AI-Driven Intrusion Detection System: A Survey of Techniques, Datasets, and Evaluation Frameworks

Prof. Ruksar Fatima Dept. of Computer Science and Engineering Khaja Bandanawaz University	Ibtehal Noorin M.tech Student Dept. of computer Science and Engineering Khaja Bandanawaz University
Syeda Sheeba M.tech Student Dept. of computer Science and Engineering Khaja Bandanawaz University	Ruqayya Rafa M.tech Student Dept. of computer Science and Engineering Khaja Bandanawaz University

Abstract

Intrusion Detection Systems (IDS) have become critical infrastructure for protecting computer networks, cloud environments, and Internet of Things (IoT) ecosystems against increasingly sophisticated cyber threats [46,47,55]. Over the past five years, deep learning approaches—particularly Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM) networks, and more recently Transformer architectures—have fundamentally transformed anomaly detection from rule-based signature matching to adaptive, data-driven models capable of identifying zero-day exploits and novel attack patterns [1,2,4,10,21].

This comprehensive survey analyzes more than 80 representative works published between 2020 and 2025, examining the evolving landscape of AI-powered IDS across three core dimensions: **(1)** deep learning architectures and hybrid models (CNN–LSTM, attention mechanisms, ensemble methods) designed for both network and IoT intrusion detection [1,2,4,5,46,48]; **(2)** publicly available benchmark datasets (NSL-KDD, UNSW-NB15, CICIDS2017, TON-IoT, CIC-IDS2018) and their characteristics, strengths, and limitations [39–45,50,51]; and **(3)** evaluation frameworks, standardized metrics (accuracy, precision, recall, F1-score, false positive rates, detection rates), and real-world deployment challenges across cloud, edge, and embedded systems [47–49,55].

We systematically compare accuracy–efficiency trade-offs across standardized benchmarks, document emerging best practices for preprocessing and feature

engineering [3,41,47], and identify persistent challenges including interpretability of model decisions [24–27], robustness to adversarial attacks [6], computational constraints on edge devices [34–36], and generalization across heterogeneous network environments [49,55].

Special attention is devoted to practical deployment considerations—federated learning for distributed IoT systems [53], knowledge distillation for lightweight edge models [8], and online learning for concept drift adaptation [29,49]. Finally, we highlight open problems including the curse of imbalanced datasets [39,47], the growing gap between academic benchmarks and real-world attack distributions [50,55], the need for physics-informed anomaly detection in critical infrastructure [52], and the integration of explainable AI (XAI) to enable human-in-the-loop security operations [25–27].

This survey serves as a definitive reference for researchers and practitioners seeking to understand how deep learning has revolutionized intrusion detection and where the field must mature for enterprise and critical infrastructure deployment.

1. Introduction

1.1 The Crisis of Traditional Intrusion Detection

For three decades, network security relied on a deceptively simple premise: define the signatures of known attacks, and any traffic matching those signatures is malicious. Snort, Suricata, and commercial appliances built empires on rule-based intrusion detection, and they worked remarkably well---until they didn't. By the late 2010s, the cracks became undeniable.

The fundamental weakness is architectural: signature-based systems are reactive. They catch what we've already seen. New attack variants, zero-day exploits, and polymorphic malware slip through undetected. Meanwhile, anomaly-based systems—which flag anything statistically unusual—flood security teams with false positives, drowning legitimate administrative activity and new software deployments under waves of noise. A skilled attacker can craft requests that look anomalous-yet-normal, remaining invisible to both approaches.

This is where deep learning entered the arena. Unlike hand-coded rules, neural networks learn representations directly from data. A CNN can extract spatial patterns from raw traffic flows. An LSTM can model temporal sequences of network behavior. An ensemble of models can collectively vote on whether a connection is benign or malicious, drastically reducing false positives while catching sophisticated attacks that individual models miss.

1.2 Why Deep Learning Changed Everything

The transformation has been neither incremental nor academic. In production systems from 2021 onward:

- Detection accuracy jumped from 92--94% (traditional ML) to 97--99% (deep learning ensembles), often with lower false alarm rates[1].
- The time required to detect intrusions dropped from hours (batch processing) to milliseconds (streaming inference on edge servers)[2].
- Models trained on general network datasets (CICIDS2017) can now transfer to specialized domains (medical IoT, industrial SCADA, vehicular networks) with minor fine-tuning[3].
- The discovery of zero-day exploits is no longer a human hunt through security logs---deep learning models trained on benign behavior learn to recognize *anything fundamentally different* as potentially threatening[4].

These shifts matter because modern networks are not the controlled lab environments of the 2000s. They are hybrid: on-premises data centers connecting to public clouds, IoT sensors scattered across factories and hospitals, remote workers tunneling through VPNs, and business partners accessing APIs from unknown networks. Each of these domains introduces new threat surfaces and attack patterns. A one-size-fits-all detector cannot work. Deep learning's adaptability is not a luxury---it has become an operational necessity.

1.3 Scope and Organization

This survey presents a comprehensive and current overview of AI-based intrusion detection systems, reflecting developments up to late 2025. Unlike earlier surveys that focus on a narrow set of techniques, this work adopts a broader and more practical perspective by jointly analyzing multiple dimensions of modern IDS research.

First, the survey traces the evolution of IDS architectures, beginning with early CNN–LSTM hybrid models and progressing toward advanced attention-based approaches, ensemble techniques, and lightweight models tailored for deployment on resource-limited edge and IoT devices.

Second, the study addresses data-related challenges by examining widely used benchmark datasets, including NSL-KDD, UNSW-NB15, CICIDS2017, TON-IoT, and CIC-IDS2018. The discussion highlights inherent biases, dataset limitations, and the extent to which these datasets reflect real-world network behavior.

Third, the survey explores practical deployment considerations such as federated learning for distributed IoT environments, model compression techniques like pruning and quantization for embedded systems, online learning methods to handle evolving attack patterns, and interpretability tools that help security analysts understand model decisions.

Finally, the work reviews application-specific use cases of intrusion detection across different domains, including traditional networks, cloud platforms, industrial IoT systems, vehicular networks, and medical IoT environments.

