisat et al. [12] reviewed IDS techniques, datasets, and challenges, offering future research directions, while Abdulganiyu et al. [13] focused on advancements in network IDS using machine learning models. IoT-specific IDS have gained attention due to the increasing prevalence of connected systems, with Sowmya and Anita [8] reviewing hybrid methods for improving scalability and performance. Furthermore, Ahmad et al. [14] and Dini et al. [15] explored machine learning and deep learning methods, enhancing IDS detection rates and reducing false positives.

2.2 Logical Analysis of Data (LAD)

Logical Analysis of Data (LAD) is a versatile methodology applied in various domains, including IDS and data classification. Boros et al. [9] introduced LAD as a method for analyzing numerical data using Boolean patterns, laying the groundwork for its applications. They further demonstrated its scalability for large-scale datasets [10]. Algorithmic advancements, such as the accelerated pattern detection algorithm proposed by Alexe and Hammer [16], have improved LAD's computational efficiency. Crama et al. [17] examined cause-effect relationships using Boolean functions, contributing to its theoretical foundations.

Applications of LAD span multiple domains. In IoT security, Chauhan et al. [18] demonstrated its potential in intrusion detection systems. Gangopadhyay et al. [19] explored LAD's utility in financial applications, including accounting analytics and fraud detection. Bonates et al. [20] refined LAD for maximum pattern detection in data mining tasks. Comprehensive surveys, such as the one by Lejeune et al. [21], review recent advancements in LAD, highlighting its versatility in data analytics and potential for further innovation.

3 Proposed Work

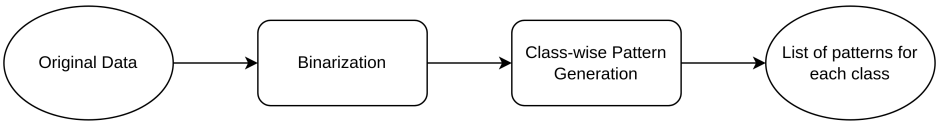

Fig. 1 LAD Process Flow

The main issue with most IDS systems is that they are resource intensive and logically ambiguous.

The proposed work is a modification of LAD which revolves around transforming data into binary format and generating rules in a Sum of Product format for the purpose of classification. The overall flow of the process is in stages:

- **Binarization:** Converting the data to binary format.
- **Pattern Generation:** Using binary data to make rules.

3.1 Binarization

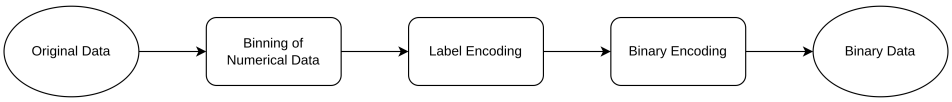

Fig. 2 Binarization Process

The binarization process that transforms every feature in the dataset into one or more binary features. This is done while aiming to encapsulate all the significant information present in them. All features in the dataset are binarized independently. Binarizing a numerical feature as is may produce too many binary features with minimum significance due to number of unique values being comparable to number of samples. So, many of these values should be collected into groups which is done by the process called binning.

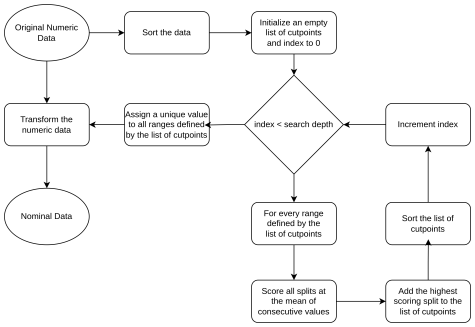

Fig. 3 Binning Process

In binning we transform continuous numerical data into various intervals. This process begins with sorting of the data. Then the arithmetic mean of all consecutive values is taken into consideration to obtain a cutpoint. These values are then scored

with the formula:

$$\text{score} = \sqrt{\frac{\sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1}(\text{rates}_i - \text{rates}_j)^2}{\frac{n^2-(n \bmod 2)}{4}}}$$

Where n is the number of classes, rates_i is the ratio of number of samples of class i that has appeared before the cutpoint to total number of samples in class i.

