SSR Journal of Artificial Intelligence (SSRJAI)

OPEN ACCESS

Homepage: https://ssrpublisher.com/ssrjai/

Volume 3, Issue 8, 2026 ISSN: 3049-0413

A Lightweight AI-Driven Intrusion Detection System for Cyber-Physical IoT Networks: Balancing Detection Performance and Edge Deployability

Peter Wonah Odey, Hashim Abdul Isah* and Okoduwa Ernest Atama

Centre for Cyberspace Studies, Nasarawa State University P.M.B. 1022, Keffi, Nasarawa State, Nigeria

Received: *05.07.2026* | ***Accepted:*** *25.07.2026* | ***Published:*** *01.08.2026*

***Corresponding author:** Hashim Abdul Isah

DOI: *10.5281/zenodo.21737248*

Abstract **Original Research**

Machine learning-based intrusion detection systems for IoT networks are predominantly evaluated on detection metrics alone; computational efficiency is left unmeasured and deployment feasibility unaddressed. This study presents a dual-axis evaluation of five lightweight ML models on the TON_IoT network traffic dataset under a bias-aware preprocessing protocol that removes testbed-specific identifiers before feature selection. The preprocessing protocol is designed to produce feature sets that generalise beyond the training topology. A hybrid mutual information and recursive feature elimination pipeline reduced 60 candidate features to 18 behavioural attributes. Five models were trained and evaluated on the held-out test partition under CPU-only, single-core inference conditions (Random Forest, LightGBM, XGBoost, a Pruned MLP, and a 1D-CNN); each was assessed on four detection metrics (accuracy, macro-F1, false positive rate, AUC-ROC) and three efficiency dimensions (per-sample latency, serialised model size, peak RAM at inference). Tree-based models achieved macro-F1 scores of 0.923–0.936 and false positive rates of 0.003. Neural network models recorded macro-F1 scores of 0.599–0.637; the margin below the tree models (0.287–0.337 points) is attributable to calibration effects under class imbalance rather than to architectural limitations. Random Forest recorded the lowest latency (0.017 ms per sample) and is recommended for gateway-class deployment. The Pruned MLP, at 218.5 KB and 0.070 ms latency, is the only model satisfying all MCU-class constraints and is recommended for severely resource-constrained endpoints. When deployment tier thresholds are applied, four models are assigned to gateway class and one to MCU class. Detection performance and computational efficiency impose different model rankings; deployment decisions require both axes of evaluation.

Keywords: Intrusion detection system, Internet of Things, Edge deployment, Lightweight machine learning, TON_IoT, Feature selection, Computational efficiency.

1. Introduction

The convergence of operational technology and internet-connected sensing has expanded cyber-physical IoT networks into critical infrastructure, including industrial control systems, smart grids, healthcare monitoring, and autonomous transport, all of which depend on continuous, low-latency communication between constrained edge devices (Kayan, Nunes, Rana, Burnap and Perera, 2022). Attack surface growth

has kept pace with deployment scale. Zhukabayeva, Zholshiyeva, Karabayev, Khan and Alnazzawi (2025) report a sustained increase in network-layer intrusion attempts targeting IoT endpoints across 2020–2024; denial-of-service, injection, and man-in-the-middle attacks account for the majority of confirmed incidents. Conventional perimeter-based security cannot protect architectures in which traffic originates at resource-constrained nodes operating outside any managed boundary; detection must therefore occur at or near the edge, under strict computational limits.