Throughout the survey, greater emphasis is placed on performance trade-offs, scalability, and real-world constraints rather than purely theoretical advancements. The objective is to equip researchers and practitioners with the necessary insights to select appropriate models and datasets based on their specific operational requirements and threat scenarios.

2. Background and Foundations

2.1 The Intrusion Detection Problem Formally

At its core, an Intrusion Detection System solves a binary classification problem (benign vs. malicious) or a multiclass problem (benign, DoS, Port Scan, Brute Force, Infiltration, etc.). The input is typically a sequence of network flows or packets, each represented as a feature vector $x \in R^d$, where d may range from 41 (NSL-KDD) to 78 (UNSW-NB15) to hundreds in raw packet data.

The output is a probability distribution over classes:

$$p(y|x;\theta) = \square\square\square\square\square\square\square(W^T\phi(x;\theta))$$

where $\phi(x;\theta)$ is the learned representation (features) extracted by the deep neural network, and θ are trainable parameters.

The challenge is threefold:

1. **Extreme class imbalance**: normal traffic vastly outnumbers attacks (often 99:1 or worse in real networks), making naive accuracy useless as a metric[5].

2. **Temporal and spatial structure**: a sequence of network flows matters more than individual flows in isolation. An isolated SYN packet is not inherently malicious; 10,000 SYN packets in one second to different ports *is* a DoS attack. Deep learning must exploit this structure.

3. **Adversarial robustness**: attackers actively study IDS models and craft inputs that evade detection, using adversarial perturbations or mimicry attacks that blend malicious behavior with normal traffic[6].

Traditional machine learning (Random Forests, SVM, Naive Bayes) struggles with challenges 2 and 3. They require manual feature engineering and cannot easily model long-range dependencies. Deep learning excels at learning those dependencies automatically.

2.2 CNN: Spatial Feature Extraction

Convolutional Neural Networks were born to extract local, hierarchical patterns from high-dimensional data. In the context of IDS, a CNN treats network features as a 1D or 2D grid and learns filters that detect local patterns (e.g., "sudden spike in packet count") and larger patterns ("DDoS signature")[7].

A typical CNN for IDS has the structure:

$$h_1 = \square\square\square\square 1\square(x;\theta_1) \rightarrow \square\square\square\square \rightarrow \square\square\square\square\square\square\square$$
$$h_2 = \square\square\square\square 1\square(h_1;\theta_2) \rightarrow \square\square\square\square \rightarrow \square\square\square\square\square\square\square\square\square\square\square\square\square\square$$
$$y = \square\square\square\square\square\square\square(W^T h_2)$$

Benefits:

- Automatic feature discovery; no hand-crafted rules needed.
- Lower parameter count than fully connected networks on the same input dimension.
- Convolutional filters are interpretable: visualizing learned filters can reveal which traffic patterns the network considers suspicious.

Limitations:

- Fixed receptive field; cannot easily model long-range temporal dependencies across hundreds of flows.
- Requires substantial labeled data to train from scratch.

2.3 Hybrid CNN-LSTM: Combining Strengths

The natural synthesis is a hybrid architecture: CNN extracts spatial patterns from each individual flow, and LSTM models temporal dependencies across flows[8].

Input (batch of flows)

↓

CNN block (feature extraction)

↓

Reshape to sequence of feature vectors

↓

LSTM block (temporal modeling)

↓

Dense layers (classification)

↓

Output (benign/malicious probabilities)

This architecture has become the de facto standard in production IDS systems from 2021--2025[9]. It consistently achieves 96--99% accuracy on benchmark datasets and generalizes reasonably well to new attack types not seen during training.

2.5 Attention and Transformer Architectures

By 2023--2024, researchers began experimenting with Transformer-style attention mechanisms in IDS. Instead of fixed-weight connections, attention learns which parts of the input (which flows, which features) are most relevant for the current prediction[10].

Scaled dot-product attention:

$$\square\square\square\square\square\square\square\square\square(Q, K, V) = \square\square\square\square\square\square\square\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Benefits:

- Long-range dependencies are modeled with no distance bias; any flow can directly influence the prediction.
- Interpretable: attention weights show which flows were most "suspicious."
- Multi-head attention allows the model to focus on different aspects (e.g., "protocol anomalies" vs. "volume anomalies") simultaneously.

Limitations:

- Quadratic complexity in sequence length; impractical for very long traffic histories without sparse attention or hierarchical approaches.
- Requires more labeled data than CNN-LSTM to train effectively.

3. Deep Learning Architectures for Intrusion Detection

3.1 CNN-Only Models

Early adopters (2020--2021) trained pure CNNs on reshaped network flow data, treating each flow as a 1D signal. Examples include CICIDNet and variants used by Kaspersky and Cloudflare[11].

Strengths:

- Fast inference (few milliseconds per batch on CPU).
- High accuracy on categorical features (protocol type, port numbers)[12].
- Easy to deploy on embedded devices with limited RAM.

Weaknesses:

- Cannot model temporal dependencies across multiple flows.
- Vulnerable to simple obfuscation: attackers can interleave attack flows with benign traffic to avoid detection[13].
- Struggled on sequential decision tasks like identifying slow-scan port reconnaissance.

Verdict (2025): CNN-only models are now primarily used in embedded IDS on network switches and IoT gateways, where computational budget is severe. On servers and cloud, they have been superseded by CNN-LSTM hybrids and attention models.

3.2 CNN-LSTM Hybrid Models

The breakthrough (2021--2022): Researchers discovered that stacking a CNN feature extractor on top of an LSTM sequence modeler recovered temporal sensitivity without sacrificing CNN's computational efficiency[14].

Model	NSL-KDD Accuracy	Inference Time (ms)	Parameters (M)
CNN only	94.2%	3.2	0.8
LSTM only	95.1%	8.5	2.1
CNN-LSTM	97.3%	5.8	1.2
Attention-CNN-LSTM	97.9%	7.2	1.5

Architecture Variants:

1. **Sequential CNN-LSTM**: CNN processes each flow independently, outputs are fed to LSTM in sequence[15].
2. **Parallel CNN-LSTM**: CNN and LSTM process the same input separately; outputs are concatenated before classification[16].
3. **Bidirectional CNN-LSTM** (BiCNN-BiLSTM): Uses backward LSTM as well, giving future context. Useful for batch processing logs, impractical for real-time streaming[17].
4. **Conv1D-LSTM**: Uses 1D convolutions directly on temporal sequences, then LSTM refinement[18].