The value with the highest score if higher than a minimum threshold is stored as a cutpoint and used to divide the data into two. The same process is repeated for both divisions of data to divide them further until either the highest score from all divisions is less than the threshold or a maximum depth of search is reached.

Table 1: Binning Example

Original Sorted Values

y	**f4**	**Cutpoint**	**Score**
b	7	-	-
b	14	10.5	0.333
a	20	17	0.667
b	20	-	-
a	22	21	0.75
a	29	25.5	0.5
a	31	30	0.25

Values after 1 Split

y	**f4**	**Cutpoint**	**Score**
b	7	-	-
b	14	10.5	0.333
a	20	17	0.667
b	20	-	-
a	22	-	-
a	29	25.5	NaN
a	31	30	NaN

Values after 2 Split

y	**f4**	**Cutpoint**	**Score**
b	7	-	-
b	14	10.5	NaN
a	20	-	-
b	20	-	-
a	22	-	-
a	29	25.5	NaN
a	31	30	NaN

Binning of Numerical Data

y	**f4**	**f4 ranges**
a	31	$(21, \infty)$
a	29	$(21, \infty)$
a	20	$(17, 21)$
a	22	$(21, \infty)$
b	20	$(17, 21)$
b	14	$(-\infty, 17)$
b	7	$(-\infty, 17)$

All selected cutpoints are collectively used to make ranges by sorting them in ascending order along with $-\infty$ and ∞. Consecutive values are used as boundaries of ranges which form the bins that all collectively ideally represent the original data in discrete or nominal form.

Every unique value in nominal features or bin created from numerical features are assigned a unique numerical value for said feature. These numerical value are processed as their binary form, using every bit of those values for a distinct binary feature. This process is repeated for every feature thereby binarizing all the data.

Table 2: Binarization Example

Original Values

y	f1	f2	f3	f4
a	1	green	yes	31
a	4	blue	no	29
a	2	blue	yes	20
a	4	red	no	22
b	3	red	yes	20
b	2	green	no	14
b	4	green	no	7

Binning of Numerical Data

y	f1	f2	f3	f4
a	$(-\infty, 1.5)$	green	yes	$(21, \infty)$
a	$(3.5, \infty)$	blue	no	$(21, \infty)$
a	$(1.5, 2.5)$	blue	yes	$(17, 21)$
a	$(3.5, \infty)$	red	no	$(21, \infty)$
b	$(2.5, 3.5)$	red	yes	$(17, 21)$
b	$(1.5, 2.5)$	green	no	$(-\infty, 17)$
b	$(3.5, \infty)$	green	no	$(-\infty, 17)$

Label Encoding

y	f1	f2	f3	f4
a	0	0	1	2
a	3	1	0	2
a	1	1	1	1
a	3	2	0	2
b	2	2	1	1
b	1	0	0	0
b	3	0	0	0

Binarized Values

y	b1_1	b1_0	b2_1	b2_0	b3	b4_1	b4_0
a	false	false	false	false	true	true	false
a	true	true	false	true	false	true	false
a	false	true	false	true	true	false	true
a	true	true	true	false	false	true	false
b	true	false	true	false	true	false	true
b	false	true	false	false	false	false	false
b	true	true	false	false	false	false	false

Table 2 demonstrates how original feature values are sequentially transformed through the steps of binning, label encoding, and binary conversion, highlighting the systematic nature of the binarization process and its effectiveness in preparing data for further analysis.

3.2 Pattern Generation

Once the data is binarized, patterns are generated for the purpose of classification. Firstly, an empty list of rules and list of patterns that contains a single blank pattern is generated as a starting point.

