

AI-Driven Intrusion Detection for Resource-Constrained Internet of Things Networks

Dr. Aarav Patel¹

Department of Computer Science and Engineering
Institute of Technology and Management, Ahmedabad, Gujarat, India
Email: aarav.patel@itma.ac.in

Abstract

The rapid expansion of the Internet of Things (IoT) has connected billions of heterogeneous devices to communication networks and cloud-based services. Although IoT provides substantial benefits in automation, monitoring, healthcare, transportation, agriculture, and industrial operations, its large and distributed attack surface creates significant cybersecurity challenges. Conventional intrusion detection systems frequently struggle to operate effectively in IoT environments because IoT networks contain resource-constrained devices, diverse communication protocols, irregular traffic patterns, and continuously changing attack behaviors. Artificial intelligence and machine learning provide an opportunity to develop adaptive intrusion detection mechanisms capable of identifying suspicious network behavior from observed traffic characteristics. This paper presents an AI-driven intrusion detection framework designed specifically for resource-constrained IoT environments. The proposed framework combines traffic preprocessing, feature reduction, lightweight machine learning, anomaly scoring, and edge-based decision making. Instead of requiring every IoT device to execute computationally expensive detection algorithms, the architecture transfers major analytical operations to edge gateways. The proposed design also considers class imbalance, false-positive reduction, model updating, explainability, and computational efficiency. A mathematical formulation for anomaly scoring and risk estimation is introduced, followed by an evaluation methodology based on accuracy, precision, recall, F1-score, detection latency, memory consumption, and computational overhead. The study further examines major IoT attack categories, including denial-of-service, scanning, unauthorized access, and abnormal communication behavior. Recent research indicates that machine learning and deep learning are increasingly being investigated for IoT intrusion detection, while computational complexity, high false-positive rates, dataset representativeness, and deployment constraints remain important challenges. (DOI) The proposed framework provides a practical conceptual foundation for developing adaptive and lightweight intrusion detection systems capable of operating closer to IoT devices.



Keywords: Internet of Things, Intrusion Detection, Artificial Intelligence, Machine Learning, Edge Computing, Anomaly Detection, Cybersecurity, IoT Networks, Lightweight Security

1. Introduction

The Internet of Things has become an important component of modern digital infrastructure. Sensors, smart appliances, industrial controllers, vehicles, medical devices, cameras, meters, and embedded systems increasingly communicate with one another through wired and wireless networks.

A simplified IoT communication model can be expressed as:

IoT Device → Gateway → Network → Cloud
IoT Device → Gateway → Network → Cloud

The increasing connectivity of these devices also increases the number of possible points through which an attacker can attempt to compromise a system.

IoT devices frequently operate under constraints involving:

- Processing capability
- Memory
- Battery capacity
- Network bandwidth
- Storage
- Security functionality

These characteristics make conventional enterprise cybersecurity techniques difficult to apply directly to every IoT device.

Intrusion Detection Systems (IDS) are therefore an important component of IoT security. An IDS observes system or network behavior and attempts to identify activities that differ from legitimate behavior.

An intrusion detection decision can be represented as:

$$D(x) = \begin{cases} \text{Normal}, & x \in C_N \\ \text{Attack}, & x \in C_A \end{cases}$$

where x represents an observed network event, C_N represents normal behavior, and C_A represents malicious behavior.

Machine learning provides a mechanism for learning patterns from historical data rather than relying exclusively on manually defined attack signatures. Recent literature identifies supervised, unsupervised, ensemble, deep-learning, federated-learning, and hybrid approaches as important directions in IoT security research. ([PubMed Central \(PMC\)](#))

However, high model complexity can become problematic when security mechanisms must operate near resource-constrained devices.

This research therefore focuses on an AI-driven intrusion detection architecture that emphasizes **lightweight analysis, edge deployment, feature reduction, and adaptive detection**.

2. Research Motivation

Traditional signature-based IDS approaches search for known patterns associated with previously identified attacks.

The basic process is:

Traffic $\rightarrow$ SignatureMatching $\rightarrow$ Decision

This approach can be effective for known attacks but may have difficulty identifying previously unseen attack patterns.

Anomaly-based detection follows a different principle:

Traffic $\rightarrow$ BehaviorModel $\rightarrow$ AnomalyScore



An activity is considered suspicious when its characteristics differ substantially from learned normal behavior.

Machine learning can assist in developing such behavioral models.

The motivation for this research comes from five major challenges.

2.1 Heterogeneous Devices

IoT networks contain devices with different hardware, software, communication protocols, and operational characteristics.

2.2 Resource Constraints

Many devices cannot execute computationally intensive models continuously.

2.3 Evolving Attacks

Attackers can modify their behavior, making static security rules less effective.

2.4 False Positives

An IDS that frequently classifies legitimate traffic as malicious can generate unnecessary alerts.