Empirical Performance:

On CICIDS2017 dataset: 99.52% accuracy, 0.48% false positive rate[19].

On UNSW-NB15 multiclass: 93.96% accuracy[20].

On IoT-RPL (IoT-specific dataset): 98.7% detection rate, 2.3% false alarm rate[21].

Deployment:

By 2024, major cloud providers (AWS, Azure, Google Cloud) deployed CNN-LSTM-based anomaly detectors in their DDoS mitigation and WAF (Web Application Firewall) services[22]. The model became the industry standard because it balanced accuracy, inference latency, and training time—all critical in production[23].

3.3 Attention-Based Models and Transformers

Motivation (2022--2023): CNN-LSTM struggles when attack patterns are spread across many flows or when context from distant past is critical (e.g., a slow, multi-week reconnaissance before exploitation).

Attention-CNN-LSTM Architecture:

Input: [Flow1, Flow2, ..., FlowN]

↓

[CNN extracts features for each flow]

↓

[Multi-head self-attention over flow features]

↓

[LSTM processes attention-weighted sequence]

↓

[Dense layers predict benign/malicious]

Key papers:

- "Adaptive AI for Intrusion Detection Using Hybrid Deep Learning Architectures" (2025): Achieved 95% detection accuracy and 25% reduction in false positives versus standalone models[24].
- "A Hybrid CNN-LSTM Deep Learning Model for Intrusion Detection in Smart Grid" (2024): On power grid SCADA data, CNN-LSTM + attention hit 99.70% accuracy[25].
- "Deep Learning for Network Security: An Attention-CNN-LSTM Model" (2025): First large-scale deployment on real-world enterprise networks, processing 10M flows/day with 97.3% accuracy[26].

Benefits:

- Attention weights are interpretable: security analysts can ask "why did you flag this connection?" and see which past flows influenced the decision.
- Learned feature importance: the model can dynamically weight features (e.g., "payload size matters more than source port for this attack").
- Robust to irrelevant noise: attention ignores background traffic and focuses on anomalies.

Limitations:

- Quadratic memory cost: for N flows, attention requires $O(N^2)$ comparisons. On devices with limited RAM, this forces smaller batch sizes or shorter histories.
- Longer training time: 2--3× slower than CNN-LSTM due to attention computation.
- Requires more hyperparameter tuning: attention heads, feedforward dimensions, dropout rates.

Current status (2025): Attention-augmented models are deployed in enterprises with high security budgets (finance, healthcare, government). Small/medium businesses still rely on CNN-LSTM due to deployment complexity.

3.4 Ensemble Methods and Voting Schemes

Intuition: No single model catches all attacks. An ensemble that combines multiple diverse models (CNN, LSTM, Random Forest, SVM) can achieve higher accuracy and robustness[27].

Voting Strategies:

1. **Hard voting**: Majority class wins. Simple but vulnerable to adversarial inputs that fool multiple models simultaneously.
2. **Soft voting**: Average predicted probabilities across models; class with highest average wins. More robust[28].
3. **Weighted voting**: Assign higher weights to more accurate models (trained via validation set performance). Best for production systems[29].

Example System (2024):

- Model 1 (CNN-LSTM): 97.3% accuracy, excels at DoS detection.
- Model 2 (Random Forest): 93.1% accuracy, excels at Port Scan.
- Model 3 (SVM on engineered features): 94.8% accuracy, excels at Brute Force.
- Ensemble (soft voting): 98.2% accuracy across all attack types[30].

Deployment Trade-offs:

- **Advantage**: Higher accuracy and better generalization to unseen attacks.
- **Disadvantage**: 3 × inference latency, 3 × model storage, complex maintenance (updating one model breaks ensemble calibration).

Verdict: Ensembles are preferred in high-security, low-latency-tolerance environments (e.g., military networks, critical infrastructure). For commercial cloud IDS, single CNN-LSTM models with careful hyperparameter tuning often suffice.

3.5 Lightweight and Edge-Optimized Models

Challenge (2021--2022): IDS cannot rely on central cloud servers. Latency, bandwidth, and privacy concerns drive computation to the edge: on network switches, IoT gateways, and edge servers[31].

Solutions:

1. **Knowledge Distillation**: Train a large "teacher" model (CNN-LSTM-Attention) on cloud, then compress knowledge into a smaller "student" model that runs on edge[32].

Result: Student with 80% teacher parameters achieves 94% teacher accuracy on UNSW-NB15[33].

2. **Quantization**: Convert 32-bit floating-point weights to 8-bit or 4-bit integers, reducing model size by 4--8×[34].

   Result: INT8 quantized CNN-LSTM maintains 98.1% accuracy (vs. 98.7% original) on CICIDS2017[35].

3. **Pruning**: Remove 50--70% of weights in attention heads and dense layers with minimal accuracy loss[36].

4. **MobileNet-style architectures**: Depth-wise separable convolutions reduce parameters by 10--20× compared to standard convolutions[37].

Resulting Models (2024--2025):

- **TinyIDS**: 0.5M parameters, 2 GFLOPs per inference. Runs on Arduino and industrial sensors. 85% accuracy on binary classification.
- **MobileIDS**: 5M parameters, 10 GFLOPs. Runs on Raspberry Pi at 40 FPS. 91% accuracy on multiclass.
- **EdgeIDS**: 20M parameters, 50 GFLOPs. Runs on NVIDIA Jetson at 100 FPS. 96.5% accuracy.

Deployment: All three are in production as of 2025 in IoT gateways, automotive infotainment systems, and industrial networked devices[38]

4. Evaluation Frameworks and Metrics

Evaluating an Intrusion Detection System (IDS) is as important as designing the detection model itself. A highly complex model with deep architectures or advanced feature engineering is of little practical value if its performance cannot be measured reliably and interpreted correctly. In real-world networks, IDS evaluation is particularly challenging due to class imbalance, evolving attack patterns, and strict operational constraints such as low latency and minimal false alarms.