Machine learning-based intrusion detection systems (ML-IDS) have been proposed as a principled response to this requirement, and the published literature records high accuracy figures across several model families (Sallam, El Barachi and Li, 2026). However, three gaps in the literature limit the translational value of existing proposals. First, no study reports the full efficiency profile (inference latency, serialised model size, and peak RAM at inference) for multiple model families evaluated simultaneously on the same dataset under CPU-constrained conditions; Sallam et al. (2026) find that 29 of 32 recent IoT-IDS proposals report detection metrics only, and the deployment cost of each proposal is therefore unmeasured. Second, the evidence on which model family performs best is contradictory: Lundqvist, Hadzic, Kirkeluten, Pedersen, Holth, Johansson and Halkjelsvik (2025) find Random Forest competitive with a compressed feedforward network on TON_IoT, while the DL-NIDS literature (in which CNN and LSTM architectures dominate reviewed proposals, Chinnasamy, Subramanian, Veerappampalayam Easwaramoorthy and Cho 2025) has not been compared against tree ensembles under controlled, identical conditions on the same dataset; no study tests all three families under identical hardware constraints, so the contradiction cannot be resolved from existing evidence. Third, cross-study comparisons are invalidated by inconsistent preprocessing: several proposals retain testbed-specific IP-address and port-number features that introduce source-routing bias and prevent generalisation beyond the training topology (Moustafa, Garg, Sitnikova, Jones and Sioutis, 2021).

This study addresses all three gaps through a unified experimental protocol applied to the TON_IoT network traffic dataset (Booij, Chiscop, Meeuwissen, Moustafa and den Hartog, 2022). Three research questions guide the investigation:

1. What are the detection performance profiles (accuracy, macro-F1, false positive rate, and AUC-ROC) of Random Forest, LightGBM, XGBoost, a Pruned MLP, and a 1D-CNN trained on bias-aware, hybrid-selected TON_IoT features?
2. What are the computational efficiency profiles (inference latency, serialised model size, and peak RAM) of the five candidate models under a CPU-only, single-core inference environment?
3. To which deployment tier (MCU-class or gateway-class) does each model's efficiency profile assign it, and how does that assignment interact with its detection performance?

The primary contribution is the only published dual-axis evaluation of five ML model families on TON_IoT that jointly measures detection performance and a three-dimensional efficiency profile (inference latency, serialised model size, and peak RAM). The evaluation rests on a bias-aware preprocessing protocol that removes testbed-specific identifiers before feature selection; cross-study comparisons thereby generalise beyond the training topology. Feature dimensionality is reduced from 44 raw flow attributes to 18 behavioural features through a hybrid mutual information-recursive feature elimination pipeline without a fixed K; the cross-validated F1 ceiling is recorded at each elimination step. The deployment thresholds of Sallam et al. (2026) are then applied as a concrete assignment matrix that maps each model to an MCU-class or gateway-class hardware tier on measured efficiency data.

Section 2 reviews the literature on ML-based IDS for IoT and cyber-physical networks and formalises the three gaps. Section 3 describes the dataset, preprocessing pipeline, feature selection procedure, candidate models, and evaluation framework. Section 4 reports detection and

efficiency results for all five models. Section 5 interprets the accuracy-efficiency trade-off and the deployment decision matrix. Section 6 states conclusions and directions for future work.

2. Related Work

2.1. ML-Based Intrusion Detection Systems

Machine learning has displaced signature-based detection as the dominant approach in network intrusion detection, largely because it can identify attack patterns that have not been explicitly enumerated in rule sets. A comprehensive survey of ML-based IDS covering 2013 to 2025 identified 87.5% of the 32 reviewed proposals as AI-based (Sallam et al., 2026), and a structured review of 96 AI-for-cybersecurity publications from 2023 to 2026 confirmed that classical classifiers and deep learning architectures now constitute the methodological mainstream (Xue, Xue, Wang and Ma, 2026). The two independent reviews converge on the same finding: within a decade, ML has become the default detection method; adoption is now broad enough that the research question has shifted from whether to use ML to which model, trained how, and deployed where.