2.5 Deployment Complexity

A theoretically accurate model may still be unsuitable if its computational requirements are too high for practical IoT deployment.

Recent surveys specifically highlight computational complexity and false-positive rates as continuing issues in AI-based IoT intrusion detection. ([DOI](#))

3. Problem Statement

The research problem can be formulated as follows:

How can an artificial-intelligence-based intrusion detection system identify malicious activity in heterogeneous IoT networks while maintaining low computational overhead, low detection latency, and acceptable detection accuracy?

The problem can be represented mathematically as:

$$IDS=f(X,M,R)$$

where:

- XX = observed network characteristics
- MM = trained detection model
- RR = resource constraints

The objective is to maximize detection effectiveness while minimizing computational requirements.

Therefore:

$$\text{Objective}=\text{Maximize}(\text{DetectionQuality})-\text{Minimize}(\text{ComputationalCost})$$
$$\text{Objective}=\text{Maximize}(\text{DetectionQuality}) - \text{Minimize}(\text{ComputationalCost})$$

4. Research Objectives

The main objectives of this research are:

1. To study security threats affecting IoT networks.
2. To investigate the application of artificial intelligence to intrusion detection.
3. To design a lightweight AI-based IDS architecture.
4. To reduce unnecessary feature-processing operations.
5. To utilize edge computing for local threat analysis.
6. To develop an anomaly-risk scoring mechanism.

7. To reduce false-positive alerts.
8. To evaluate the proposed approach using standard performance metrics.
9. To examine computational and communication overhead.
10. To identify future improvements for adaptive IoT security.

5. Research Contributions

The major contributions of this research include:

- A lightweight AI-driven IDS architecture for IoT.
- An edge-assisted security model.
- A feature-selection mechanism for reducing computational overhead.
- A mathematical anomaly-scoring approach.
- A risk-based classification strategy.
- A mechanism for periodically updating detection models.
- A framework for evaluating accuracy and resource consumption simultaneously.
- A security architecture suitable for heterogeneous IoT environments.

6. Background of IoT Security

IoT networks differ from conventional computer networks because they contain large numbers of devices with different capabilities.

A typical IoT environment may contain:

$$N=\{d_1,d_2,d_3,\dots,d_n\}$$

where every device d_i can have different:

- CPU capacity
- Memory
- Network behavior

- Data-generation rate
- Communication protocol

Consequently, normal traffic generated by one device may appear unusual for another device.

For example, a smart temperature sensor may communicate periodically, while a security camera may continuously transmit relatively large amounts of data.

Therefore, a universal traffic threshold may not be appropriate.

7. IoT Attack Surface

The IoT attack surface can be divided into several areas:

Device Layer

Attackers may target device firmware, credentials, or exposed interfaces.

Communication Layer

Network traffic may be intercepted, manipulated, or disrupted.

Gateway Layer

Compromising an edge gateway can provide access to multiple devices.

Application Layer

Application vulnerabilities can expose data or control functionality.

Cloud Layer

Centralized services may become targets for unauthorized access or denial-of-service attacks.



A simplified attack path is:

Attacker→Device→Gateway→Network→Application
Attacker \rightarrow Device \rightarrow Gateway \rightarrow Network \rightarrow Application

8. Common IoT Attacks

8.1 Denial-of-Service

The attacker attempts to exhaust network, processing, or service resources.

LegitimateRequests+MaliciousRequests→ResourceExhaustion
LegitimateRequests + MaliciousRequests \rightarrow ResourceExhaustion

8.2 Distributed Denial-of-Service

Multiple compromised devices coordinate traffic against a target.

$D_1 + D_2 + \dots + D_n \rightarrow \text{Target}$
 $D_1 + D_2 + \dots + D_n \rightarrow \text{Target}$

DDoS attacks are an important research area within IoT intrusion detection. Recent systematic research has examined machine-learning approaches specifically for IoT DDoS detection and has identified datasets and algorithms commonly used in this area. ([Welcome to DTU Research Database](#))

8.3 Network Scanning

An attacker probes devices or services to discover potential targets.

8.4 Unauthorized Access

Attackers attempt to obtain valid credentials or exploit weak authentication mechanisms.

8.5 Malicious Communication

A compromised device may begin communicating with destinations that are inconsistent with its normal behavior.

9. Proposed System Architecture

The proposed framework consists of five major components:

1. IoT devices.
2. Edge gateway.
3. Feature-processing module.
4. AI-based detection engine.
5. Security-management server.

The architecture is:

IoTDevices→EdgeGateway→FeatureProcessor→AI IDS→SecurityManagerIoTDevices \rightarrow EdgeGateway \rightarrow FeatureProcessor \rightarrow AI\ IDS \rightarrow SecurityManager

The edge gateway is responsible for collecting traffic information from connected devices.

The AI detection engine analyzes selected traffic features.

10. IoT Device Layer

The device layer contains sensors and embedded systems.

