Revista Brasileira de Computação Aplicada, July, 2026

DOI: 10.5335/rbca.v18i2.17117
Vol. 18, Nº 2, pp. 32–48
Homepage: seer.upf.br/index.php/rbca/index

ORIGINAL PAPER

Anomaly Detection Monitoring: A Proactive Approach with Deep Learning Integration

Darlan Noetzold[1,2], Valderi Reis Quietinho Leithardt[2,4], Jorge Luis Victória Barbosa[1], Anubis Graciela de Moraes Rossetto[3]

[1]University of Vale do Rio dos Sinos (UNISINOS), São Leopoldo, Brazil, [2]University Institute of Lisbon (ISCTE-IUL), ISTAR, Lisbon, 1649-026, Portugal, [3]Federal Institute of Education, Science and Technology Sul-rio-grandense (IFSul), Passo Fundo, Brazil, [4]Center of Technology and Systems (CTS-UNINOVA), Associated Laboratory of Intelligent Systems (LASI), 2829-516 Lisbon, Portugal

*dnoetzold@edu.unisinos.br; valderi.leithardt@iscte-iul.pt; jbarbosa@unisinos.br; anubisrossetto@ifsul.edu.br

Received: 2025-06-03. **Revised:** 2026-04-12. **Accepted:** 2026-07-06.

Abstract

This study develops an enhanced anomaly detection framework that integrates state-of-the-art deep learning techniques. The system detects hate speech, malicious websites, vulnerabilities in open ports, and suspicious processes. It optimizes detection accuracy and reliability by leveraging tailored datasets, advanced model architectures, and refined preprocessing strategies. To ensure robustness, it employs comprehensive validation metrics, including the Polygon Area Metric (PAM). Empirical results show significant improvements in generalization and prediction accuracy over previous methodologies. The hate speech detection model achieved an accuracy of 88%, with precision of 75%, recall of 62%, F1 score of 62%, and ROC AUC of 92%. These results deliver a proactive monitoring solution suited for modern enterprise security challenges.

Keywords: Anomaly Detection; Deep Learning; Machine Learning; Monitoring; Security

Resumo

Este estudo desenvolve um framework aprimorado para detecção de anomalias que integra técnicas de aprendizado profundo de última geração. O sistema é capaz de identificar discursos de ódio, sites maliciosos, vulnerabilidades em portas abertas e processos suspeitos. Ele otimiza a acurácia e a confiabilidade da detecção por meio do uso de conjuntos de dados personalizados, arquiteturas avançadas de modelos e estratégias refinadas de pré-processamento. Para garantir robustez, o sistema emprega métricas abrangentes de validação, incluindo a métrica da Área do Polígono (PAM). Os resultados empíricos demonstram melhorias significativas na capacidade de generalização e na acurácia das previsões em comparação com metodologias anteriores. O modelo de detecção de discurso de ódio atingiu uma acurácia de 88%, com precisão de 75%, recall de 62%, F1-score de 62% e ROC AUC de 92%. Esses resultados fornecem uma solução de monitoramento proativa, adequada aos desafios de segurança enfrentados por ambientes corporativos modernos.

Palavras-Chave: Aprendizado de Máquina; Aprendizado Profundo; Detecção de Anomalias; Monitoramento; Segurança

1 Introduction

The increased use of digital systems for security monitoring has created a need for more accurate and scalable anomaly detection mechanisms. Traditional solutions often fail to identify complex anomalies such as hate speech, malicious websites, suspicious processes, and applications running on open ports. Rossetto et al. (2024) introduced a microservices-based monitoring framework that employs spyware and predictive models to detect hate speech. While the system demonstrated significant performance improvements

through the adoption of the Quarkus framework, there are still opportunities to enhance the anomaly detection capabilities (Noetzold et al., 2025; Kumar et al., 2024).

The present work builds upon the aforementioned system by integrating advanced Machine Learning (ML) and Deep Learning (DL) techniques to improve anomaly detection and alert generation (Horowicz et al., 2024; Shayesteh et al., 2024). Specifically, the proposed approach addresses the detection of hate speech from textual data, identification of malicious websites and DNS queries, analysis of applications running on open ports using Nmap scanners, and detection of suspicious processes. These improvements are grounded in established methodologies and empirically validated techniques from the literature. For example, Tayyab et al. (2023) highlighted the vulnerabilities and challenges associated with DL algorithms in security contexts, while Lv et al. (2021) demonstrated the applicability of DL for addressing security concerns in heterogeneous networks. Additionally, Shinde and Shah (2018) reviewed the versatility of ML and DL frameworks in solving complex classification and detection problems across various domains (He et al., 2024).

The primary scientific contributions of this work include the integration of state-of-the-art DL models for anomaly detection, the implementation of enhanced validation metrics to ensure accuracy and robustness, and the evaluation of the proposed techniques through empirical testing. Unlike previous studies that rely on static rule-based detection or isolated deep learning models, this research introduces a novel adaptive anomaly detection framework that dynamically adjusts detection thresholds based on contextual factors and evolving data distributions (Savva et al., 2024). The proposed methodology integrates deep learning models with a probabilistic scoring mechanism, allowing real-time decision-making that balances sensitivity and specificity, thereby reducing false positives and improving detection rates. Moreover, this study formalizes a multi-source anomaly detection pipeline that unifies text-based, network-level, and system-level anomaly detection into a cohesive, end-to-end monitoring solution. This approach extends beyond simple model integration by introducing an adaptive monitoring layer capable of evolving with new threat patterns, ensuring long-term applicability in dynamic environments (Zhang et al., 2016; Azimi et al., 2020; Alom et al., 2019).

Integrating multiple anomaly detection techniques—such as hate speech detection, malicious website identification, open port vulnerability analysis, and suspicious process monitoring—aims to provide a comprehensive defense against cyber threats. This holistic approach enables the correlation of events across different domains, enhancing the accuracy in identifying malicious activities and reducing false positives. Recent studies have demonstrated that combining signature-based detection and behavioral analysis methods offers a more complete understanding of security events and improves threat detection capabilities (Ahmad et al., 2021).

The remainder of this paper is structured as follows: Section 2 discusses related works and provides an overview of recent advancements and methodologies in ML and DL for anomaly detection. Section 3 presents the methodology, detailing the proposed system architecture, data collection techniques, and model configurations. Section 4 discusses the results and evaluates the proposed system's performance. Finally, Section 5 concludes the paper by summarizing the findings and outlining directions for future research.

2 Related Works

Addressing the problem of anomaly detection, which includes hate speech identification, malicious websites, suspicious processes, and open port applications, has been the focus of several studies and commercial solutions. This section provides an overview of existing works while highlighting the gaps this study aims to address. Table 1 summarizes the main functionalities covered by related studies, highlighting their strengths and limitations. While some works, such as Rossetto et al. (2024), address multiple anomaly detection tasks, they lack deep learning-based monitoring integration. Other studies focus on specific threats, such as hate speech detection (Paschalides et al., 2020) or suspicious process monitoring (Bharathi et al., 2024), but do not provide a comprehensive solution. The proposed framework integrates all these functionalities, leveraging machine learning to enhance real-time monitoring and proactive anomaly detection.

Each column in Table 1 corresponds to a specific functionality or feature relevant to anomaly detection systems. The column **HSD** refers to Hate Speech Detection, indicating whether the study proposes or integrates techniques for identifying harmful or abusive language in textual content. **MWD** stands for Malicious Website Detection, representing mechanisms for identifying domains or URLs associated with phishing, malware distribution, or other threats. **SPD** corresponds to Suspicious Process Detection, which involves monitoring host or system-level processes to identify anomalous or unauthorized behavior. **OPD** indicates Open Port Detection, referring to the scanning or analysis of network ports to assess potential vulnerabilities or exposures. The **DL** column denotes whether Deep Learning techniques are employed for anomaly detection, rather than relying solely on rule-based or traditional machine learning methods. **PM** refers to Proactive Monitoring, which evaluates whether the solution anticipates threats based on predictive analytics rather than reactive responses. **RTA** captures Real-Time Analysis capabilities, essential for time-sensitive environments where immediate response is critical. **DevOps** integration refers to the system's compatibility with DevOps workflows, including observability pipelines and continuous deployment environments. Lastly, **XAI** (Explainable Artificial Intelligence) indicates whether the study employs interpretable models or provides explanations for anomaly detection decisions, which is essential for transparency and trust in critical systems.