Among classical machine learning classifiers, Random Forest and gradient boosting variants have received the most sustained evaluation on network traffic data. A systematic review by Shyaa, Ibrahim, Zainol, Abdullah, Anbar and Alzubaidi (2024) of 71 concept-drift-aware IDS proposals from 2019 to 2024 identified ensemble methods as the dominant classifier family; however, because the review focuses on distributional shift rather than deployment cost, none of the 71 proposals is evaluated for inference latency or memory footprint. An empirical benchmarking study by Fosic, Zagar, Grgic and Krizanovic (2023) compared seven supervised classifiers, including RF, SVM, and KNN, on UNSW-NB15 traffic reduced to eight NetFlow-compatible features; RF achieved the highest F2-score (97.68%, AUC 98.47%) but the study measured neither inference time nor model size, and the eight-feature reduction was designed for compatibility with a specific traffic monitoring format rather than for resource efficiency on constrained hardware. Support vector machines and KNN classifiers appear frequently in earlier work but have ceded ground to ensemble methods as dataset sizes have grown, since KNN inference scales poorly with training set size; Fosic et al. (2023) report KNN inference times orders of magnitude higher than RF on the same data, ruling it out for real-time deployment regardless of accuracy.

Deep learning architectures have extended detection coverage beyond what classical classifiers can capture from tabular features alone, but at a cost that the literature rarely quantifies. A PRISMA-compliant systematic review by Chinnasamy et al. (2025) of 87 DL-only NIDS papers from 2018 to 2024 found that CNN, LSTM, and autoencoder architectures dominate the literature, yet fewer than a quarter of the reviewed studies reported any measure of inference cost; the review's exclusion of studies that also employ classical ML further means that the lightweight end of the spectrum is absent from its evidence base. Autoencoder-based methods train exclusively on benign traffic and flag deviations as potential intrusions; they are therefore applicable where labelled attack data are scarce. A semi-supervised adversarial autoencoder by Shiomoto (2022) achieved performance comparable to a fully supervised MLP baseline using only 0.1% labelled samples, but the study reports no inference latency and was not evaluated on IoT-specific traffic. Neurosymbolic hybrids that pair data-driven classifiers with logic-based rule engines have been proposed to improve interpretability and robustness (Bizzarri, Yu, Jalaian, Riguzzi and Bastian, 2024); because no empirical implementation was reported in that work, the computational overhead of the approach on constrained hardware is unknown. Whereas ensemble classifiers can run inference on gateway-class hardware in microseconds, the models at the DL end of this spectrum require GPU acceleration, multi-gigabyte memory footprints, and training pipelines that cannot be reproduced on resource-constrained edge devices: the architectural distance between design and deployment is the problem the field has not resolved.

The evaluation regime the reviewed studies share is the deeper problem. Detection performance is

measured almost exclusively in accuracy and F1-score on server-class hardware under unconstrained conditions. Using a dual-stage adversarial attack framework on five DL NIDS models, Zhang, Costa-Pérez and Patras (2022) showed that decision-based black-box perturbations can reduce detection rates to near zero without violating the statistical properties of real network traffic; none of the five models had been evaluated for adversarial robustness during development; offline accuracy benchmarks therefore give no information about operational resilience. A structured review found that latency, memory consumption, and energy use were absent from the recoverable evidence layer in the majority of 96 reviewed publications (Xue et al., 2026); a complementary review of 32 IoT-IDS proposals found that 29 relied solely on offline datasets (Sallam et al., 2026). Models evaluated under unconstrained conditions on offline benchmarks carry no verifiable guarantee of operating within the memory, latency, and power budgets of the hardware on which they are intended to run, and the field has yet to establish a standard for reporting that would close this gap.

