ESI Preprints

Not Peer-reviewed

Anomaly Detection in a Portal System Network: A Review of Techniques, Challenges, and Future Directions

O.O. Green
Department of Information Communication Technology, Lagos State University of Education, Lagos, Nigeria
S.M. Yusuf
M.B. Abdulrazaq
B. Yahaya
Z. Haruna
A. Ore-Ofe
Department of Computer Engineering,
Ahmadu Bello University, Zaria, Nigeria
S.O. Omogoye
Department of Electrical and Electronics Engineering, Lagos State University of Science and Technology, Lagos, Nigeria
A.S. Adegoke
S.O. Salami
Department of Computer Engineering,
Lagos State University of Science and Technology, Lagos, Nigeria

Doi: 10.19044/esipreprint.7.2025.p638

Approved: 25 July 2025
Posted: 27 July 2025

Cite As:
Green, O.O., Yusuf, S.M., Abdulrazaq, M.B., Yahaya, B., Haruna, Z., Ore-Ofe, A., Omogoye, S.O., Adegoke, A.S. & Salami, S.O. (2025). *Anomaly Detection in a Portal System Network: A Review of Techniques, Challenges, and Future Directions.* ESI Preprints. https://doi.org/10.19044/esipreprint.7.2025.p638

Abstract

Portal system networks are vital for education, governance, and corporate operations, but face growing risks from evolving cyber threats. This study proposes a hybrid anomaly detection framework that combines the Enhanced Modified Lion Optimization Algorithm (EMLOA) with One-Class Support Vector Machines (OCSVM) to enhance threat detection in such environments. Unlike traditional rule-based or statistical methods, which lack adaptability, or conventional machine learning techniques that demand extensive labeled data and computational power, the EMLOA-OCSVM model achieves high accuracy (99.9%), low training latency (3.05 seconds), and scalability in dynamic settings. The framework employs a sigmoid function-based strategy to dynamically optimize hyperparameters (γ and ν), enhancing convergence speed and detection performance. Evaluations using the UNSW-NB15 dataset (reflecting modern attack patterns) and real-world logs from Lagos State University of Education (LASUED) demonstrate the model's practical relevance. Key innovations include dynamic threshold tuning and improved interpretability, reducing false positives without sacrificing efficiency. Robust performance is confirmed through accuracy, precision, recall, F1-score, and ROC-AUC metrics. Future research should prioritize lightweight, explainable hybrid models capable of countering advanced threats while maintaining system performance.

Keywords: Anomaly Detection, Portal Systems, EMLOA, OCSVM, Hybrid Models, Cybersecurity, Explainable AI (XAI), Real-Time Monitoring, Threshold Optimization, Machine Learning

Introduction

Portal system networks are essential platforms for educational institutions, corporate organizations, and public administration. They facilitate critical operations such as student and staff record management, payment processing, and academic activities. However, these systems face an increasing number of security threats, including data breaches, unauthorized access, and network anomalies. The early detection of abnormalities in a portal system is essential to maintaining its operational integrity and protecting sensitive data(Hashmi & Ahmad, 2020). For instances, major cyber incidents like the July 2015 data breach at the University of California, Los Angeles (UCLA), which exposed 4.5 million records at a cost of over $70 million, and the July 2023 University of Manchester was a victim of cyber-attack, resulting to vulnerabilities of about 11,000 staff and more than 46,000 students' data(Paganini, 2023)Highlight the severe consequences of insufficient anomaly detection systems. In Nigeria, the 2023 presidential elections recorded 12.9 million cyber threats reported by the Minister of Communication and Digital Economy, Isa Pantami(Ukagwu, 2023). Further, it emphasizes the need for robust security mechanisms.

Research Gap

Conventional anomaly detection techniques rely on statistical and rule-based models, which cannot recognize complex and dynamic threats

(Darveau et al., 2020). Support Vector Machines (SVM), k-means Clustering, and Deep Learning are prominent examples of machine learning-based techniques that have enhanced detection capabilities; however, they are limited in their ability to adapt to unexpected attack patterns and incur significant computing costs. Additionally, current techniques produce a high number of false positives, which results in operational inefficiencies. The Enhanced Modified Lion Optimization Algorithm (MLOA) introduces a dynamic approach to threshold tuning, improving the accuracy and robustness of anomaly detection in portal networks.

Research Objective

This review aims to analyze and compare current anomaly detection techniques in portal networks. Specifically, it will:

1. Review traditional and machine learning-based anomaly detection methods, identifying their strengths and weaknesses.
2. Examine the potential of hybrid approaches, review methodology, including the MLOA, in enhancing detection performance.
3. Highlight key challenges in anomaly detection, such as scalability, real-time processing, and adaptability.
4. Provide recommendations for future research, emphasizing AI-driven optimization techniques for improved security.

Anomaly Detection in Portal Networks

Anomaly detection is an essential element of cybersecurity and system performance monitoring in portal networks, where immediate access to financial, administrative, and academic data is essential (Fernandes et al., 2022). Portal systems, widely used in government organizations, business environments, and educational institutions, facilitate crucial tasks such as staff administration, fee payment, student registration, and result processing. However, as these systems become increasingly complex, they become more vulnerable to security concerns such as insider threats, denial-of-service (DoS) attacks, unauthorized access, and data leaks. Deviations or abnormalities from normal system behavior that may indicate malicious activities or system failures are known as portal network anomalies (Mittal et al., 2024). Early anomaly detection prevents potential threats, lowers security risks, and ensures ongoing service availability.

Types of Anomalies in the Portal Network

Anomalies in portal systems can be categorized into three main groups based on particular characteristics and implications for network operations:

1. Point Anomalies: Point anomalies refer to individual data points significantly deviating from normal system behavior. These anomalies are often isolated incidents within a dataset, indicating unusual system activities. For instance, in Portal Networks, irregular login attempts: A single failed login attempt using an incorrect password is normal, but login attempts from an unusual geographic location (e.g., a student from Nigeria logging in from China without prior travel records) could indicate unauthorized access, unexpected large file transfers: A student or staff account suddenly transferring an abnormally large amount of data within a short time frame may indicate data exfiltration, and payment processing errors: A payment transaction failing multiple times due to unverified bank details may be a fraudulent attempt or a sign of a system misconfiguration.
2. Contextual Anomalies: Contextual anomalies refer to data points that appear normal in one context but are considered unusual in another. These anomalies are highly dependent on the surrounding data and external factors such as time of access, user privileges, or operational conditions. In a Portal Networks, accessing the System Outside Normal Hours: A university administrator logging in to process student results during official working hours is normal, but if the same administrator logs in at midnight and attempts to modify records, it could signal an insider attack, irregular examination registration patterns: If a student registers for an examination minutes before the deadline every semester, it may be a habit. However, if multiple students suddenly register within the same short time frame, it could indicate a system malfunction or coordinated cheating attempt, and high central processing unit (CPU) or memory utilization during Off-Peak hours: A portal system experiencing high network traffic at night when no scheduled operations are planned could suggest a brute-force attack or an unauthorized system scan.
3. Collective Anomalies: Collective anomalies occur when a group of related data points collectively indicate an abnormal pattern, even if individual data points appear normal. These types of anomalies are common in coordinated cyberattacks and fraudulent activities in portal networks. For example in Portal Networks, multiple failed login attempts across different accounts: If multiple users experience failed login attempts within a short period, it could indicate a brute-force attack, where attackers try different passwords to gain unauthorized access, simultaneous access from multiple locations: A single student logging in from different geographic locations within minutes suggests that credentials have been compromised and are being used by unauthorized individuals, and unusual surge in fee

payment transactions: A sudden spike in fee payments from newly created student accounts could indicate a security vulnerability being exploited for financial fraud.

Importance of Anomaly Detection in Portal Networks

Effective anomaly detection is crucial for ensuring the security, integrity, and availability of portal systems. Some key benefits include:

1. Prevention of Cyber Threats: Early detection of unauthorized access, malware infections, and phishing attacks helps in proactive threat mitigation.
2. Minimizing Data Breaches: Identifying anomalies in data access patterns can prevent leakage of sensitive student or staff information.
3. Enhancing System Performance: Detecting network bottlenecks, slow query responses, and resource overuse helps maintain optimal system efficiency.
4. Regulatory Compliance: Many institutions must comply with data protection regulations (e.g., GDPR, NITDA Regulations), requiring real-time monitoring and anomaly detection.

Techniques for Early Anomaly Detection

Early anomaly detection is essential for securing portal system networks, which handle academic, administrative, and financial transactions. The three basic techniques commonly used for the detection of anomalies in portal networks are traditional methods, machine learning approaches, and hybrid approaches.

Traditional Methods for Anomaly Detection in Portal Networks

Traditional anomaly detection methods have been widely used in portal network security and performance monitoring due to their simplicity and interpretability. These methods rely on predefined rules, statistical thresholds, and probability distributions to identify abnormal activities within a network. While effective in detecting well-defined anomalies, traditional methods often struggle with adaptive threats, dynamic data patterns, and high-dimensional datasets.

This section explores two primary traditional methods used in anomaly detection: Rule-Based Detection and Statistical Models.

