Intelligent Network Management and Automated Fault Detection Using Deep Learning Approaches

Mrs. P. Sofia
Assistant professor
School of computer science
Sri Ramakrishna College of Arts and Science for women, Coimbatore

Ms. T Sreevanitha
Assistant professor
School of computer science
Sri Ramakrishna College of Arts and Science for women, Coimbatore

ABSTRACT

Modern communication networks support cloud services, enterprise applications, Internet of Things (IoT) deployments and latency-sensitive digital services. Their scale and dynamic behaviour make manual monitoring and rule-based fault management increasingly difficult. Network faults such as link degradation, congestion, routing instability, device failures and abnormal traffic can propagate across interconnected components and cause service disruption. This article presents a deep learning-oriented framework for intelligent network management and automated fault detection. The proposed framework combines continuous telemetry collection, data preprocessing, temporal and spatial feature learning, anomaly classification, fault localisation and controlled remediation. A hybrid model design is discussed in which a temporal neural network, such as a Long Short-Term Memory (LSTM) network, learns changes in time-series performance indicators, while a one-dimensional Convolutional Neural Network (1D-CNN) extracts local patterns from multivariate telemetry windows. An optional autoencoder supports anomaly detection when labelled fault examples are limited. The framework is designed to integrate with network management systems through telemetry interfaces and an event-driven decision layer. The article also outlines a reproducible evaluation methodology using public network intrusion and traffic datasets alongside a separately collected operational telemetry dataset. Evaluation measures include precision, recall, F1-score, false alarm rate, detection latency, localisation accuracy and resource overhead. No experimental performance values are claimed because the framework is proposed for future implementation and empirical validation. The discussion identifies key deployment challenges, including data imbalance, concept drift, explainability, privacy, model robustness and safe automation. The proposed approach provides a structured foundation for moving from reactive monitoring toward predictive, data-driven and human-supervised network operations.

KEYWORDS

Intelligent network management; deep learning; automated fault detection; network anomaly detection; fault localisation; LSTM; convolutional neural network; AIOps; network telemetry; predictive maintenance.

1. INTRODUCTION

Computer networks have evolved from relatively static infrastructures into highly distributed environments containing physical routers and switches, virtual network functions, wireless access points, cloud platforms, edge devices and software-defined control systems. These environments generate large volumes of operational data, including interface counters, packet and flow statistics, latency measurements, routing events, system logs and configuration changes. Although this data provides visibility into network behaviour, extracting timely and actionable information from it remains challenging. Traditional network management commonly relies on threshold-based alarms, static rules, protocol-specific diagnostics and administrator expertise. Such techniques remain useful for known conditions but can struggle with interacting symptoms, changing traffic patterns and previously unseen failure modes.

A network fault is not limited to complete device failure. It may appear as gradual packet loss, increasing round-trip time, queue build-up, intermittent connectivity, routing convergence delay, wireless interference, resource exhaustion

or abnormal traffic. A fault in one component may create symptoms in other components, making root-cause identification difficult. For example, application response time may increase because of congestion on an upstream link, a failing interface, a routing change or an overloaded service. A monitoring system that reports only threshold violations may generate multiple alerts without identifying the underlying event.

Deep learning offers methods for learning complex representations from large and heterogeneous datasets. Recurrent architectures can model temporal dependencies, convolutional architectures can detect local patterns, autoencoders can learn normal operating behaviour, and attention-based models can relate events across longer time windows. Recent literature has examined deep learning for network intrusion and anomaly detection, including software-defined networking environments [1]–[3]. However, security anomaly detection and operational fault management are related but distinct tasks: the former focuses on malicious or policy-violating behaviour, while the latter must also identify availability, performance and infrastructure failures.

1.1 Research Problem

The central problem addressed in this article is the lack of a unified, explainable and operationally safe pipeline that converts network telemetry into timely fault alerts, likely fault locations and recommended recovery actions. A model may detect an anomaly accurately but still be operationally weak if it cannot distinguish transient variation from a persistent fault, explain the evidence, or integrate with existing management workflows.

1.2 Objectives and Contributions

- Develop a layered architecture for telemetry-driven network monitoring, fault detection, localisation and response.
- Describe how CNN, LSTM and autoencoder components can be combined for complementary feature learning.
- Define a methodology for dataset preparation, model training, validation and realistic testing.
- Establish evaluation measures that include detection quality, latency, operational overhead and safety.
- Identify implementation risks and future research directions for deployable intelligent network operations.