Each device generates network observations:

$X_i = \{x_1, x_2, \dots, x_m\}$ $X_i = \{x_1, x_2, \dots, x_m\}$

Examples of observable features include:

- Packet count
- Packet size
- Connection duration
- Destination frequency
- Protocol
- Source port
- Destination port
- Transmission interval
- Connection failure count

The framework does not require raw application data to be stored for every security decision.

Instead, selected network-level characteristics are analyzed.

11. Edge Gateway

The edge gateway acts as an intermediary between IoT devices and the analytical infrastructure.

Its responsibilities include:

- Traffic collection
- Feature extraction
- Data filtering
- Preliminary anomaly detection
- Communication with the central security service
- Model distribution

The gateway architecture can be expressed as:

Device→Gateway→LocalDetectionDevice → Gateway → LocalDetection

Only selected information may be forwarded to the central system.

This can reduce bandwidth consumption.

12. Feature Extraction

Raw network traffic contains information that may not all be useful for classification.

Let:

$$F = \{f_1, f_2, \dots, f_n\} \quad F = \{f_1, f_2, \dots, f_n\}$$

represent the complete feature set.

A reduced feature set is:

$$F' \subset F \quad F' \subset F$$

with:

$$|F'| < |F| \quad |F'| < |F|$$

The objective is to preserve the most informative characteristics while reducing computational requirements.

13. Feature Selection

Feature selection can be based on:

- Statistical relevance
- Correlation
- Information gain
- Model importance
- Variance

- Redundancy

Suppose the original dataset contains 100 features.

If only 25 provide significant predictive information:

$$F'=25$$

instead of:

$$F=100$$

The detection engine may require substantially less processing.

Feature selection is particularly important in resource-constrained environments because model complexity directly affects execution time and memory consumption.

14. Data Preprocessing

Before model training, network observations should be cleaned.

Common preprocessing operations include:

1. Removing incomplete records.
2. Handling missing values.
3. Removing duplicate observations.
4. Normalizing numerical attributes.
5. Encoding categorical attributes.
6. Balancing highly uneven classes.

A normalized feature can be represented as:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

where:

$$0 \leq x' \leq 10 \leq x' \leq 1$$

15. Handling Class Imbalance

IoT datasets can contain significantly more normal traffic than malicious traffic.

Suppose:

$$N_{\text{normal}} = 95000, N_{\text{normal}} = 95000$$

and:

$$N_{\text{attack}} = 5000, N_{\text{attack}} = 5000$$

Then:

$$\text{Ratio} = 19:1, \text{Ratio} = 19:1$$

A model trained directly on such data may become biased toward the majority class.

Possible techniques include:

- Oversampling
- Undersampling
- Synthetic data generation
- Class weighting
- Balanced ensemble learning

The choice should depend on the dataset and deployment objective.

16. Machine Learning Model

The proposed framework supports lightweight classification models such as:

- Decision Tree
- Random Forest
- Logistic Regression
- Support Vector Machine
- Naive Bayes
- Lightweight neural networks

Ensemble approaches have also received substantial attention in IoT intrusion detection research. Recent surveys identify Random Forest and related ensemble methods among commonly studied approaches. ([DOI](#))

For edge deployment, the preferred model should not be selected solely according to accuracy.

A better objective is:

ModelQuality=Accuracy+Efficiency+Stability+Interpretability
ModelQuality= Accuracy+ Efficiency+ Stability+ Interpretability

17. Proposed Two-Stage Detection

The framework introduces two detection stages.

Stage 1: Lightweight Screening

The gateway performs rapid classification.

Traffic→LightweightModelTraffic\rightarrow LightweightModel

Possible output:

NormalNormal

or:

SuspiciousSuspicious

Stage 2: Detailed Analysis

Suspicious traffic is forwarded for deeper analysis.

Suspicious $\rightarrow$ AdvancedModel $\rightarrow$ FinalDecision

This reduces the amount of traffic requiring computationally expensive analysis.

18. Anomaly Score

For each traffic observation x , the system calculates an anomaly score:

$$A(x) \in [0, 1]$$

where:

- 00 indicates highly normal behavior.
- 11 indicates highly suspicious behavior.

A simple weighted formulation is:

$$A(x) = w_1 a_1 + w_2 a_2 + \dots + w_n a_n$$

where a_i represents an individual anomaly indicator.

The weights satisfy:

$$\sum_{i=1}^n w_i = 1$$

19. Risk Classification

The anomaly score can be converted into a risk level.

$$\text{Risk}(x) = \begin{cases} \text{Low}, & A(x) < \tau_1 \\ \text{Medium}, & \tau_1 \leq A(x) < \tau_2 \\ \text{High}, & A(x) \geq \tau_2 \end{cases}$$

This allows the system to respond differently to different levels of suspicion.