Rule-Based Detection

Rule-based anomaly detection is one of the earliest and most widely used methods for identifying suspicious activities in a network(Moore, 2025). It establishes predefined rules and thresholds that flag anomalies when specific conditions are met. This method is beneficial for detecting known threats and enforcing security policies based on historical patterns.

How Rule-Based Detection Works

Rule-based detection operates by establishing predefined rules and thresholds that define expected system behavior. Administrators manually configure these rules based on historical patterns, security policies, and operational requirements(Moore, 2025). The system continuously monitors user activities, such as login attempts, data transfers, and transaction requests, and compares them against the established rules. If a deviation is detected, such as multiple failed login attempts, unauthorized data access, or abnormal system usage, the system flags the activity as a potential anomaly and triggers an appropriate response. This response may include sending alerts to administrators, temporarily blocking user access, or initiating security protocols to mitigate potential threats(Duffield et al., 2009). While simple and effective for known attack patterns, rule-based detection requires constant updates and tuning to remain effective against emerging threats and dynamic cyberattacks(Emesoronye, 2024).

Advantages of Rule-Based Detection

1. Simple to Implement: Rules are easy to define and require minimal computational resources.
2. Effective for Known Threats: Works well in environments with predictable attack patterns.
3. Immediate Response: Since rules are predefined, the system can instantly flag anomalies without requiring complex computations.

Limitations of Rule-Based Detection

1. High False Positives: Legitimate activities may be flagged simply because they violate predefined thresholds.
2. Inability to Detect New Threats: Rule-based methods cannot adapt to evolving cyber threats that do not fit predefined patterns.
3. Manual Rule Updating: Requires continuous manual updates to remain effective, making it impractical for large-scale dynamic environments.

Statistical Models

Statistical models improve upon rule-based detection by using mathematical and probabilistic approaches to detect anomalies. These models assume that normal system behavior follows a predictable distribution and that any deviation from this pattern may indicate an anomaly(Rousseeuw & Hubert, 2018).

How Statistical Models Work

Statistical models detect anomalies by analyzing deviations from established behavioral patterns within a system. The process begins with collecting baseline data over time to define a normal probability distribution of system activities, such as login frequency, transaction amounts, or network traffic levels. Once this baseline is established, the model continuously monitors new data and compares it against expected patterns. If a data point significantly deviates from the predicted distribution, it is flagged as an anomaly, indicating potential security threats, system malfunctions, or unusual activity. Common statistical techniques, such as Gaussian Mixture Models and Autoregressive Integrated Moving Average (ARIMA), are used to detect these variations (Li et al., 2016). While statistical models offer a more flexible and data-driven approach than rule-based detection, they may struggle with dynamic system behaviors and require periodic recalibration to maintain accuracy in evolving environments.

1. Gaussian Mixture Models (GMM) are probabilistic models that assume regular system activity follows a combination of numerous Gaussian (normal) distributions rather than a single uniform distribution, in comparison with typical statistical models that assume a single mean and variance for normal behavior. Due to their ability to capture changes in system activity, these models are very useful for identifying anomalies in dynamic contexts such as portal networks, financial transactions, and network traffic monitoring. GMM operates by clustering data points into several Gaussian distributions, each of which represents a separate area of normal behavior. The probability density function (PDF) of a GMM(Yu et al., 2023) is given by:

$$P(X) = \sum_{i=1}^{k} \pi_i N(\frac{X}{\mu_i} C_i) \tag{1}$$

Where: P(X) = Probability of data point (X) in the mixture model.
K = Number of Gaussian components (clusters)
π_i= Mixing coefficient for the i^{th} Gaussian component (weights, such that $\sum_{i=1}^{k} \pi_i = 1$)
$N(\frac{X}{\mu_i} C_i)$ = Gaussian (normal) distribution with mean μ_i and covariance matrix C_i.

For instance, a university portal system may show various trends in the behaviour of instructors, administrators, and students. GMM is capable of considering every user similarly by breaking out user behaviour into many clusters based on variables like transaction patterns, access rights, and login frequency. Following the determination of the distributions, GMM evaluates new data points and calculates their probability of belonging to one of the predefined clusters. A data point is considered abnormal if it significantly deviates from the expected probability range.

2. The Autoregressive Integrated Moving Average (ARIMA) is a time-series forecasting model used in anomaly detection to identify unexpected deviations from normal system behavior by analyzing historical trends and predicting future values (Cipra, 2020). The model consists of three key components: Auto Regression (AR), which uses past observations to forecast future values; Integration (I), which stabilizes non-stationary data by differentiating values over time; and Moving Average (MA), which refines predictions by accounting for past errors. When a newly observed data point significantly deviates from the forecasted trend, it is flagged as an anomaly, prompting further investigation. ARIMA is particularly useful in monitoring portal login trends, network traffic fluctuations, payment transactions, and server load balancing, where patterns evolve. The Autoregressive Integrated Moving Average (ARIMA) model is typically represented as ARIMA (p, d, q)(Cipra, 2020), where:

p = Number of autoregressive (AR) terms
d = Number of differencing operations (integration step)
q = Number of moving average (MA) terms
The general mathematical form of an ARIMA model is:

$$Y_t = C + \sum_{I=1}^{P} \emptyset_i Y_{t-1} + \sum_{j=1}^{q} \theta_j \in_{t-j} + \in_t \tag{2}$$

Where: Y_t= Value of the time series at time
C = Constant term (optional)
$\emptyset_i$ = Coefficients of the autoregressive (AR) terms
θ_j = Coefficients of the moving average (MA) terms
$\in_t$ = Error term (white noise) at time
P= Number of past values used (lag order in AR model)
q = Number of past error terms used (lag order in MA model)

The differencing operation (I component) is applied to make the time series stationary:

$$Y_t' = Y_t - Y_{t-1} \tag{3}$$

Its advantages include effectiveness for seasonal or cyclical data, suitability for small datasets, and interpretability, making it easier for administrators to act on detected anomalies. However, ARIMA assumes linear relationships, requires extensive parameter tuning, and is less effective for real-time anomaly detection than advanced machine learning models. Despite these limitations, ARIMA remains a valuable tool for time-series anomaly detection, particularly when combined with other techniques to enhance accuracy and efficiency.

Advantages of Statistical Models

1. More Adaptable than Rule-Based Detection: It can detect unusual patterns without requiring predefined rules.
2. Effective for Time-Series Analysis: Well-suited for detecting trends and unexpected deviations in network behavior.
3. Lower False Positives: More precise than simple rule-based thresholds.

Limitations of Statistical Models

1. Assumes Static Data Distributions: Dynamic and evolving threats are challenging for many statistical models to address.
2. Computationally Expensive: Real-time anomaly detection in large-scale portal networks necessitates a substantial amount of computational resources.
3. Less Effective for Complex Attacks: It frequently fails to distinguish between real system modifications and subtle malicious activity.

Machine Learning Approaches for Anomaly Detection

Machine learning (ML) approaches for anomaly detection involve training models to recognize deviations from expected patterns in data. These approaches automate threat detection, making them crucial for cybersecurity, financial fraud detection, and portal network security(Bablu, 2025). ML techniques can be broadly classified into supervised, unsupervised, and semi-supervised learning, with deep learning emerging as a powerful tool for detecting anomalies in complex datasets.

Supervised Learning for Anomaly Detection

Models that distinguish between normal and abnormal activity are trained using labelled data in supervised learning for anomaly detection(Bablu, 2025). Common methods include random forests, which mix many decision trees to improve accuracy, and decision trees, which use hierarchical criteria for classification. While neural networks can effectively classify anomalies if sufficient labelled data is available, support vector machines (SVMs) establish decision boundaries to differentiate between normal and abnormal data. This approach is extremely accurate, but it depends on the availability of datasets that are appropriately labelled(Trebar, 2021).

Advantages of Supervised Learning for Anomaly Detection

1. High Accuracy: Since supervised learning models are trained on labelled data, they can achieve high precision in distinguishing normal and anomalous behavior.
2. Clear Decision Boundaries: Algorithms like Decision Trees, Random Forests, and Support Vector Machines (SVMs) create well-defined boundaries between normal and abnormal data points, improving interpretability.
3. Effective for Known Anomalies: If past anomalies are well-documented, the model can learn to detect similar future anomalies with high reliability.
4. Scalability: Once trained, supervised models can efficiently analyze large datasets and detect anomalies in real time.
5. Customizable: Models can be fine-tuned to detect specific types of anomalies based on historical data, making them adaptable to different portal systems and cybersecurity needs.

Application of Supervised Learning for Anomaly Detection

1. Detecting unauthorized access by classifying user login behaviors.
2. Identifying fraudulent transactions in university fee payment systems.

Limitations of Supervised Learning for Anomaly Detection

1. Requires a large amount of labeled data, which is often unavailable.
2. Struggles with new types of anomalies not seen in the training data.