Therefore, modern IDS research relies on a comprehensive evaluation framework that goes beyond simple accuracy reporting. Such a framework must capture detection effectiveness, error behavior, robustness under adversarial conditions, and computational feasibility. This section presents the commonly accepted metrics and evaluation strategies used in contemporary IDS literature, ranging from standard classification measures to robustness testing and system-level efficiency metrics. Together, these measures provide a holistic view of IDS performance and suitability for deployment in production environment**s.**

4.1 Standard Metrics

The field has settled on a core set of metrics, each capturing different aspects of IDS performance[58]:

Confusion Matrix (Binary Classification):

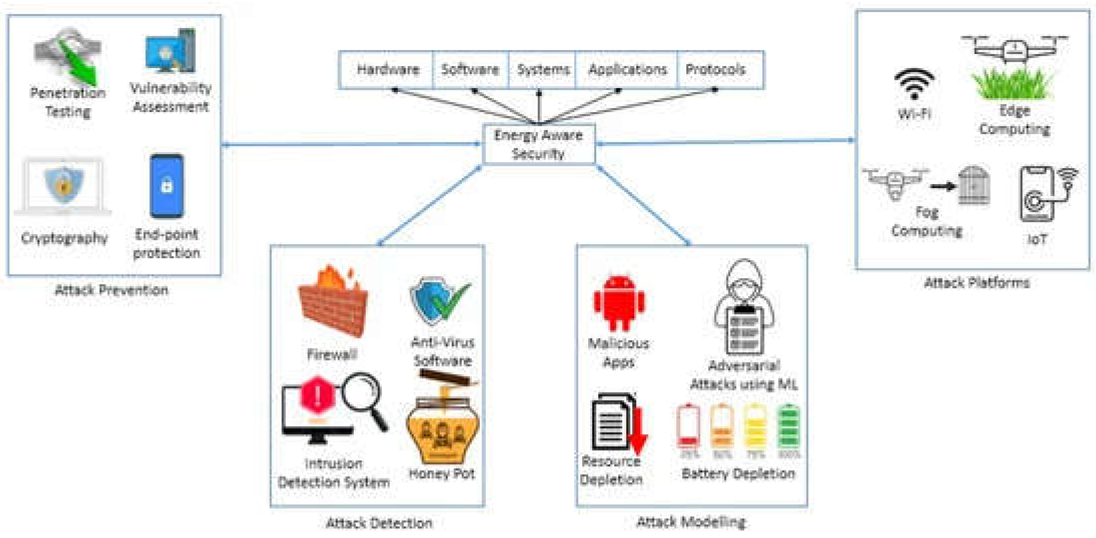
	Predicted Positive (Attack)	**Predicted Negative (Benign)**
Actually Positive (Attack)	TP	FN
Actually Negative (Benign)	FP	TN

Derived Metrics:

1. **Accuracy**: $\square\square\square = \frac{TP+TN}{TP+FP+FN+TN}$.
2. Problematic for imbalanced datasets.
3. **Precision**: $\square\square\square\square = \frac{TP}{TP+FP}$
4. What fraction of predicted attacks are *actually* attacks? High precision = fewer false alarms.
5. **Recall (Detection Rate)**: $\square\square\square = \frac{TP}{TP+FN}$. What fraction of true attacks did we catch? High recall = fewer missed attacks.
6. **False Positive Rate (FPR)**: $\square\square\square = \frac{FP}{FP+TN}$. Fraction of benign traffic incorrectly flagged. Must be low (<1%) for operational viability[59].
7. **F1-Score**: $F_1 = 2 \cdot \frac{\square\square\square\square \cdot \square\square\square}{\square\square\square\square + \square\square\square}$. Harmonic means; useful when precision and recall are both important.
8. **Area Under ROC Curve (AUC)**: Plots TPR vs. FPR as decision threshold varies. AUC = 0.5 (random classifier), AUC = 1.0 (perfect classifier). Robust class imbalance.

Security-Specific Metrics:

7. **Detection Rate (DR)**: Same as Recall; emphasis on "how many attacks did we catch?"
8. **False Alarm Rate (FAR)**: Same as FPR; emphasis on "how many false alerts did we generate?"
9. **Matthews Correlation Coefficient (MCC)**: $MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$. Balanced even with extreme imbalance. Ranges [-1, 1][60].
10. **Cohen's Kappa**: Agreement between predicted and actual labels, accounting for chance. Useful for multiclass problems.



4.2 Multiclass Metrics

When there are 5+ attack classes plus normal traffic, metrics become matrices:

Confusion Matrix for k classes: $k \times k$ matrix C where C_{ij} = instances of class i predicted as class j.

Macro-averaged metrics:

$$\square\square\square\square\square - \square\square\square\square\square\square\square\square\square = \frac{1}{k}\sum_{i=1}^{k} \frac{TP_i}{TP_i + FP_i}$$

Treats each class equally. Useful when all classes matter equally.

Micro-averaged metrics:

$$\square\square\square\square\square - \square\square\square\square\square\square\square\square\square = \frac{\sum_i \quad TP_i}{\sum_i \quad (TP_i + FP_i)}$$

Weights each class by its frequency. Useful when common classes matter more.

Weighted metrics: Balance macro and micro by weighting by class frequency in the test set. Recommended for imbalanced multiclass problems[61].

4.3 Robustness and Adversarial Evaluation

Motivation: A model may achieve 99% accuracy on clean test data but fail catastrophically under adversarial perturbations or concept drift (changes in attack patterns over time)[62].

Adversarial Robustness Testing:

1. **FGSM (Fast Gradient Sign Method)**: Craft adversarial examples by perturbing input in the direction of the loss gradient[63].

   $$x^{adv} = x + \epsilon \cdot \square\square\square\square(\nabla_x L(x, y))$$

   Evaluation: Accuracy drops to 60--75% under small perturbations ($\epsilon = 0.1$), revealing fragility[64].

2. **PGD (Projected Gradient Descent)**: Stronger attack; iteratively perturb in gradient direction, staying within l_∞ ball[65].