2. RELATED WORK AND RESEARCH BACKGROUND

Research on intelligent network operations spans network anomaly detection, intrusion detection, traffic forecasting, fault diagnosis and autonomous network management. These areas share a dependence on telemetry and pattern recognition, but they differ in target labels and operational consequences. Intrusion detection generally classifies traffic or behaviour as benign or malicious. Fault detection identifies deviations from expected service or infrastructure behaviour, while fault localisation estimates the component or dependency responsible for the observed symptoms. A practical network management platform may need all three capabilities.

2.1 Deep Learning for Network Anomaly Detection

Deep neural networks learn feature representations through multiple nonlinear transformations. In network monitoring, inputs may include packet-flow statistics, interface utilisation, packet loss, latency, jitter, CPU and memory usage, queue length, routing changes and event counts. Supervised classifiers can distinguish known fault categories when sufficient labelled examples exist. Unsupervised and semi-supervised approaches are useful when failures are rare or labels are incomplete. Autoencoders, for example, learn to reconstruct typical observations; unusually high reconstruction error can indicate behaviour that differs from the learned baseline. Such scores require calibration because unusual behaviour is not necessarily a fault.

A recent survey categorises deep learning-based network intrusion detection approaches into reconstruction or generative models, transformer-based sequence models, convolutional and feed-forward networks, and hybrid deployment-oriented approaches [1]. This taxonomy is useful for selecting candidate model families, although intrusion detection results should not be treated as direct evidence of operational fault-management performance. A survey focused on software-defined networks also highlights the importance of datasets, preprocessing, model selection and evaluation design [2].

2.2 Temporal Modelling and Unknown Faults

Network measurements are time-dependent. A short increase in utilisation may be harmless during a traffic burst, whereas a sustained rise combined with packet loss and latency may indicate congestion. LSTM networks and other recurrent models are designed to retain information across sequences and can therefore represent temporal dependencies. Transformer-based architectures may capture longer-range relationships, but their computational requirements and data needs must be assessed before deployment. Research on unknown, atypical and polymorphic network intrusions similarly identifies generalisation to previously unseen patterns as a continuing challenge [3].

2.3 Standards and Autonomous Network Management

The International Telecommunication Union (ITU-T) Recommendation Y.3177 describes an architectural framework for AI-based network automation for resource and fault management in future networks, including IMT-2020 [4]. ITU-T Y.3061 provides an architecture framework for autonomous networks [5]. These recommendations reinforce the importance of separating data collection, intelligence, decision-making and network control functions. They also provide useful architectural context for a system that combines learning-based fault detection with policy-governed automation.

2.4 Research Gap

A recurring gap is the transition from model-level detection to reliable operational action. Many studies emphasise classification metrics, while production network management additionally requires fault localisation, alert correlation, drift monitoring, service-level context, explainability and rollback mechanisms. This article addresses that integration gap through a proposed end-to-end framework and a deployment-oriented evaluation plan.

3. TECHNICAL BACKGROUND AND DEEP LEARNING COMPONENTS

3.1 Network Telemetry as Multivariate Time Series

Let a network observation at time t be represented by a feature vector x(t) containing measurements from one or more devices or links. A sequence of length T is represented as $X = [x(t-T+1), \ldots, x(t)]$. Features may include inbound and outbound throughput, packet error rate, packet drops, interface utilisation, round-trip time, jitter, queue depth, CPU load, memory use and routing-event indicators. The data is multivariate because several metrics are measured together, and it is non-stationary because traffic demand and operating conditions change over time.

Before model training, telemetry should be aligned to a common time interval, checked for missing or duplicated records, and associated with device, interface, service and topology identifiers. Continuous variables may be scaled using statistics derived only from the training partition. Categorical fields such as device type or event class can be encoded separately. Time-based splitting is important: randomly mixing adjacent time windows across training and test sets can cause leakage and produce an unrealistically optimistic estimate.

3.2 Long Short-Term Memory Networks

An LSTM is a recurrent neural network architecture designed to learn dependencies over sequences. Its input, forget and output gates regulate the information retained in the cell state. For network management, an LSTM can learn the temporal evolution of latency, throughput, packet loss and resource utilisation. A sequence classifier can map a telemetry window to a fault category, while a sequence predictor can forecast expected measurements. The difference between predicted and observed values may provide an additional anomaly signal. LSTM performance depends on window length, feature scaling, model capacity and the representativeness of the training data.