Semi-Supervised Learning for Anomaly Detection

Semi-supervised learning for anomaly detection combines a larger pool of unlabeled data with a smaller amount of labeled data to identify patterns and abnormalities. This approach allows models to generalize from a limited number of labelled instances and correctly classify new data. A one-class support vector machine (OCSVM) is another well-liked technique that only uses standard data for training and labels deviations as anomalies(Aug, 2023). This method strikes a balance between the accuracy of supervised learning and the flexibility of unsupervised learning and self-training neural networks that use initially labeled data to predict labels for unlabeled data, improving detection over time(Yu et al., 2023). For instance, a university portal system analyses both labelled and unlabeled login data using semi-supervised learning to identify unauthorized access attempts. It is first trained on a small sample of authenticated staff and student logins, picking up typical behaviors such as IP addresses, device usage, and login times. After that, self-training neural networks categorize new, unlabeled logins, highlighting anomalous behavior, such as midnight logins from unidentified devices. Furthermore, an OCSVM that has only been trained on typical logins might detect irregularities like multiple unsuccessful tries from various IP addresses. The system's accuracy improves with time, enhancing its capacity to identify possible security risks.

Advantages of Semi-Supervised Learning for Anomaly Detection

1. Requires Less Labeled Data: Unlike supervised learning, semi-supervised models can learn effectively with only a small set of labeled data, reducing the cost and effort of manual data labeling.
2. Improved Generalization: By leveraging both labeled and unlabeled data, these models can identify complex patterns and anomalies that may not be captured in a purely supervised setting.
3. Better Adaptability: Semi-supervised techniques, such as Self-Training Neural Networks and One-Class SVM, can adapt to new or evolving anomalies without extensive retraining.
4. Enhanced Detection of Rare Anomalies: Since most real-world datasets contain very few labelled anomalies, semi-supervised learning helps detect outliers in large amounts of unlabelled data.
5. More Robust in Dynamic Environments: Suitable for portal systems where user behavior changes frequently, such as variations in student logins, course registrations, or financial transactions.

Application of Semi-Supervised Learning for Anomaly Detection

1. Detecting new types of cyber threats by learning from past attack patterns.
2. Improving fraud detection by refining classification models with new labeled data over time.

Limitation of Semi-supervised Learning for Anomaly Detection

1. Requires careful selection of labeled data to avoid bias.
2. Can struggle with highly dynamic environments where behaviors frequently change.

Unsupervised Learning for Anomaly Detection

Unsupervised learning algorithms identify patterns in typical behavior and highlight deviations to detect abnormalities, eliminating the need for labeled data. Clustering methods like Density-Based Spatial

Clustering of Applications with Noise (DBSCAN) identify outliers based on data density, while K-Means groups comparable data points and labels those that do not fit as anomalies. Models of statistics, such as Gaussian Mixture Models (GMM), identify deviations as anomalies by assuming that typical behavior follows multiple Gaussian distributions. Principal Component Analysis (PCA) highlights anomalous patterns by reducing the dimensionality of the data. Effective anomaly detection in complex systems is made possible by these strategies, which do not require prior knowledge of certain attack patterns(Claudius & Andersen, 2022). For instance, applying unsupervised learning techniques like K-Means Clustering and DBSCAN, a university portal system can enhance security and identify unauthorized access without requiring predetermined labels. To find patterns of typical behavior, the system gathers login information such as time, IP address, device type, and frequency of access(Encyclopedia, 2025). K-Means Clustering clusters students based on typical login behaviors, such as accessing the portal from familiar locations during class hours. If a student logs in on many devices simultaneously, from strange places, or at strange times, the system has detected an abnormality. The DBSCAN also detects irregularities, such as many student accounts rapidly logging in from the same IP address, which may indicate credential sharing or hacking incidents. These techniques allow the portal system to automatically detect anomalies, enhancing cybersecurity and preventing unwanted access.

Advantages of Unsupervised Learning for Anomaly Detection

1. There's no Need for Labelled Data: Unsupervised learning models are perfect in scenarios where labelled anomalies are hard to come by because they can identify anomalies without the need for a pre-labelled dataset.
2. Adaptability to New Patterns: By continuously learning from new data, these models can identify anomalies that were previously undetected and adjust to changing system behaviors.
3. Effective for Large Datasets: Unsupervised learning techniques, such as clustering and probabilistic models, handle large and complex datasets efficiently without manual annotation.
4. Flexibility in Anomaly Detection: Algorithms such as K-Means, DBSCAN, PCA, and GMM can detect point, contextual, and collective anomalies, among other kinds of anomalies.
5. Reveals Hidden Structures in Data: Revealing underlying patterns in data, methods such as Principal Component Analysis (PCA) and Gaussian Mixture Models (GMM) increase the accuracy of anomaly identification.

6. Beneficial for Real-Time Monitoring: A lot of unsupervised learning models are capable of real-time operation, identifying irregularities in dynamic settings like fraud detection, network security, and system performance monitoring.
7. Scalability: Unsupervised learning adapts well to new and growing datasets without needing further human interaction, in contrast to supervised learning, which necessitates ongoing data labelling.

Application of Unsupervised Learning for Anomaly Detection

1. Identifying unusual patterns in student behavior, such as repeated unsuccessful login attempts, within a portal network.
2. Identifying sudden surges in network traffic that may indicate a cyberattack.

Limitations

1. Can have high false positive rates, especially if the normal behavior is highly variable.
2. Less effective for real-time detection without frequent model updates.

Deep Learning

Deep learning, a subset of machine learning, utilizes artificial neural networks (ANNs) to identify complex patterns and detect anomalies in large, unstructured datasets. Autoencoders, an unsupervised deep learning technique, compress and reconstruct normal data patterns; if a data point cannot be accurately reconstructed, it is flagged as an anomaly. This helps in detecting unauthorized access and fraudulent transactions in portal networks. Long Short-Term Memory Networks (LSTMs), a form of recurrent neural network (RNN), assess sequential data to anticipate expected values, flagging deviations as anomalies, useful for recognizing suspicious login activities and network traffic spikes. Generative Adversarial Networks (GANs) employ a generator-discriminator framework to model normal data distributions; significant deviations indicate anomalies, aiding in the detection of fraud and cybersecurity threats within educational portal systems (Analysis & Vision, 2019).

Advantages of Deep Learning for Anomaly Detection

1. Automatic Feature Extraction: Deep learning eliminates the need for manual feature engineering by automatically extracting relevant characteristics from raw data in comparison with typical machine learning models.

2. Detecting anomalies in huge, high-dimensional datasets with complex patterns that are too challenging for conventional algorithms is a strength of deep learning models.
3. Effective for Sequential and Time-Series Data: Models such as LSTMs are perfect for identifying anomalies in system logs, network traffic, and user activity trends since they are well-suited for analyzing sequential data.
4. Adaptability to Changing Data: Deep learning models are useful in dynamic settings where usage patterns change, such as university portal networks, because they can continuously learn and adjust to new behaviors.
5. Robustness to Noisy Data: Deep learning algorithms can handle unstructured and noisy data, which is common in real-world applications like payment transactions, cybersecurity monitoring, and portal access logs.
6. Scalability for Big Data: Deep learning techniques, particularly Autoencoders and GANs, can process large amounts of data, which makes them ideal for cloud-based networks and large-scale enterprise systems.
7. Reduced False Positives: Unlike rule-based systems that might flag normal variations as threats, deep learning models improve anomaly detection precision, reducing needless alerts.
8. Versatile Application Across Domains: Deep learning works well for a variety of anomaly detection tasks, such as fraud detection, intrusion detection, performance monitoring, and security threat detection.

Application of Deep Learning for Anomaly Detection

1. Detecting fraud in payment systems by modeling normal transaction behaviors.
2. Identifying security threats in university networks by analyzing system access patterns.

Limitations

1. Requires large datasets
2. High computational power.

Hybrid Approaches

A hybrid anomaly detection system combines several types of approaches, including traditional methods (rule-based), statistical analysis, machine learning (ML), and optimization algorithms, to enhance the accuracy, efficacy, and flexibility of detecting network anomalies. This approach minimizes the drawbacks of each technique while maximizing its benefits(Yuan, 2022).

Literature Review: Methodology

This review explores the foundational concepts of hybrid methods in detection systems, emphasizing their role in balancing complementary strengths and mitigating individual limitations. Hybrid approaches integrate multiple detection techniques, such as rule-based systems and machine learning models, to enhance accuracy, adaptability, and real-time responsiveness.

Rule-based methods, while efficient and straightforward to deploy, often struggle with identifying novel or evolving threats due to their reliance on predefined patterns. Conversely, machine learning models offer greater flexibility in detecting unforeseen anomalies but demand extensive training data and may introduce computational complexity. By combining these methodologies, hybrid systems achieve a more robust framework: they maintain the precision of rule-based detection for known threats while leveraging machine learning's predictive capacity to uncover emerging risks. Such integration is particularly valuable in educational portal system network contexts, where scalable, real-time threat detection must accommodate both predictable vulnerabilities and dynamic, innovative attack strategies. This synergy positions hybrid systems as a promising solution for safeguarding digital infrastructures without sacrificing efficiency or adaptability.

A hybrid anomaly detection system follows a structured workflow to efficiently identify and mitigate security threats in a network. The process begins with data collection, where network logs, traffic data, and system events are gathered. This data includes essential features such as IP addresses, timestamps, packet sizes, protocol types, and user activity logs, providing a foundation for further analysis(Green et al., 2025).

Next, the system undergoes preprocessing, which involves feature extraction to convert raw traffic data into meaningful attributes. feature selection is employed with variance thresholding, a simple but powerful feature selection technique, because of its ability to eliminate near-constant variables especially when the threshold is set to 0.01, retaining six critical features: protocol type, source/destination IPs, source/destination port numbers, and attack labels as detailed in Appendix A. This step ensures that only the most relevant data is used for anomaly detection.