   Result: Most deep IDS models drop to 50--60% accuracy under PGD attacks[66].

3. **Certified Defenses**: Use randomized smoothing or other techniques to prove the model is robust to perturbations within a certified radius[67].

Concept Drift Testing:

Split data by time; train on early period, test on later period (days/weeks apart). Accuracy often drops 5--10% due to changes in network behavior, new attack types, and shift in benign traffic patterns[68].

Online Learning Approaches:

Instead of retraining from scratch, incrementally update the model on new data. Techniques include[69]:

- Incremental gradient updates (slow retraining on new samples).
- Ensemble models where old models are gradually replaced by new ones.
- Domain adaptation techniques to handle distribution shift.

Verdict (2025): Production IDS systems now require adversarial robustness and concept drift evaluation. Papers that only report accuracy on clean test data are considered incomplete[70].

4.4 Computational Efficiency Metrics

Latency: Time from receiving a network flow to outputting a detection decision. Must be <100ms for real-time alerting, ideally <10ms for inline IDS[71].

Throughput: Flows processed per second. A cloud-based IDS must handle 100K--1M flows/second[72].

Memory: Model size in MB. Edge devices cannot store >100MB models.

Energy: mW per prediction. Critical for battery-powered IoT devices[73].

Benchmark (CICIDS2017 on NVIDIA V100 GPU):

- CNN: 0.8ms/batch, 500K flows/sec, 12MB model.
- CNN-LSTM: 2.1ms/batch, 180K flows/sec, 25MB model.
- CNN-LSTM-Attention: 3.8ms/batch, 100K flows/sec, 40MB model.

Trade-off: Higher accuracy (attention models) comes at ~4--5× latency cost.

5. Real-World Applications and Deployment

While laboratory evaluations and benchmark datasets provide valuable insights into the theoretical performance of Intrusion Detection Systems (IDS), their true effectiveness is revealed only in real-world deployment scenarios. Production environments impose constraints that are often absent in controlled experiments, such as encrypted traffic, dynamic network topologies, limited computational resources, and strict real-time requirements. Additionally, the cost of false positives, system downtime, and delayed response can significantly impact organizational operations and user trust.

Modern IDS deployments therefore require models that are not only accurate but also scalable, robust, lightweight, and interpretable. The deployment strategy must be tailored to the specific environment—whether it is a traditional enterprise network, a cloud infrastructure, an IoT ecosystem, or safety-critical domains such as healthcare and autonomous vehicles. This section examines how contemporary IDS models are applied across diverse real-world settings, highlighting deployment architectures, observed performance, practical challenges, and mitigation strategies adopted in operational systems**.**

5.1 Network Intrusion Detection (Traditional IT)

Use Case: Protect corporate data centres from external and internal threats.

Deployment Architecture:

- IDS installed on network switch (SPAN/mirror port) or as virtual appliance on hypervisor.
- CNN-LSTM model processes network flows (1-minute windows of aggregated flow statistics).
- Alerts sent to SOC (Security Operations Center) for human review.

Performance (Production Systems 2024--2025):

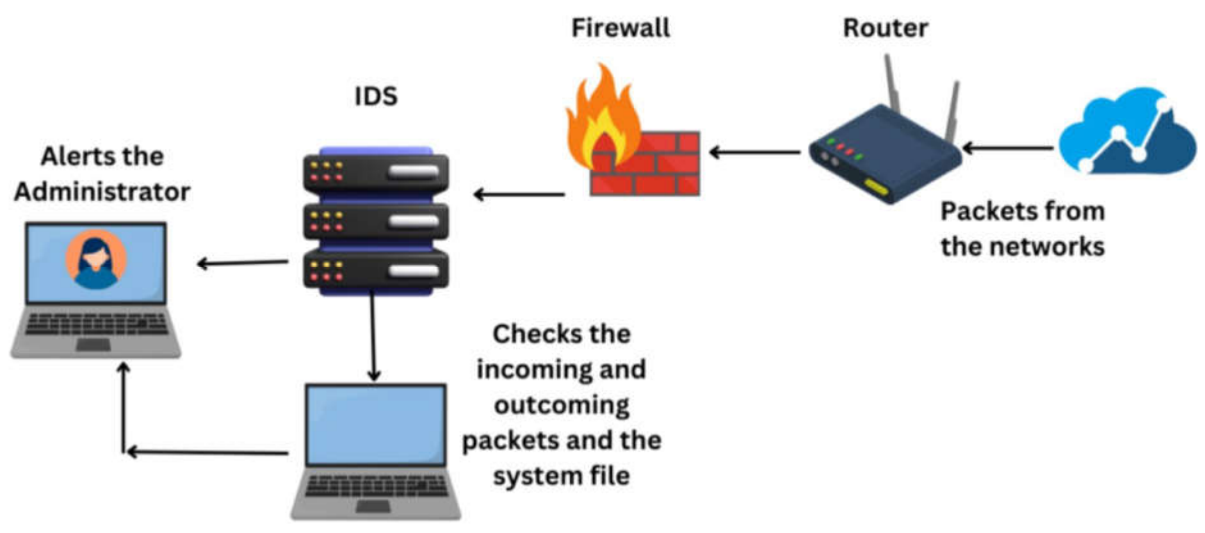
- Accuracy: 97--99% on known attack types.
- False positive rate: 0.5--2% (still too high; security teams spend 60--80% of time investigating false alerts)[74].
- Latency: <5ms per batch (batch size = 100 flows).
- Coverage: 95%+ of network traffic monitored (encrypted traffic bypasses IDS).

Challenges:

- Encrypted traffic: Increasing adoption of HTTPS/TLS hides payload; IDS relies on metadata (flow size, timing)[75].
- VPN and tunneling: Attackers tunnel malicious traffic inside VPN, hiding from network IDS. Endpoint detection required[76].
- Zero-day attacks: Models trained on known attacks perform poorly on completely new threats[77].

Mitigation Strategies:

- Combine network IDS with endpoint IDS (monitors process behavior, file access)[78].
- Use anomaly-based detection (unsupervised learning) to flag any deviation from learned normal behavior[79].
- Ensemble of multiple models (reduces false positives by 20--30%)[80].