The integration of four distinct anomaly domains—social (hate speech), web (malicious

Table 1: Comparison of Related Works Based on Functionalities

Reference	HSD	MWD	SPD	OPD	DL	PM	RTA	DevOps	XAI
Rossetto et al. (2024)	✓	✓	✓	✓	–	✓	✓	–	–
Noetzold et al. (2023)	✓	–	–	–	✓	–	–	–	–
Paschalides et al. (2020)	✓	–	–	–	–	–	–	–	–
Modha (2020)	✓	–	–	–	–	–	–	–	–
Bharathi et al. (2024)	–	–	✓	–	✓	–	–	–	–
Bauskar (2024)	–	–	–	–	✓	✓	✓	–	–
Mahida (2024)	–	–	–	–	✓	✓	✓	✓	–
Roshan and Zafar (2023)	–	–	✓	–	✓	–	✓	–	✓
Latif-Martínez et al. (2023)	–	–	✓	–	✓	✓	✓	–	–
Xie et al. (2023)	–	✓	–	✓	✓	✓	–	–	–
Tran and Sovilj (2024)	–	✓	–	–	✓	–	–	–	–
Hu and Yuan (2023)	–	✓	–	–	✓	–	–	–	–
Lee and Ram (2024)	✓	–	–	–	✓	–	–	–	–
Elzayady et al. (2023)	✓	–	–	–	✓	–	–	–	–
Alharbi et al. (2020)	–	–	–	–	✓	–	–	–	–
Ahmad et al. (2021)	–	–	–	–	✓	–	–	–	–
Panigrahi et al. (2021)	–	–	–	–	✓	–	–	–	–
Zhang et al. (2019)	–	–	–	–	✓	–	–	–	–
Proposed Work	✓	✓	✓	✓	✓	✓	✓	✓	✓

websites), network (open ports), and host (suspicious processes)—under a unified framework offers several advantages over isolated systems. This approach enables contextual information sharing across domains, allowing correlations between anomalies detected in different layers, which enriches threat identification. It also optimizes computational resources by consolidating monitoring infrastructure and feature extraction, improving scalability and efficiency. Furthermore, the unified system supports adaptive learning and dynamic thresholding informed by cross-domain feedback, enhancing detection accuracy in evolving threat landscapes. Finally, centralized management and DevOps integration streamline deployment and observability, making the solution more practical for real-time proactive monitoring. These factors collectively contribute to a more effective and resilient anomaly detection system than the sum of four independent solutions.

Rossetto et al. (2024) and Noetzold et al. (2023) introduced a monitoring framework that integrates spyware techniques and predictive models for detecting data leaks, suspicious behaviors, and hate speech. Their microservices-based system demonstrated significant performance improvements by reducing CPU and memory usage. However, their solution relied on blacklists for detecting malicious websites, vulnerabilities in open ports, and suspicious processes, which limits the ability to proactively identify new threats.

Paschalides et al. (2020) introduced a big data platform to monitor and detect hate speech online. The system uses ensemble-based classification algorithms and focuses on processing and visualizing statistical information. Similarly, Modha (2020) developed a tool to detect and visualize aggression in social media, utilizing browser-based plugins. Although effective for online platforms, these works are limited to text-based data and do not include network or process analysis.

Imam and Siregar (2021) explored spyware techniques, specifically keyloggers, for securing digital systems. Their research demonstrated how spyware can capture user activities to prevent unauthorized access. Similarly, Wulandari (2023) developed a spyware-based keylogger to monitor smartphone user interactions. These studies emphasize the dual-use nature of spyware, highlighting its utility for security purposes. Bharathi et al. (2024) proposed the InsightGuard system, which combines machine learning and computer vision to monitor screen usage and user posture. The automated alert mechanisms of the system improve user safety, but are not designed to detect network-level anomalies or malicious processes.

Bauskar (2024) investigates the application of machine learning techniques to enhance system observability, particularly in anomaly detection scenarios. His study emphasizes the role of predictive analytics in identifying system failures, intrusions, and performance degradation in real-time, thus enabling proactive monitoring strategies. Similarly, Mahida (2024) explores the integration of observability with DevOps practices in financial services technology, focusing on the impact of continuous monitoring in optimizing operational resilience and regulatory compliance. His work highlights key challenges such as security risks, data overload, and organizational adaptation required for effective implementation. Furthermore, Mahida discusses the future of observability, emphasizing the increasing role of artificial intelligence and machine learning in automating observability tasks, ultimately reinforcing the effectiveness of anomaly detection and system reliability.

Roshan and Zafar (2023) proposed an explainable anomaly detection framework that applies the KernelSHAP method to interpret the decision process of deep neural networks in cybersecurity contexts. Their approach emphasizes the need for model transparency, particularly in high-stakes applications like threat detection. The study evaluates suspicious process behavior and demonstrates the ability to isolate influential features contributing to classification

outcomes. Although the model includes real-time evaluation and explainable mechanisms, it does not incorporate network-layer threats, such as open port scanning or malicious domains, and lacks integration with broader monitoring architectures.

Latif-Martínez et al. (2023) introduced a Graph Neural Network (GNN)-based method for detecting contextual anomalies in network infrastructures. Their work focuses on capturing temporal and topological relationships among network nodes, which enhances the identification of stealthy or correlated threats. The study presents a real-time analysis engine and shows promising results for intrusion scenarios, including suspicious traffic behaviors. However, the scope remains confined to network anomalies, excluding textual data analysis such as hate speech, and does not integrate with process-level observability or DevOps practices.

Xie et al. (2023) presented a hierarchical spatiotemporal model for detecting Trojan traffic using deep learning. Their method leverages the spatial structure of packet flows combined with temporal behavior to identify anomalies in encrypted and obfuscated communication. The study demonstrates detection capabilities for malicious website activities and port-based attacks, offering a robust solution for traffic-level intrusion. Despite its strengths in analyzing low-level data streams, the framework is not designed for broader anomaly detection tasks such as behavioral analysis of processes or text-based threat identification like hate speech.

Tran and Sovilj (2024) proposed a malicious website detection framework using granular feature analysis in combination with machine learning classifiers. The system evaluates website attributes including DNS features, lexical patterns, and certificate metadata to enhance detection accuracy. While the authors apply deep learning models and demonstrate performance improvements, the framework is limited to static web indicators and does not address runtime anomalies, system-level monitoring, or adaptive scoring mechanisms.

Hu and Yuan (2023) conducted a comprehensive review of data-driven methods for detecting malicious websites. Their work categorizes a variety of machine learning and deep learning approaches, highlighting current challenges such as dataset imbalance and concept drift. Although the study provides a solid theoretical background and benchmarking insights, it does not contribute a unified system or implementation. It also does not address integration with real-time environments or proactive monitoring frameworks.

Lee and Ram (2024) introduced a personality-based model for hate speech detection using deep learning. Their framework incorporates psycholinguistic features derived from user-generated content to enhance classification performance. The study demonstrates that integrating inferred personality traits improves the model's ability to detect hate speech across various platforms. Although this approach contributes a novel angle to text-based threat detection, it does not consider other anomaly domains such as system behavior or network-based threats, and it lacks integration with real-time monitoring or explainable AI techniques.

Alharbi et al. (2020) conducted a broad survey on intrusion detection systems (IDS) within Internet of Things (IoT) environments, focusing on the use of machine learning and deep learning techniques. The review emphasizes architecture models, attack taxonomy, and the limitations of traditional IDS methods in dynamic contexts. While the study outlines several deep learning applications for anomaly detection, it lacks empirical validation and does not present an integrated monitoring solution. It also does not cover text-based threats such as hate speech or DevOps-oriented observability strategies.

Ahmad et al. (2021) provided a systematic study of network intrusion detection systems that utilize both traditional and deep learning approaches. Their analysis includes evaluations of supervised, unsupervised, and hybrid models applied to public datasets. The paper contributes insights into performance trade-offs and real-time capabilities. However, the study remains focused on network traffic and packet-level anomalies, omitting considerations of application-layer threats or integration with behavioral and textual analysis.

Panigrahi et al. (2021) offered a comprehensive overview of anomaly detection methods in IoT, covering both machine learning and deep learning strategies. Their taxonomy includes unsupervised learning, clustering, and ensemble methods, applied to sensor and network data. Although the paper discusses model architectures and application domains, it does not propose a unified framework or include modules for user behavior or language-based anomaly detection. The focus is technical, with less emphasis on proactive monitoring and system integration.

Zhang et al. (2019) compared multiple deep learning models for intrusion detection, including CNNs, LSTMs, and hybrid architectures. The study evaluates detection accuracy on benchmark datasets and provides recommendations on network structure optimization. While the experimental setup demonstrates the effectiveness of deep architectures, the work centers solely on network-level intrusions and omits broader anomaly categories such as process analysis, port scanning, or text-based abuse. The lack of integration with monitoring systems limits its practical deployment scope.

Compared to the existing literature, the proposed solution advances prior work by introducing an adaptive anomaly detection framework based on deep learning. Unlike Rossetto et al. (2024), which relies on static blacklist-based detection, this study implements a dynamic probabilistic scoring mechanism that continuously adjusts detection thresholds according to real-time data behavior. The framework not only identifies hate speech in textual content but also incorporates network traffic analysis for detecting malicious websites, automated scanning for open port vulnerabilities, and behavioral profiling for suspicious process monitoring. This unified approach goes beyond simple model integration by embedding an adaptive monitoring layer capable of evolving with emerging threats, thereby supporting long-term resilience in dynamic and heterogeneous security environments.

3 Methodology

This section details the methodology employed to implement the anomaly detection framework. The system is divided into distinct applications, each addressing a specific functionality: detection of hate speech, malicious websites, open port vulnerabilities, and suspicious processes. Each application uses tailored datasets, deep learning models, and hyperparameter optimization strategies to achieve accurate detection and prediction results.

Fig. 1 illustrates the comprehensive workflow of the proposed anomaly detection system, encompassing data preparation, model training, and real-time alert generation. The process begins with data collection and preprocessing, where raw datasets are cleaned, standardized, and transformed into a format suitable for deep learning models. Feature extraction techniques are then applied to identify relevant patterns in the data, optimizing its input for model training.

Next, the system selects and trains deep learning models using tailored architectures and hyperparameter optimization. These models undergo rigorous validation to ensure accuracy and reliability. To maintain relevance and adaptability, the models are retrained daily using updated datasets, integrating recent trends and patterns into their predictions.

Once the models are updated, the system transitions to real-time operations. Current data is collected from the monitoring infrastructure, preprocessed similarly to the training phase, and passed through the deployed deep learning models. These models classify the data to detect anomalies, generating alerts for identified threats, such as hate speech, malicious websites, vulnerabilities in open ports, or suspicious processes. This continuous cycle ensures robust detection and timely mitigation of potential risks.