The initial detection layer utilizes signature-based rules to identify known attack patterns, such as SQL injections and DDoS attacks. Simultaneously, statistical methods establish a baseline for normal network behavior, allowing deviations to be flagged as potential anomalies. This dual-layer approach improves the system's ability to recognize both pre-defined threats and unusual patterns.

The next phase involves machine learning-based analysis, where unsupervised learning techniques cluster network traffic data to detect suspicious activities without requiring labeled data. To improve performance, optimization and fine-tuning techniques such as Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and the Lion Optimization Algorithm (LOA) are used to adjust detection parameters dynamically. Additionally, dynamic threshold adjustment minimizes the false positive rate, ensuring that alerts are only triggered for genuine threats.
Hybrid anomaly detection-based systems (HADBS) have been widely explored in network security due to their ability to combine complementary detection techniques. However, despite their practical advantages, these systems present several challenges.

First, the integration of multiple detection models can substantially increase system complexity and computational overhead. This added intricacy may hinder deployment in resource-constrained environments. Second, optimizing the decision boundary threshold, balancing exploration (detecting novel anomalies) and exploitation (leveraging known patterns), poses a significant challenge. Overly sensitive models may lead to elevated false positive rates, particularly when parameter tuning is not meticulously calibrated. Finally, ensuring seamless compatibility and real-time performance across heterogeneous detection mechanisms remains a persistent issue, especially in large-scale or highly dynamic networks. Scalability and latency concerns may arise when attempting to maintain responsiveness across distributed or evolving infrastructures.

Addressing these limitations is critical to advancing the practicality and reliability of HADBS in real-world network environments.

Ultimately, this literature review serves as a gateway into the diverse world of hybrid anomaly detection. It not only explains the methodologies used to combine different techniques but also highlights the challenges and future directions that researchers are likely to explore as they strive to build more robust and versatile security systems for portal system networks.
Table 1 shows the pros and cons of some HADBS reviewed for this work.

Table 1: Pros and cons of HADBS reviewed

S/N	Author(s)	Methodology	Merit	Limitation
1	Rasim & Fargana, 2019	The proposed methodology employs a two-phase feature selection approach, where Particle Swarm Optimization (PSO) first identifies essential features while addressing dataset imbalance, followed by Ant Colony Optimization (ACO) to extract high-information, low-correlation features. For classifier optimization, a Genetic Algorithm (GA) fine-tunes ensemble models, including XGBoost and SVM, with the system evaluated on NSL-KDD, UNSW-NB15, and CSE-CIC-IDS2018 datasets to validate performance.	The system achieves 97.87%, 92.63%, and 90.38% of accuracy on CSE-CIC-IDS2018, UNSW-NB15, and NSL-KDD datasets, respectively,	Reliance on a metaheuristics generic detection threshold that requires manual tuning or modification. AUC-ROC validation remains unexplored.
2	Ugrenovic, & Tom , 2020	The paper presents an out-of-distribution (OOD) detection framework based on anomaly detection methods (One-Class SVM, Isolation Forest, Local Outlier Factor) and its ensembles, which are evaluated using softmax outputs from a DenseNet trained on CIFAR-10. The process entails training detectors on in-distribution data before testing on synthetic/noisy OOD datasets.	The ensembles model achieves 95% accuracy on UNSW-NB15, and NSL-KDD datasets. Practicality: Reduces OOD data requirements for training.	Unmeasured AUC-ROC performance, architecture-dependent softmax reliance, and high ensemble computational costs are all important drawbacks.
3	Pu et al., 2021	The SSC-OCSVM model detects anomalies by dividing high-dimensional data into 2D subspaces via Sub-Space Clustering (SSC). One-Class SVM identifies deviations in each subspace, while Evidence Accumulation (EA) aggregates results, flagging data points as anomalies if they are repeatedly marked as suspicious, enabling improved detection of rare or subtle threats without the need for labeled data.	SSC-OCSVM excels at detecting rare attacks like user-to-root(U2R) and remote-to-local (R2L), with up to 90% accuracy and a low false positive rate, even in mixed-attack situations	The model relies on a fixed threshold, requiring manual tuning across datasets and network conditions.
4	Danijela Protic & Miomir Stankovic, 2022	The WK-FNN model employs two parallel classifiers: Weighted k-Nearest Neighbor (WK-NN) and Feedforward Neural Network (FNN) to analyze network traffic. A bitwise XOR detects disagreements between their outputs. The percentage of disagreements triggers alerts, categorized into five severity levels, ranging from negligible to high-priority, enhancing intrusion detection responsiveness.	Both classifiers demonstrated high accuracy across all daily datasets wk-NN and FNN classifiers leverage the strengths of both a memory-based learner and a neural model, enhancing classification robustness.	High computational cost, it is memory-intensive, as it requires storing and inverting large matrices during training, which can limit scalability.
5	Ahmed Jamal et al., 2023	This review examines how machine learning and deep learning techniques enhance the security of Cyber-Physical Systems (CPS). Using a literature-based approach, this analysis examines models such as K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) for threat detection. It also explores CPS security frameworks,	The study highlights several merits of using ML for CPS security. LSTM demonstrated high accuracy, high reliability, and robustness in detecting a range of attacks like denial-of-service (DoS), false data injection (FDI), replay attacks, and time synchronization attacks.	The paper lacks metrics, benchmarking, and discussion on integration, scalability, and computational challenges.
6	Alqahtani & Alshaher, 2024		The Decision Tree outperformed ANN and SVM in intrusion detection, achieving high accuracy, F1-score, AUC, and low FPR.	Reliance on a generic detection threshold that requires manual tuning or modification.
7	Green et al., 2025		MLOA beats traditional methods with 97% recall, 98% accuracy, with ROC-AUC of 97%, and faster threat detection.	Higher computation limits real-time use and requires an optimal threshold

		including hybrid automata, multi-agent systems, and LSTM-based anomaly detection strategies. Alqahtani and AlShaher developed an anomaly-based intrusion detection system with Decision Tree, SVM, and ANN. The algorithm, which has been trained on over 172,000 network traffic records (Kaggle dataset), classifies behaviour as normal, suspect, or unknown. Feature selection and preprocessing increased detection accuracy, hence improving cybersecurity threat identification in dynamic networks. This study proposes a Modified Lion Optimization Algorithm (MLOA) combined with OCSVM to detect cyber threats like DoS and MitM (U2R &R2L) attacks in portal systems. Using datasets from LASUED and UNSW-NB15, it employs dynamic threshold tuning, adaptive feature selection, and real-time monitoring to enhance detection accuracy and resilience against evolving threats.		beyond parameter optimization.

Challenges in Hybrid Anomaly Detection Systems (HADBS)

Despite their potential, hybrid anomaly detection systems (HADBS) face significant challenges, including reliance on static detection thresholds requiring manual tuning, high computational overhead from complex architectures, and limited adaptability to evolving attack patterns. The absence of standardized performance metrics like AUC-ROC further complicates objective model evaluation, while scalability remains constrained by the real-time processing demands of large-scale networks. These limitations underscore the need for more adaptive, efficient frameworks that balance detection accuracy with computational feasibility in dynamic portal environments.

Proposed Solution: Enhanced MLOA Using Sigmoid-Based Age Ratio

To address the threshold sensitivity and optimization challenges in existing HADBS, we propose an Enhanced Modified Lion Optimization Algorithm (EMLOA) by integrating a sigmoid-based Age Ratio function into the generic Lion Optimization Algorithm (LOA) objective function.

The objective function (OF) equation for the LOA is as stated in equations 4 *(Rajakumar, 2012)*

$$f\left(X^{pride}\right) = \frac{1}{2(1+\| X^{m\_cub} \|)} [f(X^{male}) + f(X^{female}) + \frac{Age_{mat}}{age(cub)+1} \sum_{C=1}^{\|X^{m_{cubs}}\|} \frac{f\left(X_c^{m_{cubs}}\right) + f\left(X_c^{f_{cubs}}\right)}{\| X^{m_{cub}} \|)}] \quad (4)$$

Enhancing the MLOA: Modifying the generic objective function of the LOA, defined in equation (4), by replacing the Age Ratio equation (5) with a sigmoid-based Age Ratio function;

$$\text{Age Ratio} = \frac{Age_{mat}}{age(cub)+1} \quad (5)$$

The goal of the replacement is to achieve a smoother Age Ratio for equation 5, thereby improving the gradient behavior for both exploration and exploitation. The sigmoid function-based exponential decay is defined as;

$$f(x) = \frac{1}{1+e^{-x}} \quad (6)$$

The new Age Ratio is presented in equation 7:

$$Age\ ratio = \frac{1}{1+e^{-\lambda(age(cub)-Age_{mat})}} \quad (7)$$

Where $\lambda \in [0.1,1]$ is Suitable for exploration and $\lambda \in [6,10]$ for exploitation. The Python implementation of the process for the sigmoid-based Age Ratio function for the EMLOA is detailed in Appendix B.

The flowchart in Figure 1 shows the process of enhancing the MLOA parameters using the sigmoid-based Age Ratio function.