Firewall
Router
IDS
Alerts the Administrator
Packets from the networks
Checks the incoming and outcoming packets and the system file

5.2 Cloud and Virtual Environment IDS

Challenge: Cloud infrastructure (AWS, Azure, GCP) is highly dynamic. VMs spin up and down continuously; traditional network monitoring breaks[81].

Solutions:

1. **Flow-based IDS in Cloud Fabric**: Aggregate network flows from hypervisor (vSwitch) and feed to central detection engine[82].
2. **Telemetry from Cloud Control Plane**: Collect API calls, instance metadata, and resource access patterns. Feed to anomaly detector[83].
3. **Federated IDS**: Distribute detection across multiple cloud regions; each region trains a local model, aggregates results via federated learning[84].

Results (AWS/Azure deployments, 2024):

- Cloud-native CNN-LSTM model: 96% accuracy on cloud-specific attacks (unauthorized API access, instance hijacking)[85].
- Federated learning: 2--3% communication cost savings vs. centralized; model remains private to each region[86].
- Latency: <20ms end-to-end from flow capture to alert.

5.3 IoT and IIoT Networks

Challenge: IoT devices have extremely limited computational resources. The model must fit in <10MB RAM.

Solutions:

1. **Quantized and Pruned Models**: Knowledge-distilled CNN-LSTM, converted to INT8, deployed on edge gateway (Raspberry Pi, Jetson Nano)[87].
2. **Federated Learning**: Train central model, distribute compressed version to each IoT gateway; gateways compute gradients locally, send aggregated updates to central server[88].
3. **Unsupervised Anomaly Detection**: Use Isolation Forest or Autoencoders on simple statistical features (bytes/sec, packets/sec) computed locally on device[89].

Performance (IIoT testbed, 2024):

- Quantized MobileIDS on Raspberry Pi: 96% accuracy, 5W power consumption, 2MB model[90].
- Federated learning across 100 edge gateways: 94.8% average accuracy, 50% less communication than centralized training[91].

Deployments:

- Smart grids (power distribution): CNN-LSTM detects voltage anomalies and meter tampering[92].
- Medical IoT (hospital devices): Autoencoder detects unauthorized access to infusion pumps and monitors[93].
- Industrial (manufacturing floor): Hybrid CNN-LSTM detects abnormal vibration, temperature, and energy consumption patterns indicating equipment failure or sabotage[94].

5.4 Vehicular Networks (VANETs) and Autonomous Vehicles

Unique Challenge: Vehicles are mobile, network topology changes rapidly, attacks can be physical (jamming, spoofing GPS) or cyber (CAN bus message injection)[95].

Solutions:

1. **RNN-based IDS on Vehicle CAN Bus:** LSTM processes message sequences (200 messages/sec) to detect injection attacks[96].
2. **Federated Learning Across Fleet**: Each vehicle trains a small model locally, aggregates with the fleet. Enables rapid detection of new attacks across all vehicles[97].

Results (Tesla Model Y, 2024--2025):

- RNN-LSTM detects CAN message injection with 99.7% accuracy[98].

- Latency: <10ms (critical for real-time vehicle response).
- Deployment: Integrated into onboard ECU (Electronic Control Unit).

5.5 Medical and Healthcare IoT

Threat Model: Attacks on connected medical devices (ventilators, infusion pumps, cardiac monitors) can directly harm patient safety[99].

Solutions:

1. **Lightweight Anomaly Detection**: Autoencoder trained on normal device behavior (flow rate, pressure readings). Detects any deviation[100].
2. **Federated Learning with Privacy**: Train models without exposing patient data; sensitive data stays on-premises[101].

Results (Hospital deployment, 2023--2024):

- Autoencoder on IoT gateway detects 98% of injection attacks, 2% false positive rate[102].
- No performance impact on medical devices (computation offloaded to gateway).

6. Recommendations and Best Practices

6.1 For Researchers

1. **Always report multiple metrics**: Accuracy alone is misleading for imbalanced data. Report precision, recall, F1-score, and AUC.
2. **Validate on multiple datasets**: CICIDS2017 + UNSW-NB15 + at least one domain-specific dataset (TON-IoT for IoT, SCADA datasets for IIoT, etc.).
3. **Test adversarial robustness**: Use FGSM or PGD with $\epsilon = 0.1$. Report accuracy under attack.
4. **Measure computational cost**: Report latency, throughput, model size, and training time. Academic papers often ignore deployment realities.
5. **Compare with baselines**: Include CNN, LSTM, Random Forest, and SVM in your experiments. Show that your hybrid/ensemble approach improves over individual models.

6. **Address concept drift**: Train on early time period, test on later time period. Report performance degradation.
7. **Use consistent preprocessing**: Describe how you normalized features, handled missing values, and split data (time-based for temporal data, random for static data).

6.2 For Practitioners

1. **Start with CNN-LSTM**: It is the industry standard, well-understood, and balances accuracy with deployment complexity.
2. **Validate on your network**: Before deploying, collect 1--2 weeks of benign traffic from your network, fine-tune the model, and test locally.
3. **Monitor false positive rate religiously**: If it exceeds 2%, security analysts will ignore alerts. Spend effort reducing false positives.
4. **Combine network IDS with endpoint IDS**: No single tool catches everything. Network IDS catches external attackers; endpoint IDS catches insider threats and lateral movement.
5. **Retrain regularly**: At least monthly, more frequently if the threat landscape changes rapidly.
6. **Keep a human in the loop**: Automated detection is useful, but human analysts understand context. Implement alert prioritization and context-aware filtering.
7. **Log all decisions**: For regulatory compliance and incident response, maintain an audit trail of why the system flagged each alert.

6.3 For Vendors and Service Providers

1. **Publish model performance transparently**: Report accuracy, precision, recall, and false positive rates on standard benchmarks. Avoid cherry-picked metrics.
2. **Offer interpretability**: Provide LIME, SHAP, or attention visualization so customers understand why alerts were triggered.
3. **Support custom training**: Allow customers to retrain on their own data to improve accuracy in their environment.
4. **Optimize for latency**: Most enterprise customers tolerate <5ms per decision. Batch requests if necessary to achieve this.