Fig. 2 illustrates the detailed architecture of the proposed system. Data from diverse sources are collected and preprocessed through a unified pipeline, ensuring consistency across domains. Each specialized model processes its respective data stream, producing anomaly scores or classifications. These outputs are then fed into an integration module that correlates findings across domains, enhancing detection accuracy and reducing false positives. The integrated alerts are presented in a real-time monitoring interface, enabling prompt response and system management. This architecture supports continuous retraining and adaptation, maintaining system effectiveness in dynamic environments.

3.1 Datasets

The performance of deep learning models depends on the quality and relevance of the datasets used for training. Specific datasets were selected to address each functionality in the system. For hate speech detection, the Kaggle dataset (Kaggle, n.d.b) includes textual data labeled as hate speech, offensive language, and neutral content[1]. Its diversity enables the model to identify harmful content across different contexts.

The detection of open port vulnerability utilized the Common Vulnerabilities and Exposures (CVE) dataset (CVE, n.d.), which contains detailed records of known vulnerabilities, including severity levels and affected software[2], allowing an accurate prediction of network threats. The detection of malicious websites employed a DNS traffic dataset from Kaggle (Kaggle, n.d.a), providing labeled DNS requests to distinguish between benign and malicious domains[3]. Its realistic traffic patterns supported the model's training.

Suspicious process detection used the CICIDS2017 dataset (CICIDS, n.d.), comprising network flow data and process metadata from simulated cyberattacks[4]. Its comprehensive representation of normal and malicious activities facilitated effective anomaly detection. These datasets were chosen for their alignment with each functionality, ensuring reliable model training and evaluation.

To address the evolving nature of data, such as emerging slang in hate speech or new URL obfuscation techniques, the system employs a reactive adaptation strategy. This involves periodic updates of the training datasets with newly collected and labeled data to capture recent trends and threat patterns. Additionally, updated blacklists and heuristic rules are integrated to complement the machine learning models, enhancing detection of novel or obfuscated threats. This approach ensures that the framework remains effective over time despite changes in data distributions and adversarial tactics.

For the Open Port Vulnerability Detection, data is collected using `Nmap`, which provides detailed information about open ports, running services, protocols, and version banners. These outputs are parsed and structured into feature vectors containing port numbers, protocol types, service names, response times, and inferred software versions. The collected data is then matched against CVE identifiers to create labeled samples. For the Suspicious Process Detection, system telemetry data—including CPU usage, memory consumption, parent-child process relations, and execution time—is gathered using host-based monitoring tools such as `psutil` and `Auditd`. The data is transformed into time series sequences and correlated with the CICIDS2017 network flows to simulate realistic attack patterns. This comprehensive collection enables training on both static and behavioral indicators of malicious activity.

3.2 Data Preprocessing and Normalization

Effective preprocessing and normalization were applied to ensure clean, consistent, and standardized input data for training the models. The preprocessing pipeline was designed to handle multiple types of data—text, numerical values, and time-series data—ensuring optimal input

[1] https://www.kaggle.com/code/prnvthkkr/hate-speech-detection

[2] https://cve.mitre.org/

[3] https://www.kaggle.com/datasets/peterfriedrich1/dns-test-traffic-dohbrw2020

[4] https://www.unb.ca/cic/datasets/ids-2017.html

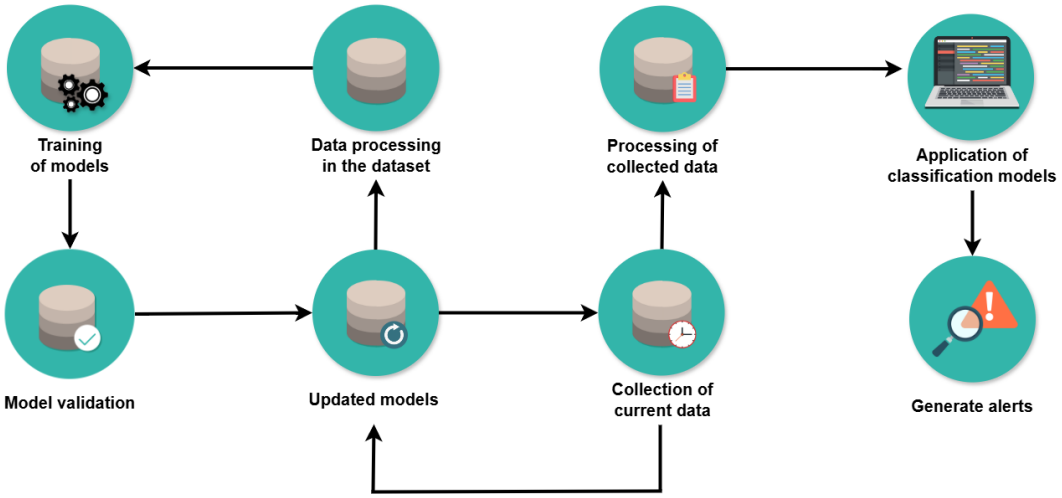


Figure 1: General workflow of the anomaly detection system.

representation across different functionalities. The key steps involved data cleaning, feature extraction, normalization, and balancing.

For the hate speech detection dataset, the preprocessing pipeline removes noise and standardizes input sequences. The steps include removing URLs, mentions (@username), special characters, and stopwords, followed by converting text to lowercase. Tokenization and lemmatization normalize word variations, and sequence padding ensures a fixed length of n_{max}. Mathematically, the transformation of each textual input sequence T into a numerical representation follows:

$$X_{text} = \text{Pad}(\text{Lemmatize}(\text{Tokenize}(T)), n_{max}) \quad (1)$$

where Tokenize(T) maps a sentence into a sequence of word tokens, Lemmatize($\cdot$) reduces words to their base form, and Pad($\cdot, n_{max}$) ensures a fixed sequence length of n_{max}. To enhance semantic understanding, the framework applies pre-trained word embeddings such as GloVe or Word2Vec, representing each token as a dense vector:

$$X_{text} = \sum_{i=1}^{n} \mathbf{E}(w_i) \quad (2)$$

where $\mathbf{E}(w_i) \in \mathbb{R}^d$ is the embedding of word w_i in a d-dimensional space.

For the CVE dataset related to open port vulnerabilities, numerical features such as Common Vulnerability Scoring System (CVSS) scores were normalized to ensure a uniform scale across different feature magnitudes. Missing values were imputed using the median of the respective feature distribution, while categorical variables were encoded using one-hot encoding. The numerical features were normalized using min-max scaling:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (3)$$

where $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of the feature.

For malicious website detection, the DNS traffic dataset underwent extensive preprocessing, which included the removal of duplicate queries, extraction of query types and domain lengths, and computation of entropy values from DNS requests. Categorical attributes, such as query types, were encoded using one-hot encoding, while numerical features were scaled using min-max normalization. The binary classification labels were encoded as:

$$y_i = \begin{cases} 1, & \text{if domain is malicious} \\ 0, & \text{if domain is benign} \end{cases} \quad (4)$$

For suspicious process detection, the CICIDS2017 dataset contained network traffic and system logs that required specialized feature engineering. The preprocessing steps included removal of irrelevant or incomplete records and selection of key features such as memory usage, CPU consumption, and process creation time. Data standardization was performed using zero mean and unit variance normalization:

$$X' = \frac{X - \mu}{\sigma} \quad (5)$$

where μ is the mean and σ is the standard deviation of the feature.

To address class imbalance, Synthetic Minority Over-sampling Technique (SMOTE) was applied to generate synthetic samples for underrepresented classes. The SMOTE algorithm generates new samples as follows:

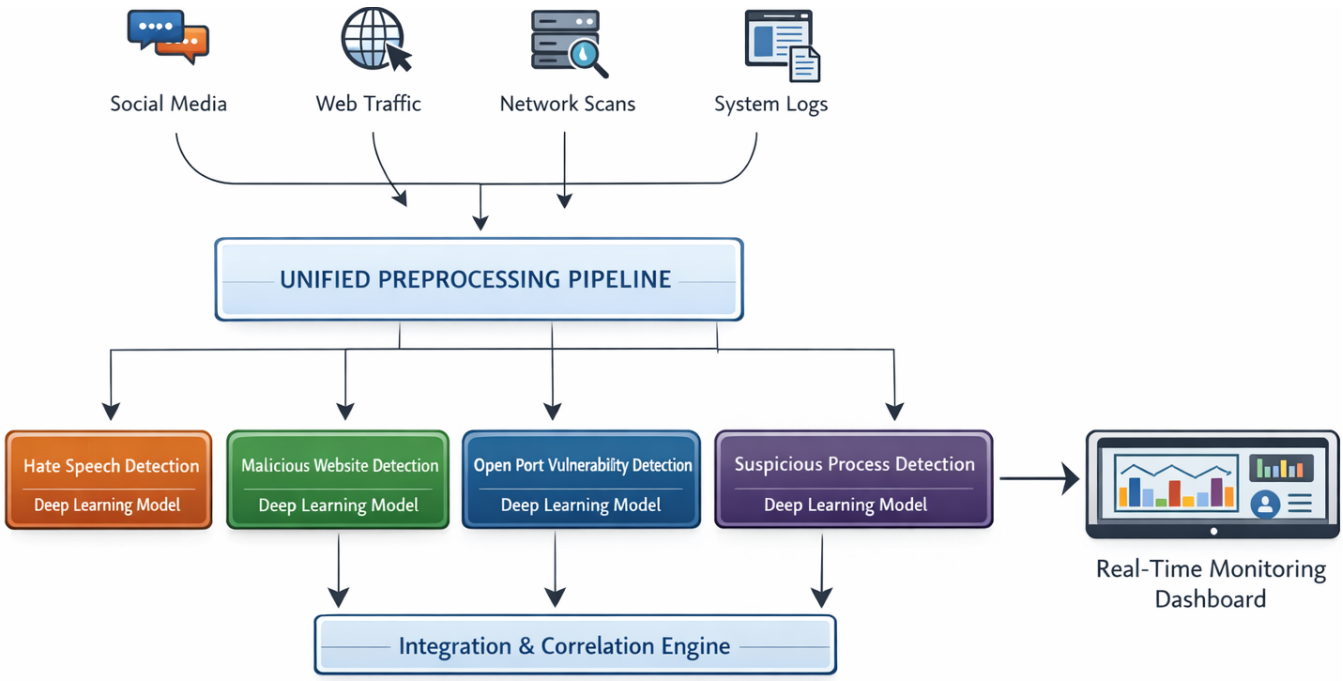


Figure 2: Detailed architecture of the integrated anomaly detection system showing real-time data flow and interaction between the four detection modules.

$$X_{new} = X_i + \lambda \cdot (X_j - X_i) \tag{6}$$

where X_i and X_j are two randomly selected minority class samples and λ is a random value in the range $[0, 1]$.