Following this, from each existing pattern two new pattern for every feature in binary data are generated by adding each feature once as is and once as its inverse

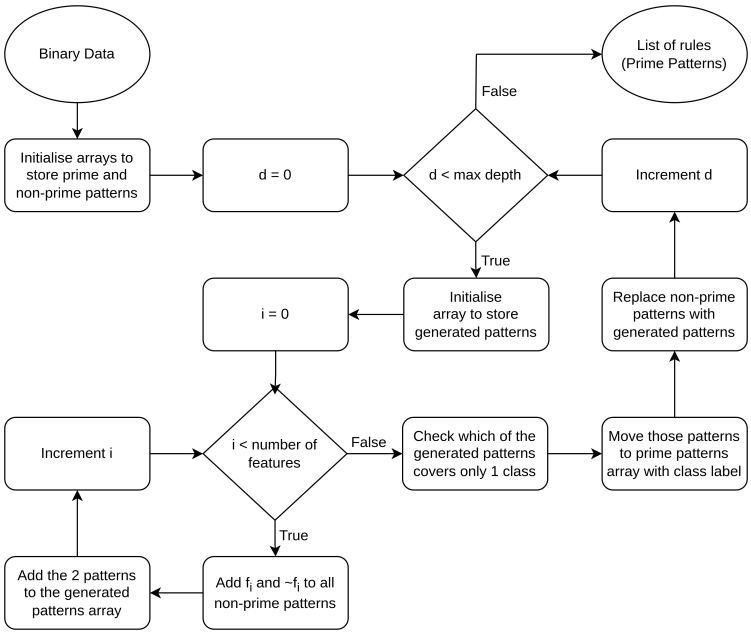

Fig. 4 Pattern Generation Process

to the pattern. Next, patterns that specifically cover instances of a single class are identified and moved to the list of rules with the association with said class. The observations that are covered by these rules are removed from consideration for new rules. The patterns that do not cover a single class are retained for further iterations. This process is repeated until either all data points are covered by the generated rules or a predefined maximum number of iterations is reached.

3.3 Classification

For classification of each sample a short circuiting approach is used. The evaluation is done by matching the sample to every rule in the order the rules were generated. Which stops as soon as a matching rule is found and the sample is classified with the class associated with that rule. If no rules match the sample, its class is set to NULL.

4 Results and Analysis

This section presents the results of our experiments evaluating the proposed intrusion detection method using the NSLKDD and KDDCup99 (10%) datasets. Additionally, we discuss the limitations of the approach.

4.1 System Configuration

The experiments and analyses were conducted on a high-performance HP Z640 Workstation equipped with an Intel Xeon E5-2620 v3 processor (12 cores, 3.20 GHz), 15.52 GiB of RAM, and an NVIDIA Quadro K2200 GPU. The system ran Fedora Linux 41 (Workstation Edition) x86_64, with the Linux 6.11.8-300.fc41.x86_64 kernel, utilizing the COSMIC 1.0.0 alpha.3_20241120-1.fc41 desktop environment and the Wayland-based `cosmic-comp` window manager. The algorithm was implemented in Rust 1.82.0 using the polars crate which was interfaced with python using the crate pyo3. Python was used to load and run datasets using the polars library.

4.2 Dataset Description

We utilize two popular intrusion detection datasets, as shown in Table 3. These

Table 3 Datasets

Dataset	Samples	Features	Classes
NSL-KDD [22]	125,973	41	23
KDD-Cup99 [23]	4,898,431	41	23

datasets contain labeled network traffic data, representing normal and malicious activities.

4.3 Performance Metrics

The following performance metrics were used to evaluate the results:

- **Accuracy**: The proportion of correctly predicted instances out of the total instances.
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$
- **Precision**: The ratio of true positive predictions to the total predicted positives (i.e., how many selected items were relevant).
$$\text{Precision} = \frac{TP}{TP + FP}$$
- **Recall**: The ratio of true positives to the actual positives in the dataset (i.e., how many relevant items were selected).
$$\text{Recall} = \frac{TP}{TP + FN}$$
- **F2-Score**: A weighted harmonic mean of precision and recall that gives more importance to recall, calculated as:
$$\text{F2-Score} = \frac{5 \cdot \text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}}.$$

- **Data Coverage**: The proportion of predicted instances out of the total instances.