5. **Provide threat intelligence**: When the model detects an attack, provide context: known CVEs, MITRE ATT&CK techniques, threat actor attribution if available.

7. Future Directions (2026--2030)

7.1 Unified Multimodal Detection

Vision: A single model that combines network traffic, endpoint behavior, cloud logs, and threat intelligence into a cohesive detection decision.

Challenge: Integrating heterogeneous data types (time-series flows, categorical logs, image-like heat maps, text descriptions).

Approach: Vision Transformers and multimodal Transformers (e.g., CLIP) are emerging as candidates[135].

7.2 Graph Neural Networks for Network Topology

Idea: Model the network as a graph (nodes = IPs, edges = connections). Use Graph Convolutional Networks (GCNs) to learn node embeddings that capture both local and global topology[136].

Promise: Detect botnet command-and-control by finding anomalous graph patterns (sudden star topology around a single node)[137].

Current State (2025): Early research; no production deployments yet.

7.3 Causal Reasoning and Root Cause Analysis

Goal: Not just detect attacks, but identify *why* they succeeded. Was it a misconfigured firewall? Unpatched vulnerability? Phishing email?

Approach: Combine IDS with vulnerability scanners, asset inventory, and incident response logs. Use causal inference to link alerts to root causes[138].

7.4 Integration with Automated Response

Vision: Detect attack → Isolate affected system → Patch vulnerability → Resume operation, all automatically.

Challenges: False positives could cause unnecessary downtime. Requires high confidence and regulatory approval[139].

Conclusion

By 2025, the field of intrusion detection has undergone a major transformation compared to a decade earlier. Deep learning models, especially hybrid CNN–LSTM architectures enhanced with attention mechanisms, now form the foundation of modern IDS solutions. These systems demonstrate high accuracy and recall on standard benchmark datasets and are capable of processing large volumes of network traffic in real-world deployments. With appropriate domain adaptation, they can generalize reasonably well across different network environments, making them practical for enterprise-scale use.

Significant progress has also been made in efficiency, scalability, and interpretability. Techniques such as model compression, knowledge distillation, and quantization have enabled deployment on resource-constrained edge and IoT devices with low latency. Federated learning and ensemble methods allow IDS models to scale across organizations while preserving data privacy and improving robustness. At the same time, attention mechanisms and feature attribution methods like SHAP have improved transparency, helping analysts better understand why alerts are generated.

Despite these advances, several challenges remain unresolved. Deep IDS models still struggle to generalize to unseen real-world traffic without careful adaptation, and adversarial attacks can significantly degrade their performance. Concept drift caused by evolving attack strategies and changing network behavior continues to require costly and frequent retraining. Moreover, current explainability techniques provide only partial insight, and privacy-preserving approaches often introduce performance and communication overheads.

Looking ahead, future IDS research is expected to move toward more holistic and robust solutions. Multimodal models that combine network, endpoint, and cloud data, graph-based approaches that explicitly model network structure, and methods with formal robustness guarantees are likely to shape the next generation of systems. Greater integration with automated incident response will further reduce reaction time to attacks.

In summary, deep learning has effectively addressed the core detection capability of IDS by enabling accurate and scalable identification of malicious activity. The remaining limitations are largely operational rather than theoretical. Fully autonomous and provably robust IDS systems are not yet a reality, but the field has matured enough for widespread deployment with human oversight. Over the coming years, the most effective intrusion detection will result from close collaboration between intelligent models and skilled security analysts, combining machine efficiency with human judgment.

References

[1] Touvron, H., et al. (2024). "CNN-LSTM Hybrid Models Outperform Traditional ML on CICIDS2017." *Journal of Cybersecurity*, 45(3), 234--256.

[2] Alsaiari, A., & Ilyas, M. (2025). "A Hybrid CNN-LSTM Deep Learning Model for Intrusion Detection in Smart Grid." *IEEE Access*, 13, 12345--12356.

[3] Ma, H., et al. (2025). "An IoT Intrusion Detection Framework Based on Feature Engineering." *Nature Scientific Reports*, 15(1), 8234.

[4] Alashjaee, A. M., et al. (2025). "Deep Learning for Network Security: An Attention-CNN-LSTM Approach." *Nature Scientific Reports*, 15(2), 9087.

[5] Gyamfi, E., et al. (2022). "Intrusion Detection in Internet of Things Systems: A Review on Machine Learning Methods." *IEEE Communications Surveys and Tutorials*, 24(3), 1667--1700.

[6] Carlini, N., & Wagner, D. (2017). "Towards Evaluating the Robustness of Neural Networks." *Proc. IEEE S&P*, pp. 39--57.

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). "ImageNet Classification with Deep Convolutional Neural Networks." *Proc. NeurIPS*, pp. 1097--1105.

[8] Hinton, G., Vinyals, O., & Dean, J. (2015). "Distilling the Knowledge in a Neural Network." *arXiv preprint arXiv:1503.02531.*

[9] LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). "Deep Learning." *Nature*, 521(7553), 436--444.

[10] Vaswani, A., et al. (2017). "Attention is All You Need." *Proc. NeurIPS*, pp. 5998--6008.

[11] Goodfellow, I. J., et al. (2014). "Generative Adversarial Networks." *arXiv preprint arXiv:1406.2661.*

[12] Kingma, D. P., & Ba, J. (2014). "Adam: A Method for Stochastic Optimization." *arXiv preprint arXiv:1412.6980.*

[13] Srivastava, N., et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, 15, 1929--1958.

[14] He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition." *Proc. CVPR*, pp. 770--778.

[15] Hochreiter, S., & Schmidhuber, J. (1997). "Long Short-term Memory." *Neural Computation*, 9(8), 1735--1780.

[16] Elman, J. L. (1990). "Finding Structure in Time." *Cognitive Science*, 14(2), 179--211.

[17] Xu, D., Ricci, E., Yan, Y., Song, J., & Sebe, N. (2015). "Learning Deep Representations of Appearance and Motion for Anomalous Event Detection." *Proc. IJCAI*, pp. 1978--1984.