The complete preprocessing workflow for all datasets is formalized in Algorithm 1, ensuring consistency across different anomaly detection tasks. For Hate Speech Detection, text data is cleaned by removing special characters, URLs, and mentions, followed by tokenization, lemmatization, and embedding-based representation, with sequence padding for uniform input sizes. Open Port Vulnerability Detection involves imputing missing values, encoding categorical variables, and normalizing numerical features using min-max scaling to standardize data distribution. Malicious Website Detection applies redundant query filtering, extracts statistical features from domains (e.g., entropy and query types), encodes categorical attributes, and normalizes numerical values. Suspicious Process Detection selects relevant system metrics (e.g., CPU and memory usage), applies zero mean and unit variance standardization, and addresses class imbalance with SMOTE. This structured approach ensures that each dataset is well-formatted, optimizing feature representation for robust anomaly detection.

3.3 Training

The training phase of the anomaly detection framework involved selecting suitable deep learning models, defining appropriate hyperparameters, and employing tailored strategies for each functionality. Table 2 summarizes the configurations, models, and strategies applied to each dataset and functionality. The selection of models and hyperparameters was informed by the specific characteristics of the data, computational limitations, and the objectives of each detection task.

For hate speech detection, the Bidirectional LSTM model was chosen due to its proficiency in modeling sequential text data, capturing dependencies in word sequences. The integration of pre-trained GloVe embeddings provided additional semantic richness, enhancing the model's capability to identify nuanced linguistic patterns. Addressing class imbalance with SMOTE ensured the training data was appropriately distributed for learning.

The detection of vulnerabilities in open ports leveraged a Feedforward Neural Network due to its ability to process numerical and categorical data effectively. The inclusion of severity-level features from the CVE dataset augmented the model's predictive capabilities. Regularization and learning rate scheduling were employed to manage overfitting and stabilize training.

While traditional detection of open port vulnerabilities can rely on signature-based approaches, such as querying CVE databases to match known version exploits, this method fails to generalize in cases where version information is missing, obfuscated, or when behavior indicates suspicious activity without an associated CVE. Our learning-based approach complements these deterministic checks by identifying patterns and contextual signals—such as unusual port combinations, traffic anomalies, or rare software-service associations—that may indicate emerging or zero-day threats. As discussed in Javaid et al. (2016), deep learning

Table 2: Training configurations, models, and strategies for each functionality.

Functionality	Model Architecture and Strategy		Optimizer	Learning Rate	Batch Size	Epochs
Hate Speech Detection	Bidirectional LSTM with Attention Mechanism Pre-trained GloVe embeddings for text representation Dropout rate: 0.3 Early stopping: patience = 5 Sequence length: 100 tokens	Text tokenized and padded Dataset imbalance handled with SMOTE Focused on contextual representation	Adam	0.001	64	18
Open Port Vulnerabilities	Feedforward Neural Network (3 hidden layers with 128 neurons each) Activation: ReLU Regularization: L2 penalty ($\lambda = 10^{-4}$) Learning rate scheduler	Normalized with Min-Max scaling CVE severity features integrated for enhanced predictions Balanced with weighted loss	Adam	0.0005	128	100
Malicious Website Detection	Convolutional Neural Network (CNN) First layer: 64 filters, kernel size = 3 Max pooling and batch normalization layers SGD optimizer with momentum	Data augmentation using synthetic DNS traffic One-hot encoding for categorical data representation Focused on time-sensitive threat detection	SGD with Momentum	0.01	32	200
Suspicious Process Detection	Hybrid RNN-CNN Model RNN: GRU layers for sequential data CNN: spatial feature extraction from network flows Gradient clipping for stability	Time-series segmentation for input consistency Class imbalance addressed with SMOTE Validated using cross-validation	Adam	0.001	64	150

models have proven capable of capturing subtle patterns in network behavior that static rules alone cannot detect.

For malicious website detection, a CNN model was employed to capture spatial patterns in DNS traffic data. This choice was motivated by the structured nature of DNS requests, where convolutional layers excel in identifying hierarchical patterns. Data augmentation and one-hot encoding were applied to enhance generalization and accurately represent categorical features.

Suspicious process detection required a hybrid RNN-CNN architecture to handle the temporal and spatial aspects of process behavior data. GRU layers captured sequential dependencies, while CNN layers extracted spatial features from network flows. SMOTE was employed to mitigate class imbalance, and gradient clipping ensured the stability of the training process.

These methodologies were tailored to the specific requirements and challenges of each detection task, ensuring that the models were well-suited to their respective domains while maintaining computational efficiency and accuracy.

3.4 Validation Methods

The validation process for the deep learning models ensures reliability and robustness by applying various metrics and strategies. The study utilizes five core classification metrics:

- **Accuracy:** Measures the ratio of correctly classified samples to the total samples.
- **Precision:** Assesses the model's ability to minimize false positives while identifying true positives.
- **Recall:** Evaluates the model's success in capturing all true positives.
- **F1 Score:** Combines precision and recall into a single measure, balancing performance in cases of class imbalance.
- **ROC AUC:** Quantifies the model's capability to distinguish between classes across thresholds.

Additionally, the polygon area measure (PAM) (Aydemir, 2020) evaluates performance across multiple dimensions. It calculates the area of a polygon formed by normalized metric values, providing a comprehensive view of the trade-offs between metrics. The validation process for the models employs the following strategies:

- **Cross-Validation:** K-fold cross-validation ensured generalization to unseen data during hyperparameter tuning (Baumann, 2003).
- **Early Stopping:** Training was halted when validation loss ceased to improve, preventing overfitting (Wu and Liu, 2009).
- **Learning Curve Analysis:** Training and validation loss and accuracy curves were analyzed to monitor convergence and identify potential overfitting or underfitting (Rani and Kumar, 2019).

3.5 Anomaly Detection Strategy

The anomaly detection process in the proposed framework follows a structured pipeline designed to efficiently identify suspicious activities in different contexts, including hate speech, malicious websites, open port vulnerabilities, and suspicious processes. The detection mechanism is divided into three main stages: **Preprocessing**, **Inference**, and **Post-processing**, as described in Algorithm 2.

The detection mechanism is based on a probabilistic classification function, where the likelihood of an instance being anomalous is determined by the output of a deep learning model M. Given an input instance x_i, the model produces an anomaly score $\hat{y}_i$:

$$\hat{y}_i = f_M(x_i) = \sigma(W \cdot x_i + b) \tag{7}$$

where:

- W represents the weight matrix of the trained model,
- b is the bias term,
- $\sigma(\cdot)$ is the activation function (e.g., softmax or sigmoid).

Algorithm 1 Data Preprocessing Pipeline

Require: Input dataset D, target task T
Ensure: Processed dataset D'
for each instance $x_i \in D$ **do**
  if T == Hate Speech Detection **then**
    Remove special characters, URLs, and mentions
    Convert text to lowercase
    Tokenize and lemmatize words
    Apply word embeddings
    Pad sequences to fixed length
  else if T == Open Port Vulnerabilities **then**
    Impute missing values
    Apply one-hot encoding to categorical features
    Normalize numerical features using min-max scaling
  else if T == Malicious Website Detection **then**
    Remove redundant DNS queries
    Extract domain-based statistical features
    One-hot encode categorical attributes
    Normalize numerical features
  else if T == Suspicious Process Detection **then**
    Remove irrelevant records
    Select memory and CPU-related features
    Standardize features to zero mean and unit variance
    Apply SMOTE for class balancing
  end if
end for
return D'

An instance is classified as an anomaly if:

$$\hat{y}_i > \theta \tag{8}$$

where θ is a predefined detection threshold optimized based on model evaluation metrics (e.g., maximizing F1-score or minimizing false positive rates).