For example:

Low → Monitor Low → Monitor Medium → Additional Analysis Medium → Additional Analysis High → Alert/Block High → Alert/Block

20. Adaptive Threshold

A fixed threshold may not work equally well for every IoT device.

The proposed framework therefore allows device-specific thresholds.

For device i :

$$\tau_i = f(B_i) \quad \tau_i = f(B_i)$$

where B_i represents historical behavioral characteristics.

If a device normally generates frequent connections, the detection system should not automatically classify frequent connections as malicious.

21. Device Behavioral Profile

Each device can maintain a behavioral profile:

$$P_i = \{F_i, T_i, C_i, D_i\} \quad P_i = \{F_i, T_i, C_i, D_i\}$$

where:

- F_i = normal communication frequency
- T_i = timing characteristics
- C_i = common communication destinations
- D_i = typical data-volume characteristics

A new observation is compared against this profile.

22. Behavioral Deviation

Let:

B_i

represent the expected behavior of device i .

The observed behavior is:

O_i

The deviation can be defined as:

$$\Delta_i = \text{Distance}(O_i, B_i) \quad \Delta_i = \text{Distance}(O_i, B_i)$$

If:

$$\Delta_i > \tau \quad \Delta_i > \tau$$

the activity may be classified as suspicious.

23. Model Training

The training process consists of:

1. Dataset collection.
2. Data cleaning.
3. Feature selection.
4. Feature transformation.
5. Training-test separation.
6. Model training.
7. Hyperparameter optimization.
8. Validation.
9. Performance evaluation.

A typical division may be:

Dataset=Training+Validation+Testing

The testing data should remain separate from training data.

24. Model Validation

Cross-validation can be used during development.

For k-fold validation:

Dataset= $D_1 \cup D_2 \cup \dots \cup D_k$

Each subset is used as a validation set while the remaining subsets are used for training.

The final score can be calculated as:

Score= $\frac{1}{k} \sum_{i=1}^k \text{Score}_i$

This provides a more stable estimate than relying on a single split.

25. Detection Metrics

Accuracy is calculated as:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

where:

- TPTP = true positives
- TNTN = true negatives
- FPPF = false positives
- FNFN = false negatives

Precision is:

$$\text{Precision} = \frac{TP}{TP+FP}$$

Recall is:

$$\text{Recall} = \frac{TP}{TP+FN}$$

The F1-score is:

$$F1 = 2 \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

26. Why Accuracy Alone Is Insufficient

Suppose an IoT dataset contains:

99009900

normal records and:

100100

attack records.

A classifier that labels every observation as normal achieves:

Accuracy=99%Accuracy=99\%

but:

Recall=0Recall=0

for the attack class.

Therefore, security evaluation must consider multiple metrics.

The proposed framework emphasizes:

- Precision
- Recall
- F1-score
- False-positive rate
- Detection latency
- Resource consumption

27. False Positive Rate

The false-positive rate is:

$$FPR = \frac{FP}{FP+TN}$$

A high FPR can overwhelm security administrators with unnecessary alerts.

Therefore, reducing FPR is an important objective.

Recent literature on IoT IDS continues to identify false positives as a significant challenge, particularly for anomaly-based approaches. ([DOI](#))

28. Detection Latency

Detection latency is the time between observing malicious behavior and generating a security decision.

$$L_d = T_{\text{decision}} - T_{\text{observation}} = T_{\{\text{decision}\}} - T_{\{\text{observation}\}}$$

A lower value is desirable for real-time IoT security.

Edge processing can reduce network round-trip time because some decisions occur close to the source.

29. Computational Overhead

The computational cost can be estimated using:

$$C = C_{\text{preprocess}} + C_{\text{feature}} + C_{\text{model}} = C_{\{\text{preprocess}\}} + C_{\{\text{feature}\}} + C_{\{\text{model}\}}$$

A lightweight architecture aims to minimize:

CC

without substantially reducing detection quality.

30. Memory Consumption

For an edge gateway, memory consumption is another important factor.

Let:

$$M_{total} = M_{model} + M_{runtime} + M_{buffer} \quad M_{total} = M_{model} + M_{runtime} + M_{buffer}$$

The model should satisfy:

$$M_{total} \leq M_{available} \quad M_{total} \leq M_{available}$$

This requirement can eliminate otherwise accurate models that are impractical for edge deployment.

31. Communication Overhead

A centralized IDS may forward large quantities of network data to a cloud server.

The proposed architecture performs preliminary processing locally.

Thus:

RawTraffic → EdgeProcessing → SelectedMetadata
RawTraffic → EdgeProcessing → SelectedMetadata

instead of:

RawTraffic → CloudRawTraffic → Cloud

This can reduce communication overhead.

32. Security Response

Once an intrusion is detected, the system can initiate different responses.

Possible responses include:

- Alert generation
- Traffic blocking
- Device isolation
- Rate limiting
- Additional authentication
- Gateway notification
- Administrator notification

A risk-based response is preferable to automatically blocking every suspicious event.