3.3 One-Dimensional Convolutional Neural Networks

A 1D-CNN applies convolutional filters across ordered telemetry samples. Filters can detect short-lived patterns such as bursts, oscillations, sudden drops or repeated error sequences. Pooling or strided convolution reduces sequence length and computational cost. CNN layers can be used alone for classification or combined with recurrent layers so that local patterns and longer temporal dependencies are both represented.

3.4 Autoencoders for Unsupervised Detection

An autoencoder contains an encoder that maps an input window into a lower-dimensional representation and a decoder that reconstructs the input. Training on predominantly healthy network observations encourages the model to represent common operating patterns. A reconstruction score, such as mean squared error, can be compared with a threshold to flag unusual observations. Thresholds should be selected using a validation set and monitored after deployment. A high reconstruction score is a candidate anomaly, not proof of a specific fault.

3.5 Hybrid Model Rationale

The proposed design uses a hybrid approach because no single model family is optimal for all conditions. CNN layers can extract local patterns, LSTM layers can model temporal context, and an autoencoder can provide an anomaly score when labelled examples are sparse. The final decision layer combines these outputs with rule-based constraints and topology information. This combination is intended to balance detection sensitivity, interpretability and operational feasibility.

4. PROPOSED INTELLIGENT NETWORK MANAGEMENT FRAMEWORK

The proposed framework consists of six logical layers: (1) telemetry acquisition, (2) data engineering and context enrichment, (3) deep learning inference, (4) event correlation and fault localisation, (5) decision and orchestration, and (6) feedback and governance. The layers are loosely coupled so that the learning models can be updated without redesigning network devices or the management interface.

4.1 Telemetry Acquisition Layer

The acquisition layer collects operational information from routers, switches, firewalls, wireless controllers, virtual network functions, servers and monitoring agents. Data sources may include SNMP counters, streaming telemetry, NetFlow or IPFIX records, syslog messages, controller events and active probes. Collection frequency should reflect the metric and operational requirements. High-frequency sampling can improve detection of short-lived incidents but increases storage, processing and transport costs. The system should preserve timestamps, source identifiers and collection quality metadata.

4.2 Data Engineering and Context Enrichment

The data engineering layer validates records, handles missing values, aligns sampling intervals and creates fixed-length observation windows. It enriches telemetry with network topology, device roles, service dependencies, maintenance windows and configuration-change events. Topology context is particularly important for localisation because correlated symptoms across connected devices may originate from a shared upstream component. Data quality indicators should accompany each window so that the model can distinguish a true measurement change from a collection failure.

4.3 Deep Learning Inference Layer

A candidate hybrid model receives a tensor of multivariate time-series features. A 1D-CNN branch extracts local patterns, and an LSTM branch learns temporal dependencies. Their learned representations are concatenated and passed to dense layers for fault classification. In parallel, an autoencoder estimates reconstruction error. The inference output may include a predicted fault class, confidence score, anomaly score and temporal evidence. If the model confidence is low or the input is outside the training distribution, the system should return an uncertain or review-required state rather than force a potentially misleading classification.

4.4 Event Correlation and Fault Localisation

Raw alerts are grouped into incidents using time proximity, shared topology, service dependency and symptom similarity. A graph representation can model devices and links as nodes and edges. The localisation component ranks candidate components by combining model evidence, graph distance, dependency relationships and recent configuration events. The ranking should be presented as a hypothesis with supporting evidence, not as an unquestionable root-cause statement. Administrators can confirm or reject the proposed cause, creating feedback for later model improvement.

4.5 Decision and Orchestration Layer

The decision layer maps incident severity and confidence to an action policy. Low-risk actions may include opening a ticket, notifying an operator, collecting additional diagnostics or recommending a configuration check. Higher-impact actions such as rerouting traffic, restarting a service or changing access policies should require explicit approval unless a separately validated policy authorises them. All actions must be logged, bounded by permissions and supported by rollback procedures.

4.6 Feedback and Governance

Confirmed incidents, false alarms, maintenance records and operator feedback are stored with appropriate access controls. Model versions, training datasets, thresholds and action policies are tracked. Drift monitoring checks whether input distributions or error rates change over time. Retraining should use reviewed data and a controlled validation process before a new model is promoted.

5. RESEARCH METHODOLOGY

This article proposes a design-and-evaluation methodology for implementing the framework. The study should be conducted in a controlled laboratory or testbed before any production deployment. The methodology is divided into problem definition, data collection, preprocessing, model development, training, validation and operational testing.