2.2. Cyber-Physical IoT Security

Industrial cyber-physical systems tightly couple digital control with physical processes, and IoT devices constitute their sensing and actuation layer. An ACM Computing Surveys review by Kayan et al. (2022) of cybersecurity across manufacturing, energy, water, transportation, and healthcare infrastructures established that IIoT is precisely this subsystem: commodity connected devices that read from and write to physical processes, inserted at the edge of control networks originally designed without internet connectivity. The convergence of IT and OT networks that enables remote monitoring and data-driven optimisation simultaneously removes the air-gap that insulated OT environments from external attack; weak boundary protection at the IT/OT interface is the most exploited vulnerability across the 16 incidents that Kayan et al. (2022) evaluated, including Stuxnet, the Ukrainian power grid attack, and the Florida water treatment compromise. A structured review by Xue et al. (2026) framed the same problem at the IoT-edge layer: the relevant threat class consists of attacks that unfold close to sensing and control processes rather than in the data centre or cloud, a characterisation that distinguishes CPS-IoT threats from the general network intrusion problem that most IDS benchmarks address.

Adversaries who breach the cyber layer of a CPS-IoT network can cause physical harm, including equipment damage, service disruption, and safety hazard, that no software patch can reverse; the asymmetry between attack cost and recovery cost is substantially larger than in IT-only environments (Kayan et al., 2022; Zhukabayeva et al., 2025). The volume and heterogeneity of IoT endpoints further produce traffic patterns that no single labelled benchmark can fully represent. A systematic review by Zhukabayeva et al. (2025) of 185 cybersecurity proposals for IIoT-edge integration across four IIoT architecture layers documented that ransomware, DDoS, malware, man-in-the-middle, and injection attacks are prevalent across all four layers, yet the reviewed proposals relied predominantly on ML models evaluated on simulated traffic in controlled testbeds rather than on operational IIoT data; that methodological choice limits the confidence with which their detection rates transfer to production deployments. Work by Vargas, Lozano-Garzon, Montoya and Donoso (2021) on KNN-based attack detection in industrial IoT illustrates the same constraint: training and testing on a VirtualBox IIoT testbed using only DoS and Fuzzer traffic from UNSW-NB15 produced accuracies of 92% to 99%, but the evaluation covered only two of nine attack types and was never validated on heterogeneous operational environments. A conceptual framework by Zeng, Yunis, Khalil and Mirza (2024) for AI-driven anomaly detection in smart city IoT, grounded in complex adaptive systems theory, identified the absence of real-environment evaluation as the primary implementation barrier; because the framework is theoretical rather than empirical, it proposes no solution to the resource-constraint problem on edge hardware.

The resource constraints of IoT edge hardware make the evaluation gap more consequential, not less. A survey of anomaly detection methods

across CPS domains by Abshari and Sridhar (2025) identified three persistent constraints on ML deployment at the IoT edge: the requirement for labelled training data that operational environments rarely produce in sufficient volume; the inability of classifiers trained on known attack signatures to detect previously unseen variants; and the computational overhead of most published architectures, which exceeds the memory and processing budgets of the gateway-class and endpoint hardware on which detection must occur. The third constraint is the least studied. Work by Ngo, Lightbody, Temko, Murphy and Popovici (2024) on FPGA-accelerated ANN and CNN anomaly detection at IIoT gateways trained and tested on the IoT-23 dataset of 325 million labelled network packets, measuring inference throughput and latency on a Xilinx Alveo U280 FPGA; the study compares FPGA against GPU and CPU implementations of the same architecture rather than against architectures of different complexity, so it cannot answer whether a simpler model would achieve comparable detection at lower hardware cost. Among the 32 IoT-IDS proposals reviewed by Sallam et al. (2026), none had been evaluated against the MCU-class deployment threshold of a model size below 500 KB, peak RAM below 256 MB, and per-sample inference latency below 1 ms that corresponds to the most constrained IoT endpoint tier. Detection proposals for cyber-physical IoT environments must therefore be evaluated not only on accuracy against known attack taxonomies but on whether they can operate within the resource envelope of the hardware they are designed to protect.