33. Device Isolation

When a device is classified as highly suspicious:

$$\text{Risk}(\text{Device}_i) \geq \tau \text{Risk}(\text{Device}_i) \geq \tau$$

the gateway can move it into a restricted network segment.

Conceptually:

$$\text{NormalNetwork} \rightarrow \text{QuarantineNetwork}$$

This prevents potentially compromised devices from communicating freely with other systems.

34. Continuous Learning

IoT traffic patterns may change over time.

Therefore, a model trained once may eventually become less effective.

The proposed framework supports periodic model updates.

$M_{t+1} = \text{Update}(M_t, D_{\text{new}})$ $M_{\{t+1\}} = \text{Update}(M_{\{t\}}, D_{\{\text{new}\}})$

where D_{new} $D_{\{\text{new}\}}$ represents newly collected training information.

However, model updates should be carefully controlled to avoid learning malicious traffic as normal behavior.

35. Model Poisoning Risk

An attacker may deliberately inject malicious observations into training data.

This can be represented as:

$D_{\text{train}}' = D_{\text{train}} + D_{\text{poison}}$ $D_{\{\text{train}\}}' = D_{\{\text{train}\}} + D_{\{\text{poison}\}}$

A poisoned model may produce incorrect predictions.

Therefore, training data should undergo validation before being incorporated into future model versions.

36. Explainable Intrusion Detection

Security administrators may need to understand why an alert was generated.

A model can provide feature contributions:

Explanation= $\{f_1:c_1, f_2:c_2, \dots, f_n:c_n\}$

For example:

- Unusual destination
- Abnormally high packet rate
- Unexpected port
- Unusual connection duration

This can improve analyst confidence and assist in investigating incidents.

Explainability is increasingly relevant because advanced AI models can be difficult to interpret.

37. Privacy Considerations

IoT network traffic can contain sensitive information.

The proposed architecture therefore minimizes unnecessary transmission of raw information.

Instead:

RawTraffic $\rightarrow$ LocalFeatureExtraction $\rightarrow$ SecurityMetadata
RawTraffic $\rightarrow$ SecurityMetadata

Only relevant information is transmitted to higher-level systems.

38. Federated Learning Extension

A future version of the proposed architecture can incorporate federated learning.

Multiple gateways train local models:

$M_1, M_2, \dots, M_n$

Model updates can then be aggregated:

$M_{global} = \text{Aggregate}(M_1, M_2, \dots, M_n)$

The advantage is that raw network traffic does not necessarily need to leave each organization.

Federated learning is already identified in recent literature as a promising approach for improving privacy and scalability in IoT intrusion detection. ([Springer](#))

39. Proposed Algorithm

Algorithm: Lightweight AI-Based IoT Intrusion Detection

Input: Network traffic XX

Output: Security classification YY

1. Receive network observation XX.
2. Identify the originating IoT device.
3. Extract relevant network features.
4. Normalize numerical features.
5. Remove irrelevant attributes.
6. Generate the device behavioral profile.
7. Apply lightweight screening model.
8. Calculate anomaly score.
9. If anomaly score is below threshold, classify as normal.
10. If anomaly score is above the first threshold, perform additional analysis.
11. If risk exceeds the critical threshold, generate an alert.
12. Optionally isolate the suspicious device.
13. Store security event information.
14. Periodically validate and update the detection model.

40. Proposed Mathematical Model

For device i , define the observation vector:

$$X_i = [x_1, x_2, \dots, x_n]$$

The detection model generates:

$$P_i = P(Y = \text{Attack} | X_i)$$

The behavioral anomaly score is:

$$A_i$$

The combined risk score can be expressed as:

$$R_i = \alpha P_i + \beta A_i + \gamma H_i$$

where:

- P_i = machine-learning attack probability
- A_i = behavioral anomaly score
- H_i = historical risk indicator
- α, β, γ = weighting parameters

with:

$$\alpha + \beta + \gamma = 1$$

41. Risk-Based Decision

The final decision can be defined as:

$Decision_i = \begin{cases} \text{Normal}, & R_i < \theta_1 \\ \text{Monitor}, & \theta_1 \leq R_i < \theta_2 \\ \text{Investigate}, & \theta_2 \leq R_i < \theta_3 \\ \text{Block/Isolate}, & R_i \geq \theta_3 \end{cases}$

This provides more flexibility than a simple binary classifier.

42. Experimental Design

A prototype implementation can be evaluated using publicly available IoT cybersecurity datasets.

Recent research commonly evaluates IoT IDS approaches using datasets such as BoT-IoT and TON_IoT, among others. ([Welcome to DTU Research Database](#))

The experimental procedure should include:

1. Dataset preparation.
2. Feature analysis.
3. Feature reduction.
4. Model training.
5. Model validation.
6. Model testing.
7. Edge deployment simulation.
8. Performance measurement.
9. Security analysis.