4.4 Tabular Results

The results presented in this section pertain to experiments conducted with a maximum depth of 4 for pattern generation. Table 4 provides an overview of the performance metrics across the datasets NSLKDD and KDDCup99 (10%). The metrics include accuracy, precision, recall, F2-score, and coverage. For the NSLKDD dataset, the accuracy achieved was 75.48%, with a precision of 85.58%, a recall of 75.48%, an F2-score of 77.30%, and a coverage of 92.56%. On the other hand, the KDDCup99 (10%) dataset exhibited significantly higher performance with an accuracy of 96.10%, a precision of 99.96%, a recall of 96.10%, an F2-score of 96.85%, and a coverage of 96.13%. These results demonstrate the effectiveness of the proposed method, particularly for the KDDCup99 dataset.

Table 4 Performance Metrics for Different Datasets

Dataset	Accuracy	Precision	Recall	F2-Score	Coverage
NSLKDD	75.48%	85.58%	75.48%	77.30%	92.56%
KDDCup99 (10%)	96.10%	99.96%	96.10%	96.85%	96.13%

Table 5 provides insights into the characteristics of the datasets used in the experiments. The KDDCup99 (10%) dataset consists of 330,994 samples with 41 features, 32 of which are preprocessed and 29 are binarized. In contrast, the NSLKDD dataset comprises 125,973 samples with 41 features, of which 54 are binarized. These details emphasize the variability in dataset size and feature representation, highlighting the adaptability of the binarization and rule-generation process.

Table 5 Dataset Samples and Features

Dataset	Samples	Features
KDDCup99 (10%)	330,994	41 (32 preprocessed, 29 binarized)
NSLKDD	125,973	41 (54 binarized)

Table 6 focuses on the number of rules generated for the datasets based on different minimum match criteria at a depth of 3. For the KDDCup99 dataset, 243 rules were generated for at least one match, 85 for a minimum of 10 matches, and 42 for a minimum of 50 matches. Similarly, for the NSLKDD dataset, the number of rules generated was 365, 150, and 65 for minimum matches of 1, 10, and 50, respectively. These statistics indicate how stricter matching criteria reduce the number of generated rules, ensuring more selective and meaningful rules.

Table 7 summarizes the overall metrics for both datasets under different matching criteria. For the KDDCup99 dataset, with at least one match, the accuracy, precision, recall, and coverage were 95.91%, 99.96%, 95.91%, and 95.94%, respectively. As the

Table 6 Number of Rules Generated for Different Minimum Matches of Sample at depth 3

Matches	KDDCup99 (10%)	NSLKDD
1	243	365
10	85	150
50	42	65

match criteria increased to 10 and 50, the accuracy and recall reduced to 93.27% and 75.80%, respectively, but the precision remained extremely high (99.98% and 99.97%). Similarly, for the NSLKDD dataset, stricter match criteria also led to lower accuracy and recall while maintaining high precision. For instance, at one match, the accuracy was 75.42%, which dropped to 71.93% and 70.08% for match criteria of 10 and 50, respectively.

Table 7 Overall Metrics for Different Matching Criteria

Dataset	Matches	Acc	Prec	Recall	F2-Score	Coverage
KDDCup99	1	95.91%	99.96%	95.91%	-	95.94%
	10	93.27%	99.98%	93.27%	-	93.29%
	50	75.80%	99.97%	75.80%	-	75.81%
NSLKDD	1	75.42%	86.13%	75.42%	-	91.52%
	10	71.93%	87.29%	71.93%	-	85.55%
	50	70.08%	88.83%	70.08%	-	81.17%

The decision matrices provided in Tables 8 and 9 offer a detailed view of classification performance for the NSLKDD and KDDCup99 datasets under varying minimum match criteria. For the NSLKDD dataset, as shown in Table 8, the number of correctly classified "Attack" and "Normal" instances decreases as the minimum match criterion becomes stricter, resulting in an increase in *NULL* (unmatched) cases. For instance, with a minimum of one match, there were 8,295 correctly classified "Attack" instances and 8,709 "Normal" instances. However, with a minimum of 50 matches, these numbers reduced to 7,725 and 8,073, respectively. A similar trend is observed in the decision matrices for the KDDCup99 dataset in Table 9. For example, with one match, 124,431 "DOS" instances and 30,515 "Normal" instances were correctly classified, while with 50 matches, these numbers dropped to 95,373 and 27,265, respectively. This illustrates the trade-off between stricter match criteria and classification coverage.