2.3. IoT-Specific Anomaly Detection

IoT-specific intrusion detection departs from general NIDS work in two material respects: the traffic to be classified is generated by heterogeneous low-power devices rather than conventional hosts, and the attack taxonomy covers threats against sensing and actuation rather than data services. A narrative survey by Chatterjee and Ahmed (2022) of 64 anomaly detection proposals across IoT application domains from 2019 to 2021 found that ML and deep learning methods dominated recent proposals, with LSTM, CNN, and autoencoder architectures most frequently applied to network-layer IoT traffic; however, as the authors note, the selection covers at most one representative paper per domain and excludes work published after July 2021, so the survey cannot characterise the shift toward transformer-based and hybrid generative architectures that followed. Statistical and geometrical methods remained competitive for online detection at the edge in the reviewed period, precisely because they impose lower computational overhead than DL approaches on the same hardware.

The dominant methodological trend in IoT-specific anomaly detection since 2022 is the combination of generative data augmentation with hybrid deep learning classifiers. Sharma, Kumar and Sharma (2025) trained a GAN-enhanced CNN-LSTM model on the CICIoT2023 smart home dataset covering 33 attack types, applying Conditional GAN-generated synthetic samples to address class imbalance and MFCC-based signal transformation to capture temporal and frequency characteristics of IoT traffic flows; the model achieved 99.4% accuracy against a CNN-LSTM baseline of 93.2%, but the study reports no inference latency, model size, or RAM footprint, so whether the architecture is deployable on any IoT gateway hardware is unknown. A split federated learning approach by Babar, Khan, Kaleem, Qureshi and Boulila (2026) partitioned a CNN-BiLSTM model across IoT devices and an edge server, evaluated on the TON_IoT network traffic benchmark, and reduced the false positive rate from 20% to 6% while reaching 99.95% binary classification accuracy by round 10; the evaluation was conducted entirely in a simulated testbed with no physical devices, and neither inference latency nor model size was reported. An InceptionTime architecture by Tareq, Elbagoury, El-Regaily and El-Horbaty (2022), evaluated on the TON_IoT network subset alongside Edge-IIoT and UNSW-NB15, achieved 99.65% multi-class accuracy using only class weights to handle imbalance; no hardware or software specification was provided, and no efficiency metric was measured.

Proposals that do quantify resource constraints represent a smaller but analytically more

instructive strand of the literature. A PureChain framework by Ibrahim, Ahakonye, Lee and Kim (2026) combining XGBoost and CNN classifiers in a closed-loop IIoT intrusion detection and prevention system reported 99.87% binary accuracy and 0.0025 s XGBoost inference latency on the IoT-CAD dataset; multi-class false positive rates of 19.78% on IoT-CAD and 34.70% on IoTForge reveal that binary accuracy alone conceals substantial classification failure at the attack-type level, and the evaluation used no physical IoT testbed. A hybrid LightGBM and autoencoder framework by Abdelhady, Ghandoura, Motwakel and Alajmi (2026) for smart grid IoT anomaly detection operated under explicit resource constraints of 512 KB memory and 15 ms latency per packet; the framework achieved 95.3% overall accuracy with a 59 KB per-device model footprint and 12 ms average processing latency, but the evaluation was simulation-based and scalability was assessed only to 500 nodes; performance under larger network sizes was not established. The reviewed IoT-specific proposals share a consistent pattern. Detection accuracy above 99% is achievable with complex DL architectures but at an efficiency cost that most proposals do not measure; where efficiency is measured, simpler models operating under explicit resource budgets produce accuracy below 99% yet remain within the deployment thresholds that constrained IoT hardware requires.