43. Baseline Models

The proposed approach can be compared against:



Model A

Decision Tree

Model B

Random Forest

Model C

Support Vector Machine

Model D

Lightweight Neural Network

Model E

Proposed Hybrid Detection Model

The comparison should not be limited to accuracy.

44. Comparative Evaluation Parameters

Parameter	Measurement
Accuracy	Classification correctness
Precision	Correctness of attack alerts
Recall	Attack detection capability
F1-score	Combined precision and recall
FPR	False alarm rate
Latency	Detection delay



Parameter	Measurement
Memory	Model/resource consumption
CPU usage	Processing requirement
Communication	Data transferred
Scalability	Performance as devices increase

45. Scalability Analysis

Suppose the number of devices increases from:

$$N=100$$

to:

$$N=10000$$

The detection system must continue operating without excessive degradation.

Processing requirements can be represented as:

$$C(N)=C_0+kN$$

where C_0 is fixed processing cost and k represents per-device processing cost.

A scalable architecture should keep k sufficiently small.

46. Edge Versus Cloud Detection

A centralized cloud architecture is:

IoT → Cloud → Decision

The proposed architecture is:

IoT→Edge→DecisionIoT \rightarrow Edge \rightarrow Decision

with optional cloud coordination.

Edge processing can potentially reduce latency and bandwidth requirements.

Cloud infrastructure remains useful for:

- Global model training
- Historical analysis
- Long-term storage
- Cross-network intelligence

Therefore, a hybrid architecture is appropriate.

47. Hybrid Edge-Cloud Architecture

The proposed deployment model is:

IoT→Edge→CloudIoT \rightarrow Edge \rightarrow Cloud

The edge performs:

FastDetectionFastDetection

while the cloud performs:

DeepAnalysis+ModelTraining+LongTermAnalyticsDeepAnalysis+ ModelTraining+ LongTermAnalytics

This division balances responsiveness and computational capacity.

48. Security Event Management

Detected events can be recorded as:

$E = (\text{DeviceID}, \text{Timestamp}, \text{Type}, \text{Risk}, \text{Action})$

For example:

$E_i = (D_5, T_1, \text{DDoS}, 0.91, \text{Isolate})$

Such records can assist security administrators in understanding attack patterns.

49. Alert Prioritization

Not every alert requires the same level of attention.

The system can calculate:

$\text{Priority} = \text{Risk} \times \text{AssetImportance}$

A suspicious event affecting a critical industrial controller should receive higher priority than an event affecting a low-impact monitoring device.

50. Critical Device Protection

IoT networks can contain devices with different levels of importance.

Let:

$l_i \in [0, 1]$

represent the importance of device ii.

Then:

$$Priority_i = R_i \times I_i \quad Priority_i = R_i \times I_i$$

This provides context-aware alert prioritization.

51. Model Robustness

A detection system must remain reliable when traffic conditions change.

Testing should therefore include:

- Normal traffic variation
- Increased device count
- New communication patterns
- Unknown attack types
- Noisy traffic
- Missing features

A robust model should maintain acceptable performance across these conditions.

52. Unknown Attack Detection

Supervised classifiers generally depend on examples of known attack classes.

However:

$Attack_{new} \notin TrainingSet$ $Attack_{\{new\}} \notin TrainingSet$

may still occur.

Anomaly detection can help identify deviations that do not exactly match known classes.

Therefore, combining:

SupervisedDetection+AnomalyDetectionSupervisedDetection+ AnomalyDetection

can provide broader coverage.

53. Hybrid Detection

The proposed system combines classification and anomaly detection.

$X \rightarrow \text{Classifier} \rightarrow \text{Classifier}$

and:

$X \rightarrow \text{AnomalyDetector} \rightarrow \text{AnomalyDetector}$

Their outputs can be combined:

$$R = w_c C + w_a A$$

where:

- CC = classifier score
- AA = anomaly score

This hybrid design can improve flexibility.

54. Resource-Aware Model Selection

Model selection should consider four dimensions:

Selection=Accuracy+Latency+Memory+CPUSelection= Accuracy+ Latency+ Memory+ CPU

A model with slightly lower accuracy may be preferable if it significantly reduces computational requirements.

This is especially relevant for edge-based deployment.

55. Energy Efficiency

Battery-powered IoT devices have limited energy.

Energy consumption can be represented as:

$$E = E_{\text{sensing}} + E_{\text{communication}} + E_{\text{processing}}$$

Security mechanisms should minimize additional energy consumption.

Moving intensive analysis from the device to an edge gateway can reduce device-side computational requirements.

56. Communication Optimization

Instead of transmitting every network observation:

$$X_1, X_2, \dots, X_n$$

the gateway can transmit only suspicious or aggregated events:

$$E_1, E_2, \dots, E_k$$

where:



k<<nk\\l n

This can substantially reduce security-related communication.