5.1 Research Questions

- RQ1: Can a hybrid CNN–LSTM model identify operational network anomalies more reliably than selected baseline methods?
- RQ2: Does combining supervised classification with an autoencoder improve detection of rare or previously unseen fault patterns?
- RQ3: Can topology-aware event correlation improve fault-localisation accuracy compared with alert-only ranking?
- RQ4: What trade-offs arise among detection quality, inference latency, computational overhead and false alarm rate?

5.2 Dataset Strategy

The study should use two complementary data sources. First, public datasets such as UNSW-NB15 or CICIDS2017 can support experiments on network traffic anomalies and security-related events. These datasets are useful for evaluating traffic classification pipelines, but they do not fully represent equipment failures, service degradation or network operations. Second, an operational telemetry dataset should be collected from a controlled network testbed. It should include healthy periods and deliberately injected conditions such as link interruption, bandwidth limitation, packet loss, delay, queue saturation, interface errors and CPU or memory pressure. Each injected event should have a timestamp, affected component, severity and ground-truth label.

The dataset documentation should record capture environment, sampling interval, feature definitions, label-generation process, missing-data rate and known limitations. Sensitive payload data should not be collected unless necessary. Flow-level metadata and performance counters are generally preferable for privacy and storage efficiency.

5.3 Preprocessing

The preprocessing pipeline should remove duplicates, correct timestamp inconsistencies, align measurements, encode categorical attributes and impute missing values using a method appropriate to the metric. Extreme values should not automatically be deleted because they may represent the very faults the model is expected to detect. Scaling parameters must be fitted on training data only. Sequence windows should be generated chronologically, with a documented overlap policy. Class imbalance can be addressed through class-weighted loss, careful resampling or focal loss; synthetic samples should be used cautiously because unrealistic network sequences can distort the decision boundary.

5.4 Model Design and Baselines

The primary model is a CNN–LSTM classifier with an auxiliary autoencoder anomaly score. Baselines should include threshold rules, logistic regression or random forest, a standalone CNN, a standalone LSTM and an autoencoder-only

detector. Model complexity should be reported alongside performance. Hyperparameters such as window size, number of filters, hidden units, dropout, learning rate, batch size and decision threshold should be tuned on validation data, not on the final test set.

5.5 Training and Validation

A chronological split should separate training, validation and test periods. Where possible, a second test should hold out devices, network segments or fault scenarios to measure generalisation. Repeated runs with fixed and documented random seeds should be conducted. Early stopping may be applied using validation loss. The final test set should be used once for the primary comparison, with additional scenario analysis reported separately.

6. EVALUATION DESIGN AND EXPECTED ANALYSIS

A credible evaluation must measure more than overall accuracy. Network fault datasets are often imbalanced, and a model can achieve high accuracy by predicting the majority healthy class. The evaluation should therefore report class-wise precision, recall and F1-score, macro-averaged metrics, confusion matrices and false alarm rate. For operational use, detection latency and localisation performance are equally important.

6.1 Detection Metrics

Precision is the proportion of predicted fault events that are correct, while recall is the proportion of actual fault events that are detected. The F1-score is the harmonic mean of precision and recall. False positive rate measures the proportion of healthy observations incorrectly flagged as faulty. For rare faults, precision–recall curves can be more informative than accuracy alone. Metrics should be calculated at both observation-window level and incident level, because several consecutive alerts may belong to one operational incident.

6.2 Fault Localisation and Time Metrics

Localisation can be evaluated using top-k accuracy, mean reciprocal rank or the proportion of incidents for which the true faulty component appears within the first k ranked candidates. Detection latency is measured from the recorded onset of a fault to the first valid detection. Time-to-localisation measures the time required to identify a candidate component. The evaluation should define whether latency includes telemetry transport, preprocessing, inference and event correlation.

6.3 Resource and Operational Metrics

The system should report inference throughput, average and tail latency, CPU and memory consumption, storage requirements and telemetry bandwidth overhead. Tests should be repeated under different traffic loads and increasing numbers of monitored devices. The system should also be evaluated during missing telemetry, delayed events, clock skew and model-service interruption. A graceful degradation mode should preserve conventional alarms when the deep learning service is unavailable.