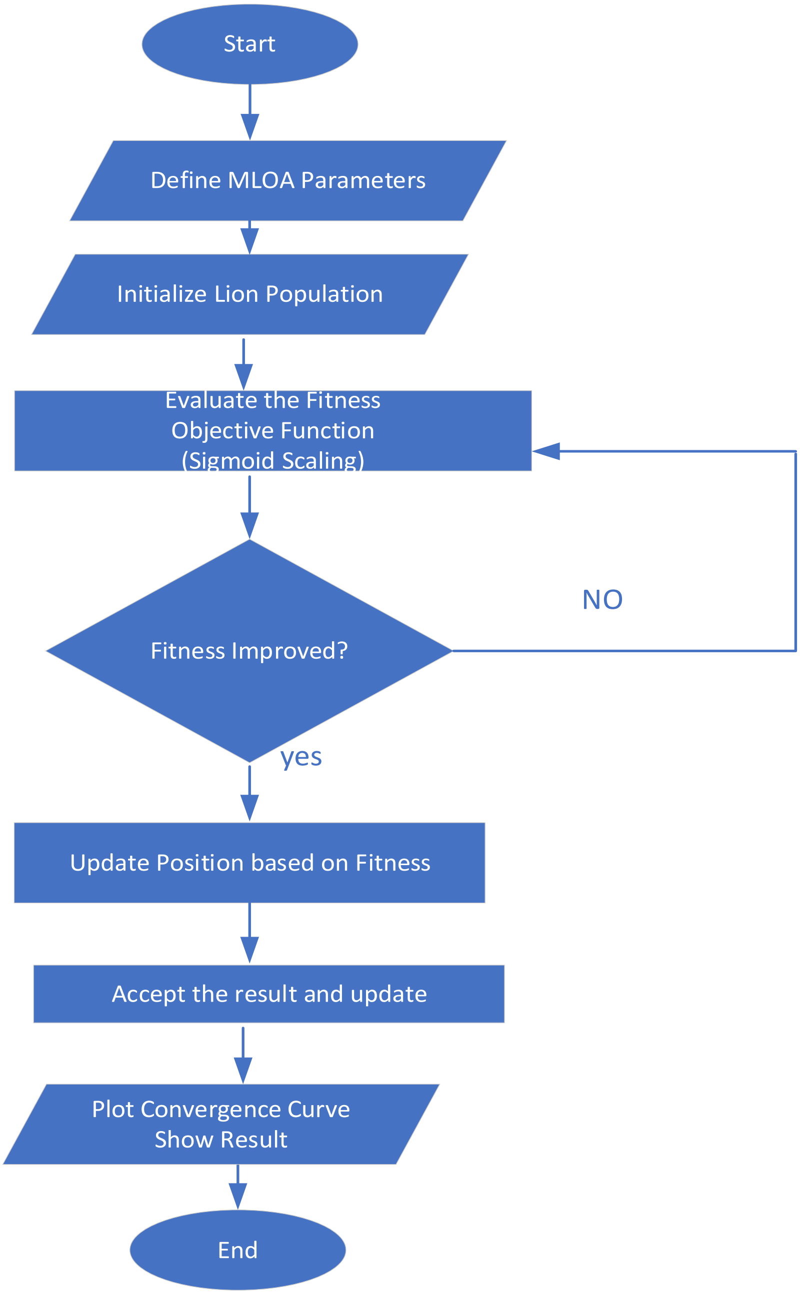

Figure 1: Enhanced MLOA Flowchart

Results

The result obtained from Appendix B is used for the validation of the optimal threshold decision boundary's exploration and exploitation of the OCSVM-based anomaly detection model for evolving threats in a portal system network in real time, as detailed in Appendix C.

Evaluation of the Enhanced MLOA

The evaluation of the Enhanced MLOA (EMLOA) involved calculating the best fitness of the lion (solution) and the best position of the lion (solution) using a Python-implemented objective function based on the sigmoid-based Age Ratio function integration for EMLOA Equation (7).

The resulting convergence behavior, visualized using the Matplotlib library, is demonstrated in Figures 2 and 3 for Enhanced MLOA (EMLOA) and Generic LOA, respectively. The best fitness value of approximately 0.38 for EMLOA and the Generic LOA of 0.95 shows that EMLOA significantly outperformed the Generic LOA.

As shown in Table 2, the cross-validation results (Appendix C) demonstrate that the chosen hyperparameter thresholds effectively optimize the OCSVM-based anomaly detection model. We evaluated model performance using accuracy, true positive rate (TPR), false positive rate (FPR), and AUC-ROC across varying best position parameter values while maintaining a constant fitness value of 0.38.

Table 2: Results of the Cross–Validation

Fitness Value (γ)	Best Position (ν)	Accuracy (%)	TPR (%)	FPR (%)	ROC-AUC
0.38	0.26	92.9	82.6	0.10	0.90
0.38	0.36	98.0	86.7	0.14	0.91
0.38	0.42	99.9	98.8	0.06	0.98
0.38	0.00093	80.6	80.2	0.08	0.89

Among the tested configurations, the model achieved optimal performance with a fitness value of 0.38 and a best position of 0.42, attaining 99.9% accuracy, a 98.8% true positive rate (TPR), a minimal false positive rate (FPR) of 0.06, and a near-perfect AUC-ROC score of 0.98. These results (Table 2, Appendix C) confirm that the selected hyperparameters significantly enhance model performance.