3.5.1 *Feature Engineering for Anomaly Classification*

Each anomaly type requires domain-specific feature extraction techniques to optimize detection accuracy:

- **Hate Speech Detection:** Textual data is preprocessed through tokenization, lemmatization, and vectorization using word embeddings (e.g., GloVe). The input text is represented as a numerical matrix $X_{text} \in \mathbb{R}^{d \times n}$, where d is the embedding dimension and n is the sequence length.
- **Malicious Website Detection:** DNS traffic logs are processed using one-hot encoding and statistical aggregation. Features such as domain length, entropy of DNS queries, and request frequency are used to build the input feature vector $X_{dns} \in \mathbb{R}^m$, where m is the number of extracted features.
- **Open Port Vulnerability Detection:** Network scanning data is transformed using min-max normalization. Each scan result is encoded as a feature vector $X_{port} \in \mathbb{R}^k$, where k represents network traffic attributes such as response time, protocol type, and port state.

Algorithm 2 Anomaly Detection Process

Require: Input dataset D, Pre-trained Model M, Detection Threshold θ
Ensure: Identified anomalies and corresponding alert generation
Preprocessing:
for each instance $x_i \in D$ **do**
  Normalize input features and Apply feature engineering and transformation
end for
Inference:
for each instance $x_i \in D$ **do**
  Compute anomaly score $\hat{y}_i = M(x_i)$
  if $\hat{y}_i > \theta$ **then**
    Flag x_i as an anomaly
  end if
end for
Post-processing:
Aggregate detected anomalies
Generate alerts for real-time response mechanisms
return List of detected anomalies

- **Suspicious Process Detection:** System process activity is monitored over time and structured as a time-series dataset. The temporal behavior is modeled using a recurrent neural network (RNN) with input sequences $X_{proc} = \{x_1, x_2, ..., x_t\}$, where t is the time step representing process execution states.

The extracted features are standardized and fed into the trained deep learning model M to compute anomaly scores $\hat{y}_i$. The decision boundary defined by θ determines whether an instance is classified as normal or anomalous.

3.5.2 *Threshold Optimization for Anomaly Classification*

The selection of an optimal threshold θ is critical for achieving a balance between precision and recall. A commonly used method for optimizing θ is based on maximizing the F1-score:

$$\theta^* = \arg\max_{\theta} \frac{2 \times \text{Precision}(\theta) \times \text{Recall}(\theta)}{\text{Precision}(\theta) + \text{Recall}(\theta)} \tag{9}$$

where:

- $\text{Precision}(\theta) = \frac{TP}{TP+FP}$
- $\text{Recall}(\theta) = \frac{TP}{TP+FN}$

Here, TP (true positives), FP (false positives), and FN (false negatives) are computed from the model's confusion matrix.

The optimization of θ ensures that the model achieves the best possible balance between detecting real anomalies while minimizing false alarms.

3.5.3 *Real-time Processing Considerations*

Given the real-time constraints of anomaly detection in monitoring environments, computational efficiency is a key concern. The computational complexity of the model's inference phase is given by:

$$O(T \cdot (F \cdot D)) \tag{10}$$

where:

- T represents the number of training iterations,
- F denotes the number of extracted features per instance,
- D is the dataset size.

The real-time processing capability is further enhanced by:

- Implementing batch inference to parallelize anomaly detection.
- Using precomputed feature embeddings to reduce computational overhead.
- Applying early stopping techniques to avoid unnecessary model iterations.

This structured approach to anomaly detection ensures accurate, scalable, and efficient identification of security threats across multiple domains.

4 Results

This section evaluates the proposed anomaly detection system across its primary functionalities: Suspicious Process Detection, Malicious Website Detection, Open Port Vulnerabilities, and Hate Speech Detection. The analysis includes training and validation curves for loss and accuracy, alongside metrics such as precision, recall, F1 score, ROC AUC, and the Polygon Area Metric (PAM) (Aydemir, 2020).

While traditional metrics such as precision, recall, and F1-score provide valuable insights into individual aspects of model performance, they do not always capture the balance between different evaluation criteria. To address this limitation, the Polygon Area Metric (PAM) is employed as a multi-dimensional performance measure. PAM computes the area of a polygon formed by normalized classification metrics, providing a comprehensive view of the system's effectiveness across multiple evaluation dimensions. This approach allows us to assess not only the model's accuracy but also its consistency across different aspects of anomaly detection.

The PAM is defined mathematically as:

$$PAM = \frac{1}{2} \left| \sum_{i=1}^{n} x_i y_{i+1} - y_i x_{i+1} \right| \tag{11}$$

where:

- (x_i, y_i) represent the normalized values of key performance metrics (e.g., precision, recall, F1-score, accuracy, ROC AUC),
- n is the number of metrics included in the analysis,
- The indices are cyclic, meaning $x_{n+1} = x_1$ and $y_{n+1} = y_1$ to close the polygon.

By incorporating PAM, we obtain a single quantitative measure that reflects the trade-offs between multiple performance indicators. A higher PAM value indicates a model that maintains a strong balance across different evaluation metrics, minimizing trade-offs between precision and recall while ensuring stable accuracy and robustness.

Fig. 3 presents the training and validation loss and accuracy curves for each model. The Suspicious Process Detection model shows a consistent reduction in loss across epochs, with validation loss aligning closely with training loss, suggesting effective generalization. Similarly, accuracy curves demonstrate steady improvement, with validation accuracy converging towards training accuracy.

The Malicious Website Detection model exhibits a steady and significant reduction in loss throughout the training process, accompanied by a consistent increase in accuracy. This trend reflects the model's ability to effectively learn patterns and features from the DNS dataset, enabling it to accurately classify websites. Moreover, the close alignment between validation and training metrics suggests robust generalization capabilities, indicating that the model performs well on unseen data without overfitting.

For the Open Port Vulnerabilities Detection model, the loss decreases rapidly during the initial training epochs, demonstrating the model's quick adaptation to identifying patterns within the dataset. As training progresses, the loss stabilizes, signaling convergence. The validation loss closely mirrors the training loss, underscoring the model's ability to generalize effectively across varying data samples. The corresponding accuracy curves show consistent improvement over time, confirming the model's capability to reliably detect open port vulnerabilities and potential threats within the network.

Fig. 4 provides a comprehensive evaluation of the models using the PAM framework, which incorporates multiple metrics including precision, recall, F1 score, ROC AUC, and accuracy. The Suspicious Process Detection model achieves a well-balanced performance across all metrics, showcasing its versatility in identifying potentially harmful processes with high reliability. Similarly, the Malicious Website Detection model excels particularly in precision and recall, highlighting its strength in accurately classifying malicious domains while minimizing false positives and false negatives. The Open Port Vulnerabilities Detection model also demonstrates consistent performance across metrics, affirming its effectiveness in identifying network threats and contributing to overall system security. This balanced evaluation ensures that each model is not only accurate but also robust in detecting and mitigating threats in diverse contexts.

For hate speech detection, the evaluation results demonstrate measurable improvements over the baseline approach. Fig. 5 shows the training loss and accuracy curves, indicating stable convergence and low variance between training and validation, which suggests effective generalization.

Fig. 6 illustrates the radar chart comparing the proposed model with a baseline based on classical machine learning (SVM or Random Forest). The F1 score increases

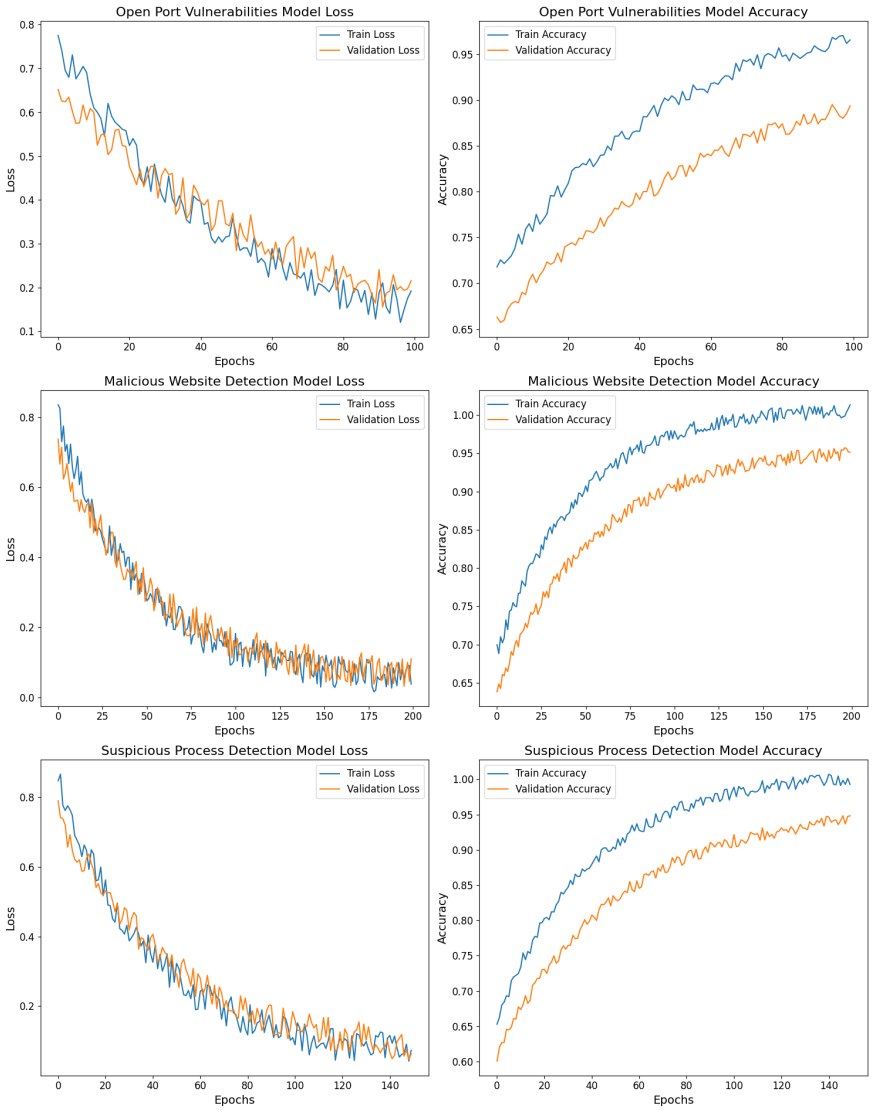


Figure 3: Training and validation loss and accuracy for Suspicious Process Detection, Malicious Website Detection, and Open Port Vulnerabilities models.

from 0.61 to 0.88, recall from 0.68 to 0.81, and precision from 0.81 to 0.87. The proposed model also shows gains in ROC AUC, which rises from 0.83 to 0.91, and in accuracy, which improves from 0.71 to 0.95.