Overall, the tables provide a comprehensive view of the performance and effectiveness of the binarization and rule-generation framework across multiple datasets, emphasizing how various factors, such as dataset characteristics and matching criteria, influence the classification results.

4.5 Limitations of Proposed Work

The proposed method has some limitations:

Table 8 Decision Matrix for NSLKDD

For Minimum 1 Matching Sample			
	Attack	**Normal**	*NULL*
Attack	8295	3336	1202
Normal	292	8709	710

For Minimum 10 Matching Samples			
	Attack	**Normal**	*NULL*
Attack	7781	2801	2151
Normal	268	8336	1107

For Minimum 50 Matching Samples			
	Attack	**Normal**	*NULL*
Attack	7725	2253	2855
Normal	247	8073	1391

Table 9 Decision Matrix for KDDCup (10%)

For Minimum 1 Matching Sample						
	DOS	**Normal**	**Probe**	**R2L**	**U2R**	*NULL*
DOS	124431	8	0	0	0	4742
Normal	3	30515	3	2	5	1574
Probe	2	8	1259	0	0	86
R2L	0	10	0	42	1	219
U2R	1	1	0	2	13	0

For Minimum 10 Matching Samples						
	DOS	**Normal**	**Probe**	**R2L**	**U2R**	*NULL*
DOS	123506	4	0	0	0	5671
Normal	1	27529	1	1	0	4570
Probe	1	2	898	0	0	454
R2L	0	8	0	123	0	241
U2R	1	2	0	1	7	6

For Minimum 50 Matching Samples						
	DOS	**Normal**	**Probe**	**R2L**	**U2R**	*NULL*
DOS	95373	3	0	0	0	33805
Normal	1	27265	2	1	0	4833
Probe	4	2	843	0	0	504
R2L	0	4	0	94	0	274
U2R	1	2	0	0	0	14

- **Long Training Time:** The training time of LAD is of the order $O(scf^d)$, where s is the number of samples, c is the number of classes, f is the number of features in the binarized data, and d is the maximum number of attributes in a rule.
- **Incomplete Coverage of Dataset:** The dataset may not be completely covered by LAD if the maximum number of attributes is not set to a sufficiently high value.

5 Conclusion

In this study, we explored the application of a Learning Algorithm for Intrusion Detection (LAD) on two well-known intrusion detection datasets: NSL-KDD and KDD-Cup99. The experiments were conducted on a high-performance workstation, and various performance metrics, including accuracy, precision, recall, F2-score, and data coverage, were used to evaluate the effectiveness of the proposed method.

The results indicated that LAD achieved a good balance between performance and data coverage across both datasets. Specifically, the method demonstrated high accuracy and precision, especially on the KDD-Cup99 dataset (10% subset), with an accuracy of 96.10% and precision of 99.96%. However, it was also observed that LAD exhibited limitations, such as long training times and incomplete dataset coverage when the maximum number of attributes was not sufficiently set.

Despite these challenges, the proposed approach provided valuable insights into improving the effectiveness of intrusion detection systems, and future work may focus on addressing the limitations related to training time and dataset coverage to enhance the overall performance of the system.

In conclusion, the findings from this study contribute to the development of more efficient and accurate intrusion detection techniques, with potential for further optimization in future research.