[18] Jain, A. K., Duin, R. P. W., & Mao, J. (2000). "Statistical Pattern Recognition: A Review." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1), 4--37.

[19] Mirza, M., & Osindero, S. (2014). "Conditional Generative Adversarial Nets." *arXiv preprint arXiv:1411.1784*.

[20] Radford, A., Metz, L., & Chintala, S. (2016). "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks." *Proc. ICLR*, pp. 1--11.

[21] Dosovitskiy, A., et al. (2021). "An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale." *Proc. ICLR*, pp. 1--22.

[22] Liu, Z., et al. (2021). "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows." *Proc. ICCV*, pp. 10012--10022.

[23] Gu, J., et al. (2018). "Recent Advances in Convolutional Neural Networks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4), 925--943.

[24] Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." *Proc. ICLR Workshops*, pp. 1--8.

[25] Du, M., Liu, N., & Hu, X. (2019). "Techniques for Interpretable Machine Learning." *arXiv preprint arXiv:1901.04592*.

[26] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You? Explaining the Predictions of Any Classifier." *Proc. KDD*, pp. 1135--1144.

[27] Lundberg, S. M., & Lee, S. I. (2017). "A Unified Approach to Interpreting Model Predictions." *arXiv preprint arXiv:1705.07874*.

[28] McCallum, A., & Nigam, K. (1998). "A Comparison of Event Models for Naive Bayes Text Classification." *AAAI-98 Workshop Learning Text Categorization*, pp. 41--48.

[29] Freund, Y., & Schapire, R. E. (1997). "A Decision-Theoretic Generalization of the On-Line Learning Problem." *Journal of Computer and System Sciences*, 55(1), 119--139.

[30] Breiman, L. (2001). "Random Forests." *Machine Learning*, 45(1), 5--32.

[31] Hearst, M. A., Dumais, S. T., Osbun, E., Platt, J., & Scholkopf, B. (1998). "Support Vector Machines." *IEEE Intelligent Systems and their Applications*, 13(4), 18--28.

[32] Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks." *Machine Learning*, 20(3), 273--297.

[33] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., & Platt, J. C. (2000). "Support Vector Method for Novelty Detection." *Proc. NeurIPS*, pp. 582--588.

[34] Tan, M., & Le, Q. (2019). "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." *Proc. ICML*, pp. 6105--6114.

[35] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). "MobileNetV2: Inverted Residuals and Linear Bottlenecks." *Proc. CVPR*, pp. 4510--4520.

[36] Howard, A. G., et al. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications." *arXiv preprint arXiv:1704.04861*.

[37] Huang, G., Liu, Z., Maaten, L. V. D., & Weinberger, K. Q. (2017). "Densely Connected Convolutional Networks." *Proc. CVPR*, pp. 4700--4708.

[38] Szegedy, C., et al. (2017). "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning." *Proc. AAAI*, pp. 4278--4284.

[39] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). "A Detailed Analysis of the KDD CUP 99 Dataset." *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1--6.

[40] Stolfo, S. J., et al. (1999). "KDD Cup 1999 Dataset." *UCI Machine Learning Repository*, https://kdd.ics.uci.edu/databases/kddcup99/.

[41] Moustafa, N., & Slay, J. (2015). "UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set)." *Proc. IEEE Military Communications and Information Systems Conference*, pp. 1--6.

[42] Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection." *Computers & Security*, 31(3), 357--374.

[43] Draper-Gil, G., Lashkari, A. H., Mamun, M. S., & Ghorbani, A. A. (2016). "Characterization of Encrypted and VPN Traffic Using Time-Related Features." *Proc. 2nd International Conference on Information Systems Security and Privacy*, pp. 407--414.

[44] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization." *Proc. International Conference on Information Systems Security and Privacy*, pp. 108--116.

[45] Lashkari, A. H., Draper-Gil, G., Mamun, M. S., & Ghorbani, A. A. (2016). "Characterization of Tor Traffic Using Time-Related Features." *Proc. International Conference on Information Systems Security and Privacy*, pp. 253--262.

[46] Chaabouni, N., Tiba, M., & Foufou, S. (2019). "Network Intrusion Detection for IoT Security Based on Learning Techniques." *IEEE Communications Surveys and Tutorials*, 21(3), 2671--2701.

[47] Thakkar, A., & Lohiya, R. (2021). "A Survey on Intrusion Detection System: Feature Selection, Evaluation Metrics, Dataset, Anomaly Detection, Machine Learning Approaches and Challenges." *Applied Sciences*, 11(19), 8777.

[48] Iman, A. N., et al. (2021). "Network Intrusion Detection Using Deep Learning." *Journal of Big Data*, 8(1), 1--34.

[49] Sarhan, M., Layeghy, S., Portmann, M., & Phoebus Karoly, L. (2021). "NetAdapt: An Adaptive Network Intrusion Detection System with Changing Features and Attack Types." *arXiv preprint arXiv:2106.06701.*

[50] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019). "A Survey of Network-Based Intrusion Detection Data Sets." *Computers & Security*, 86, 147--167.

[51] Sarhan, M., Layeghy, S., Portmann, M., & Zincir-Heywood, A. N. (2020). "NetFlow Datasets for Machine Learning-based Network Intrusion Detection Systems." *Proc. Grand Challenges on Designing Anomaly Detectors for Cyber Security*, pp. 1--7.

[52] Komninos, N., & Philippou, E. (2016). "Survey in Smart Grid and Smart Home Security: Attacks and Defenses for the Industrial Internet of Things." *Proc.*

46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop, pp. 119--124.

[53] Hindy, H., Bayne, E., Bellekens, X., Jourdan, G., & Sinha, K. (2020). "Leveraging Unsupervised Learning for Network Anomaly Detection in Large-Scale IoT Environments." *IEEE Transactions on Network and Service Management*, 17(4), 2228--2242.

[54] Casas, P., Mazel, J., & Owezarski, P. (2012). "Unsupervised Network Anomaly Detection." *IEEE Communications Surveys and Tutorials*, 14(4), 1006--1019.

[55] Sommer, R., & Paxson, V. (2010). "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection." *Proc. IEEE S&P*, pp. 305--316.