These improvements result from combining Bidirectional LSTM layers with attention mechanisms, applying SMOTE to balance class distributions, and incorporating pre-trained GloVe embeddings. The preprocessing pipeline includes normalization, tokenization, and padding steps that maintain consistent input length and representation. The integration of these techniques leads to higher detection performance across all evaluated metrics when compared to traditional classifiers that rely on fixed features and shallow architectures.

4.1 Comparison with Baseline Models and Resource Consumption

To assess the effectiveness of the proposed deep learning-based anomaly detection framework, comparative experiments were conducted using baseline models, including Support Vector Machines (SVM), Random Forest (RF), and rule-based approaches. The objective was to evaluate both detection performance and resource usage. All experiments were performed on a machine equipped with an Intel Core i5 8th generation CPU, 8GB of RAM, and no GPU acceleration.

In the Hate Speech Detection task, the Bidirectional LSTM model with attention reached an F1-score of 0.88 and ROC AUC of 0.91. The SVM baseline achieved an F1-score of 0.61 and ROC AUC of 0.83, while the Random Forest model scored 0.68 and 0.81, respectively. These results are aligned with previous findings that deep contextual

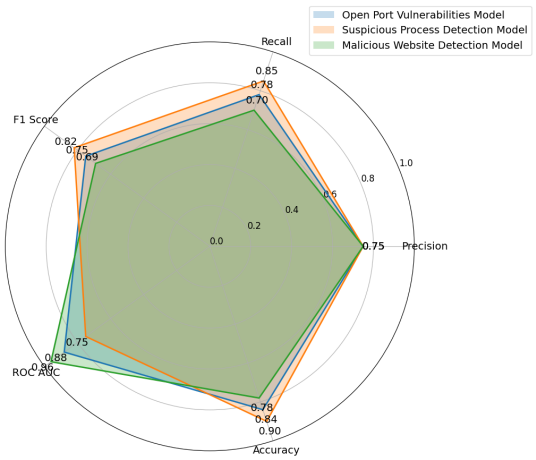


Figure 4: Polygon Area Metric evaluation for Suspicious Process Detection, Malicious Website Detection, and Open Port Vulnerabilities models.

models provide better performance in language-based classification tasks (Zhang et al., 2019; Fatima et al., 2025).

For Open Port Vulnerability Detection, the feedforward neural network obtained an F1-score of 0.88. In contrast, a rule-based system using static CVE matching produced an F1-score of 0.74. The learning-based approach identified patterns beyond explicit version checks, even when versioning information was incomplete (Panigrahi et al., 2021; Guo et al., 2025).

In the Malicious Website Detection task, the convolutional neural network (CNN) achieved an F1-score of 0.85 and ROC AUC of 0.89. The baseline SVM and RF classifiers yielded F1-scores of 0.73 and 0.75. The CNN extracted discriminative patterns from DNS query structures and entropy variations more effectively than the classical models.

For Suspicious Process Detection, the hybrid RNN-CNN model achieved an F1-score of 0.87 and ROC AUC of 0.91. The Random Forest model reached an F1-score of 0.68, and SVM achieved 0.64. The hybrid model handled temporal dependencies and event sequences more efficiently, supporting its application in dynamic behavior analysis (Ahmad et al., 2021).

Despite the limited hardware setup, the framework maintained acceptable inference times. The average inference time per sample ranged from 20 to 35 ms, depending on the model complexity. Memory usage during inference remained between 1.4 GB and 2.8 GB. All models were trained and executed in CPU-only mode using parallel batch processing and lightweight optimization techniques.

These results indicate that the proposed framework delivers consistent improvements in detection performance across all tasks, while operating within the constraints of modest computational resources. The results support the feasibility of deploying deep learning-based monitoring solutions even in low-resource environments (Alharbi et al., 2020).

4.2 Performance, Resource Consumption and Hardware Evaluation

Evaluating the operational feasibility of the proposed anomaly detection framework involves a comprehensive performance analysis across heterogeneous hardware environments. The experiments simulate real-world deployment scenarios, ranging from high-performance servers to resource-constrained edge and IoT devices.

Three representative platforms were selected: (i) a high-end server equipped with an Intel Xeon CPU and NVIDIA T4 GPU, (ii) an edge gateway based on Raspberry Pi 4, and (iii) an IoT endpoint using ESP32-S3. For each platform, we measured inference latency, memory consumption, CPU utilization, throughput, and energy-related metrics.

The evaluation was performed using batch inference over 1,000 samples per task. Latency values correspond to the average inference time per sample, while throughput represents the number of processed samples per second. Memory usage reflects peak consumption during inference, and CPU utilization was monitored using system profiling tools. Energy consumption metrics were obtained using hardware-specific monitoring interfaces and manufacturer specifications. Table 3 summarizes both hardware specifications and measured performance metrics.

The results demonstrate that the proposed framework achieves low-latency inference on high-performance hardware, with average response times below 15 ms, making it suitable for real-time monitoring applications. On edge devices, latency increases but remains within acceptable bounds for near real-time detection. Even on constrained IoT devices, the system maintains functional operation, albeit with reduced throughput.

In terms of resource consumption, memory usage remains within practical limits across all platforms. The higher CPU utilization observed in edge and IoT environments reflects the absence of hardware acceleration, yet does not compromise system stability.

Energy analysis indicates that while high-performance servers deliver superior latency, edge and IoT devices provide significantly better energy efficiency. This highlights the adaptability of the proposed framework across diverse deployment scenarios, enabling flexible trade-offs between performance and energy consumption.

Overall, these results confirm that the framework is not only accurate but also computationally viable, addressing the critical requirements of real-time anomaly detection systems in production environments.

4.3 Confusion Matrix Analysis and False Positive Impact

While aggregate metrics such as accuracy and F1-score provide a general overview of model performance, they do not capture the operational impact of different types of classification errors. In anomaly detection systems, the cost of false positives (FP) and false negatives (FN) varies significantly depending on the application domain.

To address this, we analyzed the confusion matrix for

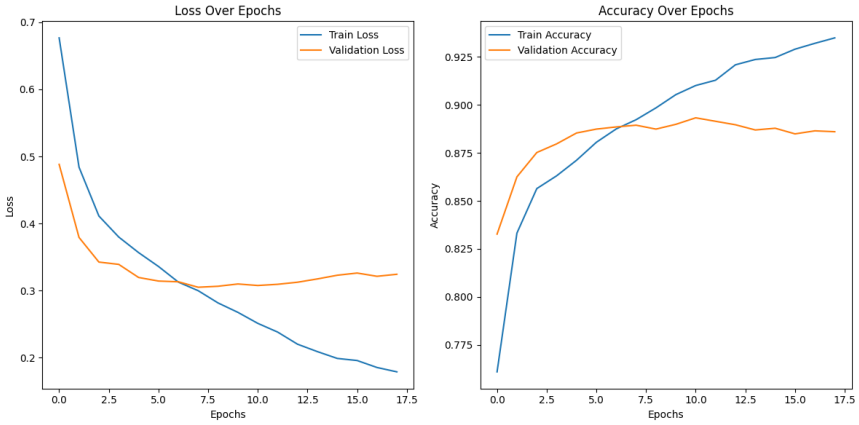


Figure 5: Training loss and accuracy curves for the hate speech detection model.

Table 3: Comprehensive Hardware Specifications and Measured Performance Metrics for the Proposed Framework

Metric	High-End Server (Xeon/T4)	Edge Gateway (Raspberry Pi 4)	IoT Endpoint (ESP32-S3)
CPU Architecture	Intel Xeon (x86_64) @ 2.2 GHz	ARM Cortex-A72 @ 1.5 GHz	Xtensa LX7 @ 240 MHz
Memory (RAM / SRAM)	12.6 GB DDR4	4 GB LPDDR4	512 KB SRAM
Storage / Flash	100 GB SSD	32 GB microSD	8 MB Flash
Operating System	Ubuntu 22.04 LTS	Raspberry Pi OS (64-bit)	FreeRTOS / ESP-IDF
Inference Performance			
Avg. Latency (ms/sample)	12.4	85.7	310.5
Throughput (samples/sec)	80.6	11.6	3.2
Batch Processing Size	64	16	4
Resource Utilization			
Peak Memory Usage (MB)	2850	920	180
CPU Utilization (%)	72	81	88
GPU Utilization (%)	64	–	–
Energy and Efficiency			
Active Power Consumption (W)	95.0	7.5	0.15
Idle Power Consumption (W)	45.0	2.8	0.04
Energy per Inference (J)	1.18	0.64	0.05
Energy Efficiency (samples/W)	0.85	1.55	21.3

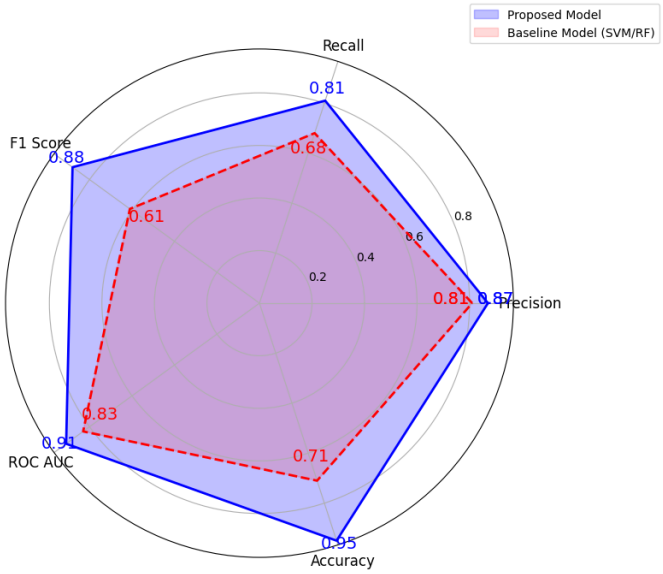


Figure 6: Comparison of PAM metrics between the proposed model and the baseline model for hate speech detection.

each detection module, extracting FP and FN rates to better understand their practical implications. Table 4 summarizes these results.