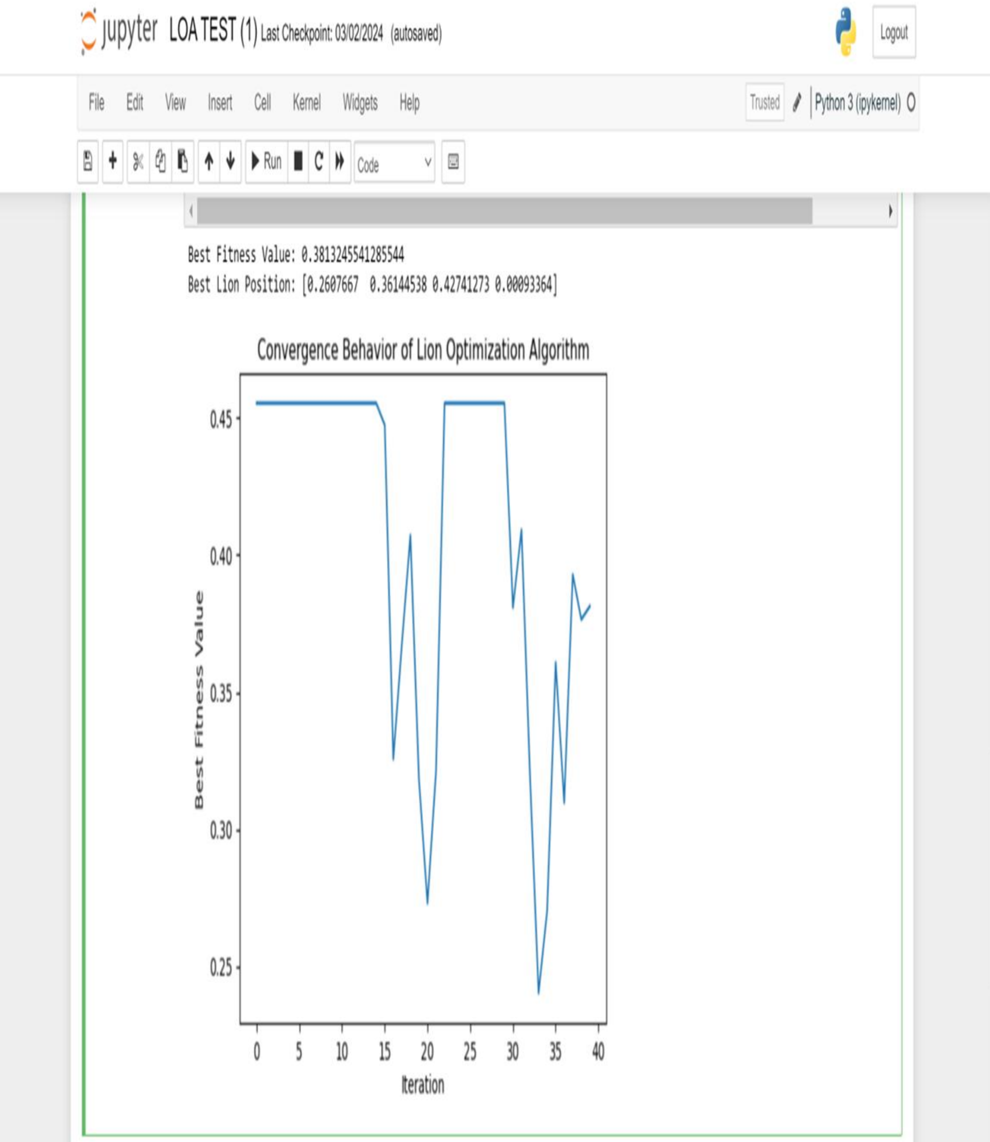

Figure 2: Convergence Behavior of the Enhanced MLOA

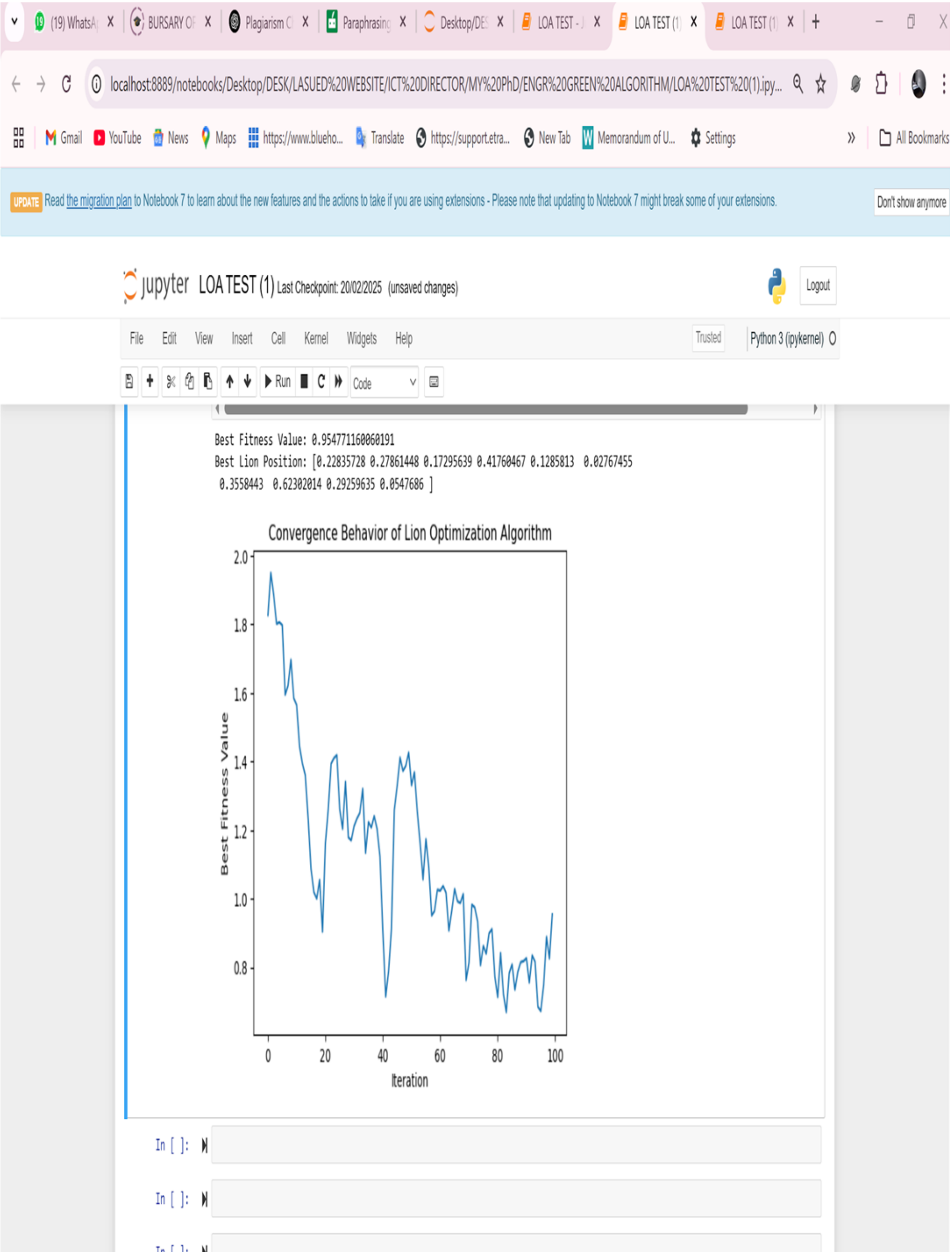

Figure 3: Convergence Behavior of the Generic Lion Optimization Algorithm

Comparative Analysis of the HADBS

Table 3 presents the comparative performance evaluation metrics of the EMLOA-OCSVM-based anomaly detection system and HADBS in the degree of Complexity level. The degree of Complexity (CL) is defined as Simple Anomalies (SA), Moderate Anomalies (MA), and High Anomalies (HA).

Table 3: Comparative Evaluation of EMLOA-OCSVM and HADBS

Matric	CL	Rasim & Fargana, 2019	Ugrenovic & Tom, 2020	Pu et al., 2021	Protic & Miomir, 2022	Ahmed Jamal et al., 2023	Alqahtani & Alshaher, 2024	Green et al., 2025	EMLOA-OCSVM-BASED MODEL
Recall (TPR)	SA	0.98	0.92	0.95	0.98	N/A	0.99	0.97	0.99
	MA	0.82	0.85	0.90	0.91	N/A	0.93	0.92	0.98
	HA	0.70	0.80	0.85	0.88	N/A	0.90	0.90	0.96
Accuracy	SA	0.98	0.95	0.96	0.99	N/A	0.99	0.98	0.99
	MA	0.92	0.90	0.92	0.92	N/A	0.94	0.94	0.98
	HA	0.85	0,80	0.87	0.85	N/A	0.90	0.91	0.96
Precision	SA	N/A	N/A	0.93	N/A	N/A	0.98	0.96	0.98
	MA	N/A	N/A	0.88	N/A	N/A	0.90	0.91	0.97
	HA	N/A	N/A	0.83	N/A	N/A	0.90	0.88	0.96
ROC-AUC	SA	N/A	N/A	0.94	N/A	N/A	0.99	0.97	0.99
	MA	N/A	N/A	0.90	N/A	N/A	0.94	0.94	0.97
	HA	N/A	N/A	0.85	N/A	N/A	0.91	0.92	0.95
F1 Score	SA	N/A	N/A	0.94	N/A	N/A	0.99	0.97	0.99
	MA	N/A	N/A	0.89	N/A	N/A	0.91	0.92	0.97
	HA	N/A	N/A	0.84	N/A	N/A	0.90	0.90	0.96
Training Time (s)	SA	N/A	N/A	9.65	N/A	N/A	NILL	14.23	3.05
	MA	N/A	N/A	10.12	N/A	N/A	NILL	12.89	5.01
	HA	N/A	N/A	11.03	N/A	N/A	NILL	13.67	7.02
Prediction Time (s)	SA	N/A	N/A	0.11	N/A	N/A	NILL	0.30	0.04
	MA	N/A	N/A	0.21	N/A	N/A	NILL	0.34	0.13
	HA	N/A	N/A	0.25	N/A	N/A	NILL	0.39	0.22

Application of the Hybrid Approach for Anomaly Detection Educational Portal Security

A hybrid anomaly detection model for education portal security combines signature-based techniques that detect known threats, such as credential reuse, with optimization anomaly-based techniques to identify unusual login behaviors, including students accessing the portal from different countries at unusual hours. By successfully tackling both known attacks, such as brute-force attempts, and previously unknown or insider threats, including unauthorized access by legitimate users, this integrated strategy enhances overall system protection.

Financial Fraud Detection

Banking systems employ a hybrid approach to financial fraud detection, using anomaly detection models to find previously unseen suspicious activities, like sudden high-value transfers from dormant accounts, and supervised models to identify known fraudulent transaction patterns. This approach enables real-time monitoring and response, improving the system's capacity to detect both established and emerging fraud schemes.

Payment Gateway System

In payment gateway systems, hybrid models integrate one-class SVMs to learn and model normal transaction behavior, while autoencoders detect deviations from these patterns as potential anomalies. Simultaneously, rule-based systems are employed to validate and flag known fraud scenarios. This layered approach enhances the detection of evolving fraud strategies while maintaining the accuracy and efficiency needed to avoid disrupting legitimate user transactions.

Cloud and IoT security

In cloud and IoT security, hybrid models combine anomaly-based detection to identify unusual resource usage or abnormal device communication patterns with signature-based techniques that recognize known malware signatures. This dual-layered approach provides robust protection across distributed and resource-constrained environments, ensuring that both known threats and emerging, unpredictable attacks are effectively managed and mitigated.

Network Traffic Monitoring

In network traffic monitoring, a hybrid approach combines unsupervised clustering techniques to detect previously unknown traffic anomalies with supervised classification models trained on known attack patterns. This integration enables the system to proactively defend against common threats like DDoS attacks while also identifying emerging traffic manipulation tactics, enhancing overall network resilience and security.

Discussion

Comparative analysis demonstrates that the proposed EMLOA-OCSVM model surpasses existing state-of-the-art methods across all key performance metrics and attack categories. The model achieves consistently high performance, with recall (up to 0.99), accuracy (up to 0.99), precision (up to 0.98), and ROC-AUC (up to 0.99), showcasing its robustness in detecting both simple and highly sophisticated network intrusions. Notably, it excels in identifying complex attacks (HA) like U2R and R2L, attaining a recall and precision of 0.96, significantly higher than previously reported models. Additionally, the model exhibits substantially reduced training and prediction times, making it well-suited for real-time intrusion detection. These results underscore the efficacy of integrating the Enhanced Modified Lion Optimization Algorithm (EMLOA) with OCSVM for hyperparameter optimization, leading to enhanced generalization, faster convergence, and more reliable anomaly detection in dynamic network environments.