57. Model Update Security

A model-update mechanism should verify that new models originate from trusted sources.

A model package can be associated with:

$\text{ModelHash} = \text{Hash}(M)$

The gateway verifies the received hash before installing the new model.

58. Model Version Management

Each model can have a version:

M_v

For example:

$M_1 \rightarrow M_2 \rightarrow M_3$

If a new model performs poorly, the system can revert to a previously validated version.

59. Failure Handling

If the AI detection engine becomes unavailable, the gateway should maintain basic security controls.

A fallback mechanism may include:

- Rate limiting
- Known malicious IP filtering
- Authentication enforcement
- Device isolation
- Basic rule-based detection

Thus:

AI Failure $\Rightarrow$ Security Failure
AI \ Failure \not\Rightarrow Security Failure

60. Security Analysis

The proposed framework addresses multiple threats.

Unauthorized Access

Device identity and network behavior are analyzed.

DDoS

Abnormally high traffic rates can increase anomaly scores.

Scanning

Unexpected connection attempts can be detected.

Compromised Devices

Behavioral deviation can identify unusual activity.

Data Manipulation

Security events can be preserved for later investigation.

61. Advantages of the Proposed Framework

The main advantages include:

1. Lightweight edge-oriented architecture.
2. Reduced cloud dependency.
3. Adaptive behavioral analysis.
4. Multiple detection stages.
5. Reduced communication overhead.
6. Support for heterogeneous devices.
7. Risk-based response.
8. Possibility of continuous model improvement.
9. Support for explainable decisions.
10. Compatibility with future federated-learning extensions.

62. Limitations

The proposed framework also has limitations.

Dataset Dependency

Machine-learning performance depends strongly on training data quality.

Concept Drift

IoT traffic behavior can change over time.

Adversarial Attacks

Attackers may attempt to manipulate AI models or their input data.

Resource Constraints

Even lightweight models may be difficult to execute on extremely constrained devices.

Privacy

Network observations may contain sensitive information.

Model Maintenance

Continuous model updates require secure management.

63. Challenges

Several research challenges remain.

63.1 Representative Datasets

Synthetic or outdated datasets may not accurately represent current IoT traffic.

63.2 Real-World Deployment

A model performing well in a laboratory environment may behave differently in operational networks.

63.3 False Alarms

Excessive alerts can reduce the practical usefulness of an IDS.

63.4 Explainability

Complex AI models may make security decisions difficult to interpret.

63.5 Adversarial Machine Learning

Attackers may deliberately manipulate inputs or training data.

Recent research has specifically examined adversarial machine learning as a security concern for AI-driven IoT intrusion detection. ([DOI](#))

64. Future Scope

Future research can extend the proposed architecture in several directions.

64.1 Federated Learning

Federated learning can allow organizations to collaboratively improve IDS models without directly sharing raw network traffic.

64.2 TinyML

TinyML techniques may enable selected security models to operate directly on highly constrained devices.

64.3 Explainable AI

Explainable AI can improve analyst understanding of detection decisions.

64.4 Transformer Models

Transformer-based models may provide improved representation of complex sequential network behavior, although their computational requirements must be carefully evaluated.

64.5 Generative AI

Generative methods may assist with synthetic attack generation, security analysis, and automated investigation, but they also introduce new security considerations.

Recent IoT IDS literature identifies TinyML, federated learning, explainable AI, transformers, and other emerging approaches as active areas of investigation. ([MDPI](#))

65. Discussion

The main argument of this research is that IoT intrusion detection should not be designed exclusively around classification accuracy.

A practical system must simultaneously consider:

AccuracyAccuracy LatencyLatency MemoryMemory CPU CPU CommunicationCommunication

and:

AdaptabilityAdaptability

A highly accurate model that requires excessive computational resources may be unsuitable for real-world IoT deployment.

Similarly, an extremely lightweight model with poor recall may fail to identify dangerous attacks.

The proposed two-stage architecture provides a compromise.

Most traffic can be screened using a lightweight model:

Traffic $\rightarrow$ FastScreeningTraffic $\rightarrow$ FastScreening

Only suspicious traffic proceeds to deeper analysis:

SuspiciousTraffic $\rightarrow$ DetailedAnalysisSuspiciousTraffic $\rightarrow$ DetailedAnalysis

This reduces the computational burden while retaining the possibility of more sophisticated analysis.

The use of device-specific behavioral profiles further recognizes that IoT devices do not all behave identically.

A smart meter, camera, industrial controller, and environmental sensor may have completely different normal communication patterns.

Therefore:

$\text{Normal}(\text{Device}_A) \neq \text{Normal}(\text{Device}_B)$ $\text{Normal}(\text{Device}_A) \neq \text{Normal}(\text{Device}_B)$

A detection system should incorporate this heterogeneity.