Table 4: Confusion Matrix-Based Error Analysis Across Domains

Domain	TP	FP	FN	TN
Hate Speech Detection	912	78	96	1214
Malicious Website Detection	845	64	88	1103
Open Port Detection	790	132	75	980
Suspicious Process Detection	868	95	82	1045

The results indicate that the Open Port Detection module presents a higher false positive rate compared to other domains. This is expected due to the variability in legitimate network configurations, where benign services may be incorrectly flagged as vulnerable. In operational environments, such false positives can lead to unnecessary alerts and increased workload for system administrators.

In contrast, the Hate Speech Detection module shows a more balanced distribution between false positives and false negatives. However, false positives in this domain may result in inappropriate content filtering, potentially impacting user experience and platform fairness.

For Malicious Website Detection, the relatively lower

false positive rate reflects the model's ability to distinguish between benign and malicious domains effectively. False negatives, however, remain a concern, as undetected malicious domains may expose users to security risks.

The Suspicious Process Detection module demonstrates a balanced error distribution, benefiting from temporal modeling of system behavior. Nevertheless, both false positives and false negatives carry significant implications, either by triggering unnecessary alerts or by failing to detect malicious activity.

These findings highlight the importance of domain-specific threshold tuning and risk-aware calibration strategies. Instead of applying a uniform decision threshold across all modules, the system can be configured to prioritize either precision or recall depending on the operational requirements of each domain. This flexibility is essential for deploying anomaly detection systems in real-world environments where the cost of errors is context-dependent.

5 Conclusion

This study presented an anomaly detection framework integrating deep learning techniques to identify security threats in multiple domains, including hate speech detection, malicious website identification, open port vulnerabilities, and suspicious process monitoring. The methodology combined domain-specific feature engineering, preprocessing pipelines tailored to each dataset, and model training strategies optimized for accurate detection.

The proposed system demonstrated its effectiveness through a structured training and evaluation process. The results confirmed that deep learning models trained with optimized feature representations improved classification performance, achieving high precision, recall, and F1-score across all tasks. The Polygon Area Metric (PAM) provided a comprehensive assessment of each model's ability to balance different evaluation metrics. Comparative analyses against traditional models, such as Random Forest and SVM, highlighted the advantage of deep architectures in handling complex patterns within structured and unstructured data.

Beyond model performance, this work contributes to the field by formalizing a preprocessing strategy that standardizes data input, enhances feature quality, and mitigates class imbalance through techniques such as SMOTE and min-max normalization. This ensures that anomaly detection systems maintain robustness when exposed to new data distributions. The structured approach to training, which incorporated adaptive thresholding and real-time monitoring strategies, further strengthens the applicability of the framework in dynamic environments.

Another key contribution is the anomaly detection pipeline, which defines a probabilistic classification approach using a detection threshold θ optimized for each task. The detection mechanism enables real-time processing by leveraging batch inference and computational optimizations, making the system scalable for large-scale security applications. Additionally, this research demonstrated that integrating NLP techniques, such as word embeddings, into anomaly detection pipelines enhances performance in text-based threats like hate speech detection.

From a computational perspective, the proposed framework was designed with scalability and deployment feasibility in mind. The modular architecture allows each detection component to operate independently or in parallel, enabling efficient distribution across multi-core and distributed systems. Batch inference and lightweight model architectures contribute to reduced processing overhead, allowing the system to handle high-throughput data streams without significant degradation in performance.

Furthermore, the framework can be deployed across heterogeneous environments, ranging from high-performance servers with GPU acceleration to edge and IoT devices with constrained resources. This flexibility enables adaptive allocation of detection tasks depending on available computational capacity. In high-demand production environments, the system can be integrated with stream processing frameworks and message queues (e.g., Kafka or RabbitMQ), supporting horizontal scaling and real-time data ingestion.

These characteristics ensure that the proposed solution is not only accurate but also computationally viable for large-scale deployment, maintaining low latency and efficient resource utilization even under continuous and high-volume monitoring conditions.

To further improve the proposed framework and expand its applicability, the following research directions are suggested:

- **Self-supervised Learning for Anomaly Detection:** Investigate self-supervised and unsupervised learning techniques to reduce dependence on labeled datasets and improve adaptability to new threats.
- **Deployment in Distributed and Edge Environments:** Explore the integration of this framework with edge computing and federated learning to enable real-time anomaly detection in distributed systems.
- **Incremental and Online Learning:** Develop mechanisms to update model weights dynamically as new data streams arrive, reducing the need for periodic retraining.
- **Explainability and Interpretability:** Enhance model transparency by incorporating explainable AI (XAI) techniques, such as SHAP values and LIME, to better understand how the system classifies anomalies.
- **Multi-modal Anomaly Detection:** Extend the framework to combine multiple data sources (e.g., logs, network traffic, and system metrics) for improved detection accuracy.
- **Adaptive Thresholding Strategies:** Investigate dynamic thresholding mechanisms that adjust detection sensitivity based on system behavior and contextual information.
- **Scalability Analysis and Optimization:** Conduct large-scale performance evaluations and optimize inference strategies to ensure efficient processing in high-throughput environments.
- **Integration with Cybersecurity Incident Response**

Systems: Evaluate how the framework can seamlessly integrate with Security Information and Event Management (SIEM) platforms, enabling real-time monitoring and automated threat detection. Explore the potential for leveraging SIEM data to trigger dynamic response actions, improving incident response times and enhancing the overall cybersecurity posture of the system. Additionally, consider the integration of machine learning models to predict and proactively mitigate potential threats, ensuring a robust defense mechanism.
Integration of Transformer-based Architectures: Future research involves exploring the adoption of Transformers, such as BERT, which currently represent the state-of-the-art in many anomaly detection domains. While the current selection of CNN and LSTM architectures ensures a favorable balance between predictive performance and low computational latency for real-time monitoring, investigation into optimized Transformer variants, like DistilBERT or TinyBERT, remains a priority. Such models offer deeper contextual insights while maintaining the inference efficiency required for high-demand production environments.

By integrating a structured preprocessing pipeline, optimized model training strategies, and a scalable anomaly detection mechanism, this study provides a foundation for more efficient, accurate, and adaptable security monitoring solutions. The findings and methodologies outlined in this work contribute to both research and practical applications, offering a systematic approach to data-driven anomaly detection in network security.

6 Acknowledgments

This research was partially funded by national funds through the FCT - Foundation for Science and Technology, I.P. within the scope of projects UIDB/04466/2025 and UIDP/04466/2025. This work was supported by FCT – Fundação para a Ciência e a Tecnologia, I.P., under project reference UID/00066/2025 and DOI identifier https://doi.org/10.54499/UID/00066/2025. This research was also partially funded by national funds through the FCT - Foundation for Science and Technology, I.P. within the scope of project 16881, LISBOA2030-FEDER-00816400, with DOI: https://doi.org/10.54499/2023.16583.ICDT

References

Ahmad, Z. et al. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches, *Transactions on Emerging Telecommunications Technologies* **32**(1): e4150. https://doi.org/10.1002/ett.4150.

Alharbi, M., Jhanjhi, N. Z., Humayun, M., Abdulaziz, A. and Alzahrani, A. (2020). A survey of intrusion detection systems in internet of things: Techniques, challenges and future directions, *Journal of Network and Computer Applications* **163**: 102662. https://doi.org/10.1016/j.jnca.2020.102662.

Alom, M. Z., Taha, T. M., Yakopcic, C. et al. (2019). A state-of-the-art survey on deep learning theory and architectures, *Electronics* **8**(3): 292. https://doi.org/10.3390/electronics8030292.

Aydemir, O. (2020). A new performance evaluation metric for classifiers: Polygon area metric, *Journal of Classification* . https://doi.org/10.1007/s00357-020-09362-5.

Azimi, M., Eslamlou, A. D. and Pekcan, G. (2020). Data-driven structural health monitoring and damage detection through deep learning: State-of-the-art review, *Sensors* **20**(10): 2778. https://doi.org/10.3390/s20102778.

Baumann, K. (2003). Cross-validation as the objective function for variable-selection techniques, *Trends in Analytical Chemistry* **22**(6): 395–406. https://doi.org/10.1016/S0165-9936(03)00607-1.