Result Analysis Summary

Using its superior scalability (maintaining 0.96 recall/accuracy on complex HA attacks), adaptability (consistent performance across SA/MA/HA scenarios), efficiency (fastest training/prediction times at 3.05s/0.04s for SA), and interpretability (clear OCSVM decision boundaries with transparent EMLOA optimization), the results show that the EMLOA-OCSVM model advances anomaly detection. It combines dynamic hyperparameter tuning using EMLOA with the explainability of OCSVM, achieving real-time capability without sacrificing performance or traceability, which makes it especially appropriate for security-critical deployments. This is in contrast to previous models that are vulnerable to complex attacks.

Limitation

While the EMLOA-OCSVM model exhibits notable advancements, there are concerns regarding generalizability because benchmark datasets like UNSW-NB15 and LASUED logs do not adequately represent the complexity of various portal network setups.

Future Works

Future work should integrate explainable AI (XAI) techniques into hybrid anomaly detection models to enhance interpretability and adaptability. This will provide administrators with clear insights into detected threats while maintaining detection accuracy, enabling more informed security decisions in portal networks.

Conclusion

This review highlights the vital importance of early anomaly detection for securing portal system networks, demonstrating that while traditional methods remain limited against evolving threats, hybrid approaches - particularly the EMLOA-OCSVM model - offer superior performance with 99.9% accuracy, rapid processing (3.05s training), and minimal false positives. Despite these advances, challenges in complexity, scalability, and real-world adaptability must be addressed through improved dynamic thresholding, enhanced interpretability, and computational optimization to ensure robust deployment across diverse network environments

Conflict of Interest: The authors reported no conflict of interest.

Data Availability: All data are included in the content of the paper.

Funding Statement: The authors did not obtain any funding for this research.

References:

1. Ahmed Jamal, A., Mustafa Majid, A. A., Konev, A., Kosachenko, T., & Shelupanov, A. (2023). A review on security analysis of cyber physical systems using Machine learning. *Materials Today: Proceedings*, *80*(xxxx), 2302–2306. https://doi.org/10.1016/j.matpr.2021.06.320
2. Alqahtani, A., & Alshaher, H. (2024). Anomaly-Based Intrusion Detection Systems Using Machine Learning. *Journal of Cybersecurity and Information Management*, *14*(1), 20–33. https://doi.org/10.54216/JCIM.140102
3. Analysis, I., & Vision, C. (2019). Computing Anomaly Score Threshold with Autoencoders Pipeline. *Computing Anomaly Score Threshold with Autoencoders Pipeline*, *11401*, 1–16. https://doi.org/https://doi.org/1 0.1 007/97 8-3-030-1 3469-3_28
4. Aug, L. G. (2023). *Semi-supervised detection of structural damage using Variational Autoencoder and a One-Class Support Vector Machine ⋆*. *2023*, 1–32. https://doi.org/10.1109/ACCESS.2023.3291674
5. Bablu, T. A. (2025). *Machine Learning for Anomaly Detection : A Review of Techniques and Applications in Various Domains*. *07*(February), 1–17. https://doi.org/https://www.researchgate.net/publication/389038707_Machine_Learning_for_Anomaly_Detection_A_Review_of_Techniques_and_Applications_in_Various_Domains
6. Bin Yu, Zongzheng Zhang, Wenshu Xie, Wenjia Zuo, Yiming Zhao, and Y. W. (2023). *Gaussian Mixture Model*. 1–9. https://doi.org/doi.org/10.3390/electronics12061397
7. Cipra, T. (2020). *Box Jenkins Methodology ARIMA Model*. https://link.springer.com/chapter/10.1007/978-3-030-46347-2_6#citeas
8. Claudius, M., & Andersen, J. P. (2022). *Unsupervised Learning The basics , K-Means , DBScan What is Unsupervised Learning (UL)* (02.11.2021 03.11.2020 revised 25.04.2021 & 06.04.2022 (eds.); pp. 1–20). https://micl-easj.dk/Machine Learning/Overheads/ML Slides Chapter 9 Unsupervised Learning.pdf

9. Danijela Protic, Miomir Stankovic, V. A. (2022). Wk-fnn design for the detection of anomalies in the computer network traffic. *Facta Universitatis - Series: Electronics and Energetics*, *35*(2), 269–282. https://doi.org/10.2298/fuee2202269p
10. Darveau, K., Hannon, D., & Foster, C. (2020). A comparison of rule-based and machine learning models for classification of human factors aviation safety event reports. *Proceedings of the Human Factors and Ergonomics Society*, *64*(1), 129–133. https://doi.org/10.1177/1071181320641034
11. Dejana Ugrenovic, Jens Vankeirsbilck, D. P., & Tom Holvoet, jeroen B. (2020). Designing Out-of-distribution Data Detection.pdf. *International Scientific Conference Electronics*, 16–18. https://doi.org/978-1-7281-7426-6/20/$31.00
12. Duffield, N., Haffner, P., Krishnamurthy, B., & Ringberg, H. (2009). Rule-based anomaly detection on IP flows. *Proceedings - IEEE INFOCOM*, *June*, 424–432. https://doi.org/10.1109/INFCOM.2009.5061947
13. Emesoronye, S. (2024). *Rule-Based vs . Machine Learning- Based Cybersecurity : Understanding the Differences Limitations :* 1–4. https://doi.org/https://www.linkedin.com/pulse/rule-based-vs-machine-learning-based-cybersecurity-obinna-emesoronye-lnj1e/
14. Encyclopedia. (2025). *k -means clustering* (online, pp. 1–19). https://doi.org/https://en.wikipedia.org/wiki/K-means_clustering
15. Fernandes, G., Rodrigues, J. J. P. C., Carvalho, L. F., Al-Muhtadi, J. F., & Proença, M. L. (2022). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, *70*(3), 447–489. https://doi.org/10.1007/s11235-018-0475-8
16. Green, O. O., Abdulrazaq. M.B., Yahaya. B., Haruna. Z, Omogoye. S.O., Adegoke, A.S. (2025). *Enhancing Portal System Resilience with a Modified Lion Optimization Algorithm (MLOA) for Cyber Threat Detection*. *21*(2025), 61–85. https://doi.org/10.19044/esj.2025.v21n9p61
17. Hashmi, A., & Ahmad, T. (2020). *FAAD : A Self-Optimizing Algorithm for Anomaly Detection*. *17*(2), 272–280. https://doi.org/doi.org/10.34028/iajit/17/2/16
18. Li, L., Hansman, R. J., Palacios, R., & Welsch, R. (2016). Anomaly detection via a Gaussian Mixture Model for flight operation and safety monitoring. *Transportation Research Part C: Emerging Technologies*, *64*, 45–57. https://doi.org/10.1016/j.trc.2016.01.007
19. Mittal, A., Gupta, A., Bhoomi, & Agarwal, K. (2024). Anomaly Detection in Cybersecurity: Leveraging Machine Learning for Intrusion Detection. *Proceedings of International Conference on*

Communication, Computer Sciences and Engineering, IC3SE 2024, *December*, 331–335. https://doi.org/10.1109/IC3SE62002.2024.10592923

20. Moore, S. (2025). *Behavior Anomaly Detection : Techniques and Best Practices*. 1–11. https://doi.org/https://www.exabeam.com/explainers/ueba/behavior-anomaly-detection-techniques-and-best-practices
21. Paganini, P. (2023). *The University of Manchester suffered a cyber attack and suspects a data breach*. Security Affairs. https://i0.wp.com/securityaffairs.com/wp-content/uploads/2023/06/University-of-Manchester.png?ssl=1
22. Pu, G., Wang, L., Shen, J., & Dong, F. (2021). *A Hybrid Unsupervised Clustering-Based Anomaly Detection Method*. *26*(1007–0214), 146–153. https://doi.org/10 . 26599 /TST. 2019 . 9010051
23. Rajakumar, B. R. (2012). The Lion's Algorithm: A New Nature-Inspired Search Algorithm. *Procedia Technology*, *6*, 126–135. https://doi.org/10.1016/j.protcy.2012.10.016
24. Rasim M. Alguliyev, R. M. A., & Fargana J. Abdullayeva. (2019). PSO+K-means Algorithm for Anomaly Detection in Big Data. *STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING*, *7*, 348–359. https://doi.org/: 10.19139/soic.v7i2.623
25. Rousseeuw, P. J., & Hubert, M. (2018). Anomaly detection by robust statistics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *8*(2), 1–14. https://doi.org/10.1002/widm.1236
26. Trebar, M. (2021). *Analysis of Machine Learning Algorithms for Anomaly Detection on Edge Devices*. 1–22. https://doi.org/10.3390/s21144946%0AFaculty
27. Ukagwu, T. J. and L. (2023, March 15). Almost 13 million cyber attacks recorded during polls –FG. *Punch*, 1–3. https://cdn.punchng.com/wp-content/uploads/2023/02/16205355/ISA-PANTAMI.jpg
28. Yuan, Y. (2022). *A Modified Hybrid Method Based on PSO , GA , and K-Means for Network Anomaly Detection*. *2022*, 1–10. https://doi.org/10.1155/2022/5985426
29. Almufti, S. M. (2015). *U-Turning Ant Colony Algorithm powered by Great Deluge Algorithm for the solution of TSP Problem* [Eastern Mediterranean University]. https://doi.org/https://www.researchgate.net/publication/318014104_U-Turning_Ant_Colony_Algorithm_powered_by_Great_Deluge_Algorithm_for_the_solution_of_TSP_Problem