6.4 Comparative Evaluation Plan

The proposed comparison should test each baseline and the hybrid model on identical chronological partitions. An ablation study should remove the CNN branch, LSTM branch, autoencoder score and topology-aware correlation one at a time to estimate their individual contributions. Thresholds should be selected on validation data to meet a stated operational false alarm target. Confidence intervals or repeated-run variability should be reported where feasible. No numerical results are presented in this manuscript; all performance claims must be made only after the experiments are completed.

Evaluation dimension	Suggested measure	Purpose
Detection quality	Precision, recall, macro-F1	Measure correct fault identification across classes
False alarms	False positive rate; alerts per hour	Estimate operator alert burden
Localisation	Top-k accuracy; MRR	Assess ranking of likely faulty components
Timeliness	Detection and localisation latency	Measure operational responsiveness

Efficiency	CPU, memory, throughput	Assess deployment feasibility

7. DISCUSSION

The proposed framework treats fault management as a pipeline rather than an isolated classification task. This distinction is important because operational usefulness depends on whether a detected anomaly can be converted into a trustworthy incident record and an appropriate response. A model that detects a fault but generates hundreds of duplicate alerts may increase workload instead of reducing it. Event correlation, severity estimation and topology-aware localisation are therefore essential components of the design.

7.1 Benefits and Potential Impact

A telemetry-driven deep learning system may help network teams identify subtle degradation before it becomes a major outage, prioritise incidents based on service impact and reduce the time spent manually correlating logs and performance counters. Sequence models can represent evolving behaviour, while anomaly detection can provide a signal for conditions not represented in labelled training data. When integrated with ticketing and orchestration tools, the framework may support a more consistent incident lifecycle from detection to closure.

7.2 Data Quality and Class Imbalance

The quality of the training data is a primary determinant of model reliability. Operational labels may be incomplete, delayed or inconsistent because the actual root cause is often confirmed only after investigation. Some faults occur rarely, and different devices may produce different telemetry patterns for the same underlying event. A model trained on one environment may not generalise to another. Data collection should therefore include diverse traffic conditions, device types and failure scenarios, with explicit documentation of label confidence. Class-weighted learning and anomaly detection can help, but they do not replace representative data.

7.3 Concept Drift and Generalisation

Network behaviour changes as applications, users, topology, routing policies and workloads evolve. This concept drift can cause a model trained on historical observations to generate false alarms or miss new faults. Monitoring should track feature distributions, prediction confidence, reconstruction error and performance on subsequently labelled incidents. Retraining should be triggered by evidence and governed through a controlled process; automatic online learning without validation can introduce instability.

7.4 Explainability and Operator Trust

Network operators need to understand why an incident was raised and which evidence supports the suggested location. Explainability can include the metrics that changed most strongly, the time interval in which the deviation began, related devices and links, similar historical incidents and confidence calibration. Feature attribution methods may be useful, but explanations should be checked for stability and should not be presented as causal proof. The interface should distinguish observed facts, model inferences and recommended actions.

7.5 Security, Privacy and Resilience

Telemetry systems can expose network topology, device identities, traffic patterns and operational weaknesses. Data should be minimised, access-controlled, encrypted in transit and at rest, and retained according to an approved policy. Models and inference endpoints should be protected against tampering, poisoning and adversarial inputs. The management platform itself must be resilient: model failure should not disable basic monitoring, and automated actions should be authenticated, authorised, rate-limited and auditable.

7.6 Human-Supervised Automation

Automation should be introduced gradually. The initial deployment can operate in shadow mode, producing recommendations without changing network state. After validation, low-risk actions such as ticket creation or diagnostic collection may be automated. High-impact changes should require approval, a maintenance policy or a rigorously tested guardrail. Every automated action should have a clear precondition, timeout, rollback path and audit record.

8. IMPLEMENTATION CONSIDERATIONS AND FUTURE DIRECTIONS

8.1 Practical Deployment Roadmap

A staged deployment reduces operational risk. In the first stage, the organisation inventories available telemetry sources and defines a small set of measurable fault scenarios. In the second stage, it builds a data pipeline and baseline monitoring dashboard. In the third stage, the deep learning model is trained and evaluated offline using chronological and scenario-based tests. In the fourth stage, the model runs in shadow mode alongside existing monitoring. The fifth stage introduces operator-reviewed incident recommendations. Only after sufficient evidence is collected should selected low-risk actions be automated.

The inference service can be deployed centrally, at an edge site or in a distributed architecture. Centralised inference simplifies model governance and provides a global view, but it may add latency and depend on telemetry transport. Edge inference can reduce response time and limit data movement, but it increases the complexity of model distribution, resource management and monitoring. A hybrid arrangement may perform local anomaly screening and send selected events to a central correlation service.