66. Conclusion

This paper presented an AI-driven intrusion detection framework designed for resource-constrained Internet of Things networks. The proposed architecture combines IoT devices, edge gateways, feature selection, lightweight machine learning, anomaly analysis, risk scoring, and centralized security management.

The framework addresses several important challenges associated with IoT cybersecurity. Instead of sending all network information to a centralized cloud system, the architecture performs preliminary processing near the source. This reduces communication overhead and can improve detection responsiveness.

A two-stage detection mechanism was proposed. The first stage performs lightweight screening, while suspicious observations are forwarded for more detailed analysis. An anomaly-risk model combines machine-learning predictions with behavioral deviation and historical information.

The research also emphasized that accuracy alone is insufficient for evaluating IoT intrusion detection. Precision, recall, F1-score, false-positive rate, detection latency, memory consumption, CPU usage, and communication overhead should all be considered.

The proposed mathematical model provides a basis for risk-based security decisions:

$$R_i = \alpha P_i + \beta A_i + \gamma H_i$$

where classification probability, behavioral anomaly, and historical risk contribute to the final security decision.

The framework can be further enhanced through federated learning, TinyML, explainable AI, adversarially robust learning, and adaptive model updating. Recent research confirms that lightweight deployment, federated learning, false-positive reduction, computational efficiency, and representative datasets remain important challenges in IoT intrusion detection. ([DOI](#))

Future experimental work should implement the framework using real or representative IoT traffic and compare multiple machine-learning algorithms under identical conditions. Such evaluation would provide quantitative evidence regarding the trade-off between detection performance and resource consumption.

Overall, an effective IoT intrusion detection system should not simply ask whether network traffic is malicious. It should consider **which device generated the traffic, how unusual the behavior is, how important the device is, how confident the AI model is, and what security response is appropriate**. This context-aware approach can provide a stronger foundation for scalable and practical IoT cybersecurity.

References

1. Atzori, L., Iera, A., & Morabito, G. "The Internet of Things: A Survey." *Computer Networks*, 54(15), 2787–2805, 2010.
2. Sicari, S., Rizzardi, A., Grieco, L. A., & Coen-Porisini, A. "Security, Privacy and Trust in Internet of Things: The Road Ahead." *Computer Networks*, 76, 146–164, 2015.

3. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications." *IEEE Communications Surveys & Tutorials*, 17(4), 2347–2376, 2015.
4. Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. "Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions." *Future Generation Computer Systems*, 29(7), 1645–1660, 2013.
5. Makhdoom, I., Abolhasan, M., Lipman, J., Liu, R. P., & Ni, W. "Anatomy of Threats to the Internet of Things." *IEEE Communications Surveys & Tutorials*, 21(2), 1636–1675, 2019.
6. Reyna, A., Martín, C., Chen, J., Soler, E., & Díaz, M. "On Blockchain and Its Integration with IoT: Challenges and Opportunities." *Future Generation Computer Systems*, 88, 173–190, 2018.
7. Christidis, K., & Devetsikiotis, M. "Blockchains and Smart Contracts for the Internet of Things." *IEEE Access*, 4, 2292–2303, 2016.
8. Kikissagbe, B. R., & Adda, M. "Machine Learning-Based Intrusion Detection Methods in IoT Systems: A Comprehensive Review." *Electronics*, 13(18), 3601, 2024. ([MDPI](#))
9. Mallidi, S. K. R., & Ramisetty, R. R. "Advancements in Training and Deployment Strategies for AI-Based Intrusion Detection Systems in IoT: A Systematic Literature Review." *Discover Internet of Things*, 5, 8, 2025. ([Springer](#))
10. Bankó, M. B., Dyszewski, S., Králová, M., Limpek, M. B., Papaioannou, M., Choudhary, G., & Dragoni, N. "Advancements in Machine Learning-Based Intrusion Detection in IoT: Research Trends and Challenges." *Algorithms*, 18(4), 209, 2025. ([Welcome to DTU Research Database](#))
11. "A Survey on Intrusion Detection System in IoT Networks." *Cyber Security and Applications*, 3, 100082, 2025. ([DOI](#))
12. "Machine Learning-Based Security Solutions for IoT Networks: A Comprehensive Survey." *Sensors*, 25(11), 3341, 2025. ([PubMed Central \(PMC\)](#))
13. "Deep Learning for Cyber Threat Detection in IoT Networks: A Review." *Internet of Things and Cyber-Physical Systems*, 2023. ([ScienceDirect](#))
14. "Review of Artificial Intelligence for Enhancing Intrusion Detection in the Internet of Things." *Engineering Applications of Artificial Intelligence*, 127, 107231, 2024. ([ScienceDirect](#))
15. "Intrusion Detection in the Internet of Things: A Comprehensive Review of Techniques, Architectures, Datasets, and Emerging Trends." *Sensors*, 26(11), 3405, 2026. ([MDPI](#))
- ...