References

[1] Abbasi, J.S., Bashir, F., Qureshi, K.N., Islam, M.N., Jeon, G.: Deep learning-based feature extraction and optimizing pattern matching for intrusion detection using finite state machine. Computers & Electrical Engineering **92**, 107094 (2021)

[2] Liu, J., Zhang, W., Tang, Z., Xie, Y., Ma, T., Zhang, J., Zhang, G., Niyoyita, J.P.: Adaptive intrusion detection via ga-gogmm-based pattern learning with fuzzy rough set-based attribute selection. Expert Systems with Applications **139**, 112845 (2020)

[3] Santhosh Kumar, S.V.N., Selvi, M., Kannan, A.: A comprehensive survey on machine learning-based intrusion detection systems for secure communication in internet of things. Computational Intelligence and Neuroscience **2023**(1), 8981988 (2023)

[4] Ugtakhbayar, N., Usukhbayar, B., Baigaltugs, S.: A hybrid model for anomaly-based intrusion detection system. In: Advances in Intelligent Information Hiding and Multimedia Signal Processing vol. 2, pp. 419–431. Springer, Singapore (2020)

[5] Alhomoud, A., Munir, R., Disso, J.P., Awan, I., Al-Dhelaan, A.: Performance evaluation study of intrusion detection systems. Procedia Computer Science **5**, 173–180 (2011)

[6] Kaur, S., Singh, M.: Hybrid intrusion detection and signature generation using

deep recurrent neural networks. Neural Computing and Applications **32**(12), 7859–7877 (2020)

[7] Chauhan, S., Gangopadhyay, S., Gangopadhyay, A.K.: Intrusion detection system for iot using logical analysis of data and information gain ratio. Cryptography **6**(4), 62 (2022)

[8] Sowmya, T., Anita, E.M.: A comprehensive review of ai-based intrusion detection system. Measurement: Sensors **28**, 100827 (2023)

[9] Boros, E., Hammer, P.L., Ibaraki, T., Kogan, A.: Logical analysis of numerical data. Math. Program. **79**(1), 163–190 (1997)

[10] Boros, E., Hammer, P.L., Ibaraki, T., Kogan, A., Mayoraz, E., Muchnik, I.: An implementation of logical analysis of data. IEEE Trans. Knowl. Data Eng. **12**(2), 292–306 (2000)

[11] Kim, G., Lee, S., Kim, S.: A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications **41**(4), 1690–1700 (2014)

[12] Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J.: Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity **2**(1), 1–22 (2019)

[13] Abdulganiyu, O.H., Ait Tchakoucht, T., Saheed, Y.K.: A systematic literature review for network intrusion detection system (ids). International Journal of Information Security **22**(5), 1125–1162 (2023)

[14] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F.: Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies **32**(1), 4150 (2021)

[15] Dini, P., Elhanashi, A., Begni, A., Saponara, S., Zheng, Q., Gasmi, K.: Overview on intrusion detection systems design exploiting machine learning for networking cybersecurity. Applied Sciences **13**(13), 7507 (2023)

[16] Alexe, S., Hammer, P.L.: Accelerated algorithm for pattern detection in logical analysis of data. Discrete Appl. Math. **154**(7), 1050–1063 (2006)

[17] Crama, Y., Hammer, P.L., Ibaraki, T.: Cause-effect relationships and partially defined boolean functions. Ann. Oper. Res. **16**, 299–325 (1988)

[18] Chauhan, S., Gangopadhyay, S., Gangopadhyay, A.K.: Intrusion detection system for iot using logical analysis of data and information gain ratio. Cryptography **6**(4), 62 (2022)

[19] Gangopadhyay, A.K., Sheth, T., Chauhan, S.: Lad in finance: accounting analytics and fraud detection. Adv. Comput. Intell. **3**(1), 4 (2023)

[20] Bonates, T.O., Hammer, P.L., Kogan, A.: Maximum patterns in datasets. Discrete Appl. Math. **156**(6), 846–861 (2008)

[21] Lejeune, M., Lozin, V., Lozina, I., Ragab, A., Yacout, S.: Recent advances in the theory and practice of logical analysis of data. Eur. J. Oper. Res. **275**(1), 1–15 (2019)

[22] Mohi-ud-din, G.: NSL-KDD. Available at: https://dx.doi.org/10.21227/425a-3e55 (2018)

[23] Cheng, M.: KDD-CUP99. Available at: https://dx.doi.org/10.21227/n4qf-v834 (2022)