8.2 Integration with Existing Network Tools

The framework should avoid replacing established network management systems. Instead, it can integrate through standard telemetry, APIs, message brokers and ticketing interfaces. A normalised incident schema should include event identifier, timestamp, affected service, candidate device or link, severity, model confidence, evidence summary, recommended action and lifecycle status. Integration testing should verify duplicate suppression, event ordering, authentication, rate limits and recovery after temporary disconnection.

8.3 Future Research Directions

- Graph neural networks may improve topology-aware fault localisation by learning relationships among network components.
- Transformer models may capture long-range dependencies across large telemetry windows, subject to latency and resource constraints.
- Federated learning may support collaborative model training across sites without centralising raw telemetry, although privacy leakage and non-identical data distributions require attention.
- Explainable and causal approaches may help distinguish correlated symptoms from plausible root causes.
- Continual learning with drift detection may support adaptation to evolving network conditions while preserving safeguards against model degradation.
- Digital-twin testbeds may enable repeatable fault injection and safe evaluation of proposed remediation policies.

9. CONCLUSION

This article proposed a deep learning-based framework for intelligent network management and automated fault detection. The framework combines telemetry collection, data preparation, CNN and LSTM feature learning, autoencoder-based anomaly scoring, topology-aware event correlation, fault localisation and policy-governed response. It emphasises that operational network management requires more than high classification accuracy: detection latency, false alarm burden, localisation quality, resource overhead, explainability and safe recovery must also be assessed.

The research methodology outlines how public traffic datasets can be complemented with controlled operational telemetry and injected infrastructure faults. It also recommends chronological evaluation, device- or scenario-held-out testing, baseline comparison and ablation analysis. Since no implementation or empirical experiment is reported here, the framework should be treated as a research proposal requiring validation. Future work should implement the architecture in a representative testbed, publish dataset and preprocessing details, compare model families under consistent conditions and evaluate human-supervised automation. Such work can help establish whether deep learning can provide reliable, scalable and accountable support for next-generation network operations.

REFERENCES

[1] F. M. Anis, M. Alabdullatif, S. Aljbli, and M. Hammoudeh, "A Survey on the Applications of Deep Learning in Network Intrusion Detection Systems to Enhance Network Security," IEEE Access, vol. 13, pp. 185357–185373, 2025. doi: 10.1109/ACCESS.2025.3624952.

[2] V. G. da Silva Ruffo, D. M. B. Lent, M. Komarchesqui, V. F. Schiavon, M. V. O. de Assis, L. F. Carvalho, and M. L. Proença Jr., "Anomaly and intrusion detection using deep learning for software-defined networks: A survey," Expert Systems with Applications, vol. 256, article 124982, 2024. doi: 10.1016/j.eswa.2024.124982.

[3] "Unknown, Atypical and Polymorphic Network Intrusion Detection: A Systematic Survey," IEEE Transactions on Network and Service Management, vol. 21, no. 1, pp. 1190–1212, 2024. doi: 10.1109/TNSM.2023.3298533.

[4] International Telecommunication Union, "Recommendation ITU-T Y.3177: Architectural framework for artificial intelligence-based network automation for resource and fault management in future networks including IMT-2020," Feb. 2021. Available: https://www.itu.int/rec/T-REC-Y.3177-202102-I

[5] International Telecommunication Union, "Recommendation ITU-T Y.3061: Autonomous networks—Architecture framework," Dec. 2023. Available: https://www.itu.int/rec/T-REC-Y.3061-202312-I

[6] A. Momand, S. U. Jan, and N. Ramzan, "A Systematic and Comprehensive Survey of Recent Advances in Intrusion Detection Systems Using Machine Learning: Deep Learning, Datasets, and Attack Taxonomy," Journal of Sensors, vol. 2023, article 6048087, 2023. doi: 10.1155/2023/6048087.

[7] Z. Xu, Y. Wu, S. Wang, J. Gao, T. Qiu, Z. Wang, H. Wan, and X. Zhao, "Deep Learning-based Intrusion Detection Systems: A Survey," arXiv:2504.07839, 2025. Available: https://arxiv.org/abs/2504.07839.

[8] "Network anomaly detection and security defense technology based on machine learning: A review," Computers & Electrical Engineering, vol. 119, article 109581, 2024. doi: 10.1016/j.compeleceng.2024.109581.