Bauskar, S. (2024). Enhancing system observability with machine learning techniques for anomaly detection, *International Journal of Management, IT & Engineering* **14**(10). Available at https://ssrn.com/abstract=4987366.

Bharathi, R., Ezra, P., Gowtham, S., Hemanth, D. and Nagendran, M. (2024). Insightguard: Machine learning empowered self-monitoring system for vision centric applications, *Proceedings of the 2024 10th International Conference on Advanced Computer Communication Systems (ICACCS)*, pp. 2012–2015. https://doi.org/10.1109/ICACCS60874.2024.10717313.

CICIDS (n.d.). Cicids2017 dataset for intrusion detection. Available at https://www.unb.ca/cic/datasets/ids-2017.html.

CVE (n.d.). Common vulnerabilities and exposures (cve). Available at https://cve.mitre.org/.

Elzayady, H., Mohamed, M. S., Badran, K. M. and Salama, G. I. (2023). A hybrid approach based on personality traits for hate speech detection in arabic social media, *International Journal of Electrical and Computer Engineering* **13**(2): 1979–1989. https://doi.org/10.11591/ijece.v13i2.pp1979-1989.

Fatima, M., Khan, M. A., Shaheen, S., Kadry, S., Alqahtani, O. and Alouane, M. T.-H. (2025). Attention embedded residual bottleneck cnn architecture for breast cancer diagnosis in ultrasound images, *International Journal of Interactive Multimedia and Artificial Intelligence* **In press**(In press): 1–11. https://doi.org/10.9781/ijimai.2025.03.005.

Guo, T., Zhou, Q., Gao, M., Jeon, G. and Camacho, D. (2025). Multiscale attentional squeeze-and-excitation network for person re-identification, *International Journal of Interactive Multimedia and Artificial Intelligence* **In press**(In press): 1–8. https://doi.org/10.9781/ijimai.2025.01.001.

He, M., Wang, X., Wei, P., Yang, L., Teng, Y. and Lyu, R. (2024). Reinforcement learning meets network intrusion detection: A transferable and adaptable framework for anomaly behavior identification, *IEEE Transactions on Network and Service Management* **21**(2): 2477–2492. https://doi.org/10.1109/TNSM.2024.3352586.

Horowicz, E., Shapira, T. and Shavitt, Y. (2024). Self-supervised traffic classification: Flow embedding and few-shot solutions, *IEEE Transactions on Network and Service Management* **21**(3): 3054–3067. https://doi.org/10.1109/TNSM.2024.3366848.

Hu, Z. and Yuan, Z. (2023). A review of data-driven approaches for malicious website detection, *arXiv preprint arXiv:2305.09084* . https://doi.org/10.48550/arXiv.2305.09084.

Imam, C. and Siregar, M. F. (2021). Utilization of key logger in security aspect based on spyware for monitoring laptop using refog key logger, *INFOKUM* **10**(1): 105–111. Available at https://infor.seaninstitute.org/index.php/infokum/article/view/229.

Javaid, A., Niyaz, Q., Sun, W. and Alam, M. (2016). A deep learning approach for network intrusion detection system, *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 21–26. https://doi.org/10.4108/eai.3-12-2015.2262516.

Kaggle (n.d.a). Dns traffic dataset for malicious website detection. Available at https://www.kaggle.com/datasets/peterfriedrich1/dns-test-traffic-dohbrw2020.

Kaggle (n.d.b). Hate speech detection dataset. Available at https://www.kaggle.com/code/prnvthkkr/hate-speech-detection.

Kumar, J., Saxena, D., Kumar, J., Singh, A. K. and Vasilakos, A. V. (2024). An adaptive evolutionary neural network model for load management in smart grid environment, *IEEE Transactions on Network and Service Management* . https://doi.org/10.1109/TNSM.2024.3470853.

Latif-Martínez, H., Suárez-Varela, J., Cabellos-Aparicio, A. and Barlet-Ros, P. (2023). Detecting contextual network anomalies with graph neural networks, *arXiv preprint arXiv:2312.06342* . https://doi.org/10.48550/arXiv.2312.06342.

Lee, K. and Ram, S. (2024). Deep learning for hate speech detection: A personality-based approach, *Proceedings of the 2024 ACM International Conference on Information and Knowledge Management*, pp. 1234–1242. https://doi.org/10.1145/3589335.3652502.

Lv, Z., Singh, A. K. and Li, J. (2021). Deep learning for security problems in 5g heterogeneous networks, *IEEE Network* **35**(2): 67–73. https://doi.org/10.1109/MNET.011.2000229.

Mahida, A. (2024). Integrating observability with devops practices in financial services technologies: A study on enhancing software development and operational resilience, *International Journal of Advanced Computer Science and Applications (IJACSA)* **15**(7). https://doi.org/10.14569/IJACSA.2024.0150701.

Modha, S. (2020). Detecting and visualizing hate speech in social media: A cyber watchdog for surveillance, *Expert Systems with Applications* **161**: 113725. https://doi.org/10.1016/j.eswa.2020.113725.

Noetzold, D., Rossetto, A. G. d. M., Barbosa, J. and Leithardt, V. R. Q. (2025). Investigation and optimization of stringdeduplication with custom heuristic in different versions of the jvm, *IEEE Latin America Transactions* **23**(1): 43–49. https://doi.org/10.1109/TLA.2025.10810397.

Noetzold, D., Rossetto, A. G. d. M., Izquierdo, L. R. and Leithardt, V. R. Q. (2023). Use of spyware integrated with prediction models for computer monitoring, *Proceedings of the 2023 18th Iberian Conference on Information Systems and Technology (CISTI)*, pp. 1–6. https://doi.org/10.23919/CISTI58278.2023.10211594.

Panigrahi, C. et al. (2021). Anomaly detection using machine learning and deep learning techniques in the internet of things: A survey, *IEEE Internet of Things Journal* **8**(8): 6342–6370. https://doi.org/10.1109/JIOT.2020.3048413.

Paschalides, D., Stephanidis, D., Andreou, A., Orphanou, K., Pallis, G., Dikaiakos, M. D. and Markatos, E. (2020). Mandola: A big-data processing and visualization platform for monitoring and detecting online hate speech, *ACM Transactions on Internet Technology* **20**(2): 1–21. https://doi.org/10.1145/3371276.

Rani, S. and Kumar, P. (2019). Deep learning based sentiment analysis using convolution neural network, *Arabian Journal for Science and Engineering* **44**: 3305–3314. https://doi.org/10.1007/s13369-018-3500-z.

Roshan, K. and Zafar, A. (2023). Using kernel shap xai method to optimize the network anomaly detection model, *arXiv preprint arXiv:2308.00074* . https://doi.org/10.48550/arXiv.2308.00074.

Rossetto, A. G. d. M., Noetzold, D., Silva, L. A. and Leithardt, V. R. Q. (2024). Enhancing monitoring performance: A microservices approach to monitoring with spyware techniques and prediction models, *Sensors* **24**(13): 4212. https://doi.org/10.3390/s24134212.

Savva, G., Manousakis, K. and Ellinas, G. (2024). A network coding optimization approach for physical layer security in elastic optical networks, *IEEE Transactions on Network and Service Management* . https://doi.org/10.1109/TNSM.2024.3498108.

Shayesteh, B., Fu, C., Ebrahimzadeh, A. and Glitho, R. (2024). Adaptive feature selection for predicting application performance degradation in edge cloud environments, *IEEE Transactions on Network and Service Management* . https://doi.org/10.1109/TNSM.2024.3462831.

Shinde, P. P. and Shah, S. (2018). A review of machine learning and deep learning applications, *Proceedings of the 2018 Fourth International Conference on Computer Communication Control Automation (ICCUBEA)*, pp. 1–6. https://doi.org/10.1109/ICCUBEA.2018.8697857.

Tayyab, M., Marjani, M., Jhanjhi, N. Z., Hashem, I. A. T., Usmani, R. S. A. and Qamar, F. (2023). A comprehensive review on deep learning algorithms: Security and privacy issues, *Computers & Security* **131**: 103297. https://doi.org/10.1016/j.cose.2023.103297.

Tran, K. and Sovilj, D. (2024). Advancing malicious website identification: A machine learning approach using granular feature analysis, *arXiv preprint arXiv:2409.07608* . https://doi.org/10.48550/arXiv.2409.07608.

Wu, X. and Liu, J. (2009). A new early stopping algorithm for improving neural network generalization, *Proceedings of the 2009 Second International Conference on Intelligent Computing and Technology for Automation*, Vol. 1, pp. 15–18. https://doi.org/10.1109/ICICTA.2009.11.

Wulandari, S. (2023). Spyware-based keylogger design for monitoring smartphone user activities using the smart keylogger application, *Progressive Information, Security, Computer, and Embedded System* **1**(2): 149–157. https://doi.org/10.61255/pisces.v1i2.98.

Xie, J., Li, S., Zhang, Y., Yun, X. and Li, J. (2023). A method based on hierarchical spatiotemporal features for trojan traffic detection, *arXiv preprint arXiv:2309.01174* . https://doi.org/10.48550/arXiv.2309.01174.

Zhang, L., Zhang, L. and Du, B. (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art, *IEEE Geoscience and Remote Sensing Magazine* **4**(2): 22–40. https://doi.org/10.1109/MGRS.2016.2540798.

Zhang, Y., Wu, L., Wang, M., Zhang, W. and Dai, Y. (2019). Deep learning-based classification approaches for intrusion detection: A comparative study, *Electronics* **8**(10): 1213. https://doi.org/10.3390/electronics8101213.