APPENDIX A

Pseudocode: Preprocessing + Feature Selection via Variance Thresholding

```
// --- PARAMETERS ---
SET UNSW_FILE_PATH ← "unsw_nb15.csv"
SET LASUED_LOG_FILE_PATH ← "lasued_network.csv"
SET VARIANCE_THRESHOLD ← 0.01
// --- FUNCTION: LOAD AND MERGE DATASETS ---
FUNCTION LoadAndMergeDatasets(file1, file2):
   LOAD unsw_df FROM file1
   LOAD lasued_df FROM file2
   SET common_columns ← INTERSECTION OF COLUMNS IN unsw_df AND lasued_df
   FILTER unsw_df TO common_columns
   FILTER lasued_df TO common_columns
   CONCATENATE unsw_df AND lasued_df INTO merged_df
   RETURN merged_df
END FUNCTION
// --- FUNCTION: ENCODE CATEGORICAL COLUMNS ---
FUNCTION EncodeCategorical(df):
   FOR EACH column IN df:
      IF column TYPE IS 'object' OR 'category':
         ENCODE column USING LabelEncoder
      END IF
   END FOR
   RETURN df
END FUNCTION
// --- FUNCTION: VARIANCE THRESHOLD FEATURE SELECTION ---
FUNCTION SelectFeaturesByVariance(df, threshold):
   INITIALIZE VarianceSelector WITH threshold
   APPLY VarianceSelector TO df → selected_features
   EXTRACT retained_columns WHERE VARIANCE > threshold
   CREATE reduced_df WITH selected_features AND retained_columns
   RETURN reduced_df
END FUNCTION
// --- MAIN PIPELINE FUNCTION ---
FUNCTION PreprocessPipeline(unsw_path, lasued_path, threshold):
   SET data ← LoadAndMergeDatasets(unsw_path, lasued_path)
   SET data ← EncodeCategorical(data)
   PRINT "Original Feature Count:", COLUMN_COUNT(data)
   SET reduced_data ← SelectFeaturesByVariance(data, threshold)
   PRINT "Reduced Feature Count:", COLUMN_COUNT(reduced_data)

   RETURN reduced_data
END FUNCTION

// --- MAIN EXECUTION ---
CALL PreprocessPipeline(UNSW_FILE_PATH, LASUED_LOG_FILE_PATH,
VARIANCE_THRESHOLD)
→ RETURNS reduced_features
```

SAVE reduced_features TO "reduced_features.csv"
PRINT "Reduced dataset saved as 'reduced_features.csv'"

Result Output: Input and Output Features

Input (x)	Output (y)
Protocol	Attack
Source IP	
Destination IP	
Source port	
Destination port	

APPENDIX B

Pseudocode: Enhanced Lion Optimization Algorithm with Dynamic λ

```
// --- PARAMETERS ---
SET POPULATION_SIZE ← 10
SET DIMENSION ← 4
SET MAX_ITERATIONS ← 40
SET LOWER_BOUND ← -10
SET UPPER_BOUND ← 10

// --- OBJECTIVE FUNCTION: Sphere Function ---
FUNCTION ObjectiveFunction(x):
   RETURN SUM(x_i^2 FOR EACH x_i IN x)
END FUNCTION

// --- SIGMOID-BASED AGE RATIO ---
FUNCTION SigmoidAgeRatio(age_cub, age_mat, λ):
   RETURN 1 / (1 + EXP(-λ * (age_cub - age_mat)))
END FUNCTION

// --- INITIALIZE LION POPULATION ---
FUNCTION InitializePopulation(size, dimension, lb, ub):
   CREATE population AS MATRIX OF size × dimension
   FOR EACH lion IN population:
      SET lion ← RANDOM_VECTOR_IN_RANGE(lb, ub, dimension)
   RETURN population
END FUNCTION

// --- DYNAMIC λ SCHEDULING ---
FUNCTION DynamicLambda(iteration, max_iterations):
   IF iteration < 0.3 × max_iterations:
      RETURN RANDOM(0.1, 0.5)
   ELSE IF iteration < 0.7 × max_iterations:
      RETURN RANDOM(0.5, 2)
   ELSE:
      RETURN RANDOM(4, 8)
END FUNCTION
```

```
// --- ENHANCED LOA MAIN FUNCTION ---
FUNCTION EnhancedLOA(pop_size, dim, max_iter, lb, ub):
   population ← InitializePopulation(pop_size, dim, lb, ub)
   fitness ← [ObjectiveFunction(ind) FOR EACH ind IN population]
   lion_ages ← [0 FOR i = 1 TO pop_size]
   best_fitness_history ← EMPTY LIST

   // Initialize Best Lion
   best_index ← INDEX_OF_MIN(fitness)
   best_lion ← COPY(population[best_index])
   best_fitness ← fitness[best_index]

   FOR iteration FROM 1 TO max_iter:
      age_median ← MEDIAN(lion_ages)
      λ ← DynamicLambda(iteration, max_iter)

      // Update Population
      FOR i FROM 1 TO pop_size:
         age_ratio ← SigmoidAgeRatio(lion_ages[i], age_median, λ)
         random_step ← RANDOM_VECTOR(-1, 1, dim)

         // New position influenced by age ratio and best lion
         population[i] ← population[i]
                      + age_ratio × random_step
                      + 0.1 × (best_lion - population[i])
         population[i] ← CLAMP(population[i], lb, ub)
         lion_ages[i] ← lion_ages[i] + 1
      END FOR

      // Re-evaluate Fitness
      fitness ← [ObjectiveFunction(ind) FOR EACH ind IN population]

      // Update Best Lion
      current_best_index ← INDEX_OF_MIN(fitness)
      IF fitness[current_best_index] < best_fitness:
         best_fitness ← fitness[current_best_index]
         best_lion ← COPY(population[current_best_index])
      END IF

      APPEND best_fitness TO best_fitness_history

      PRINT "Iteration", iteration, "/", max_iter,
            " | λ:", λ,
            " | Best Fitness:", best_fitness
   END FOR

   RETURN best_fitness_history, best_lion
END FUNCTION
```

```
// --- EXECUTION ---
CALL EnhancedLOA(POPULATION_SIZE, DIMENSION, MAX_ITERATIONS,
LOWER_BOUND, UPPER_BOUND)
→ RETURNS best_fitness_history, best_lion

// --- OUTPUT RESULTS ---
PRINT "Final Best Fitness:", LAST_ELEMENT(best_fitness_history)
PRINT "Best Lion Position:", best_lion

// --- CONVERGENCE PLOT (CONCEPTUAL) ---
PLOT best_fitness_history WITH:
   TITLE "Fitness Convergence Curve"
   X-AXIS LABEL "Iteration"
   Y-AXIS LABEL "Best Fitness (Log Scale)"
   APPLY LOG SCALE TO Y-AXIS
   ENABLE GRID
END PLOT
```

APPENDIX C
Validation code of the optimal threshold decision boundary's exploration and exploitation

```
Step 1: Define parameter sets
SET params TO [
   {gamma: 0.38, nu: 0.26},
   {gamma: 0.38, nu: 0.36},
   {gamma: 0.38, nu: 0.42},
   {gamma: 0.38, nu: 0.00093}
]

// Step 2: Initialize an empty list to store results
SET results TO empty list

// Step 3: Loop through each parameter set
FOR EACH param IN params DO:

   // Step 3.1: Train One-Class SVM model
   INITIALIZE model WITH OneClassSVM USING gamma = param.gamma AND nu =
param.nu
   FIT model ON X_train_scaled

   // Step 4: Make predictions on test data
   SET y_pred TO model.predict(X_test_scaled)

   // Step 4.1: Convert predictions from {-1, 1} to {1, 0}
   FOR EACH value IN y_pred DO:
      IF value == -1 THEN
         REPLACE value WITH 1  // Anomaly
      ELSE
         REPLACE value WITH 0  // Normal
```

```
    END IF
  END FOR

  // Step 5: Evaluate performance metrics
  SET confusion_matrix TO COMPUTE_CONFUSION_MATRIX(y_test, y_pred)

  SET TP TO confusion_matrix[1][1]
  SET FN TO confusion_matrix[1][0]
  SET FP TO confusion_matrix[0][1]
  SET TN TO confusion_matrix[0][0]

  IF (TP + FN) > 0 THEN
    SET TPR TO TP / (TP + FN)  // True Positive Rate
  ELSE
    SET TPR TO 0
  END IF

  IF (FP + TN) > 0 THEN
    SET FPR TO FP / (FP + TN)  // False Positive Rate
  ELSE
    SET FPR TO 0
  END IF

  SET ROC_AUC TO COMPUTE_ROC_AUC(y_test, y_pred)

  // Step 6: Store result in the list
  APPEND {
    gamma: param.gamma,
    nu: param.nu,
    TPR: ROUND(TPR, 4),
    FPR: ROUND(FPR, 4),
    ROC_AUC: ROUND(ROC_AUC, 4)
  } TO results

END FOR

// Step 7: Display results
FOR EACH res IN results DO:
  PRINT res
END FOR
```