2.4. Lightweight and Edge-Deployable IDS

The subset of IoT-IDS proposals that explicitly targets resource-constrained deployment is small but provides the most directly comparable prior art to this study. Lundqvist et al. (2025) evaluated Decision Tree, LightGBM, a standard Feedforward Neural Network, and a TinyML-quantised FNN on the TON_IoT network dataset under throttled Raspberry Pi 4 conditions, testing at 4-core, 1-core at 1.5 GHz, and 1-core at 600 MHz configurations; F1-scores ranged from 0.976 to 0.987 across models; the TinyML FNN reached 31 KB model size and approximately 52,306 packets per second at 600 MHz, but RAM consumption at inference was not measured and the evaluation used CPython on a Linux host rather than bare-metal MCU firmware, so throughput figures are order-of-magnitude approximations for MCU-class deployment. Haruna, Ibrahim and Muhammed (2026) compared XGBoost, LightGBM, and Deep Forest for botnet detection on the Bot-IoT dataset under Docker-simulated constraints of one virtual CPU and 512 MB RAM; XGBoost achieved zero false positives at 0.41 ms inference latency and 2,417 packets per second, while LightGBM produced a false positive rate of 0.3333 despite 99.99% accuracy, confirming that leaf-wise training efficiency at fitting time does not produce faster inference under single-core sequential conditions. Salim, Comivi, Nurbek, Park and Park (2022) deployed an LSTM and Decision Tree combination on the Bot-IoT dataset and tested on a physical Raspberry Pi 3B+, reporting hardware-constrained inference latency; the study evaluated a single model configuration and did not report model size or RAM footprint. A lightweight IDS by Tiwari, Lakshmi, Das, Tripathy and Li (2024) applied PSO feature selection and PTQ quantisation to MARS and GAM classifiers on the WUSTL-IIoT-2021 dataset, reducing inference latency by 22% at 8-bit quantisation; the study reported total test-set inference time rather than per-sample latency and was deployed on an Azure IoT Edge simulation rather than physical hardware. These four studies share no common benchmark dataset; no study reports all three efficiency dimensions of inference latency, model size on disk, and peak RAM at inference simultaneously, and none compares more than one architecture family on the same dataset under the same hardware constraint.

2.5. Feature Selection for IoT Network Traffic

Feature dimensionality is a practical constraint in IoT-IDS as much as an accuracy concern: a model trained on 44 raw flow features from a network capture cannot claim edge deployability if the inference overhead of those features exceeds the memory budget of the target device. Li, Othman, Chen and Mi Yusuf (2024) conducted a direct comparison of feature selection and feature extraction on the TON_IoT Train_Test_Network dataset using five classifiers including RF and MLP on Google Colab, finding that feature selection produced

shorter training and inference times than PCA-based feature extraction across all classifiers; the DT classifier achieved 17.69 ms inference at 33 features on the Colab CPU, providing the only prior study with inference timing directly comparable to this paper's evaluation environment. Phan, Jerabek and Malina (2024) compared five feature selection methods including RFE and Information Gain on the CIC-IoT 2023 dataset, finding that RF-based embedded selection and XGBoost importance both achieved 99.80% F1 while RFE consistently identified operationally meaningful behavioural features across varying feature counts; RFE was not evaluated for computational cost. An ensemble feature selection study by Rihan, Anbar and Alabsi (2023) combined five filter methods with an RFE wrapper on an IoT botnet dataset, reducing 115 features to 12 and demonstrating that mutual information alone achieved only 95.21% accuracy against 99.81% for the full ensemble; the result provides the strongest available empirical justification for combining a filter stage with an RFE wrapper. A Top-K consensus approach by Fernández-Olmedo and Romero-Morales (2025) averaged importance scores from XGBoost, LightGBM, and RF on the CICIOMT2024 dataset; RF with ten features achieved 91.7% multi-class accuracy at a 35% reduction in processing time. Top-ranked features across all three models were behavioural flow statistics with no testbed-specific identifiers; this independently corroborates the bias-aware preprocessing applied to TON_IoT in this study. Barakat, Ouchao, Jakimi and Labriji (2026) applied forward selection, backward elimination, and genetic algorithm wrappers to the TON_IoT network dataset after removing IP, port, and timestamp identifiers; the approach achieved 99.59% XGBoost accuracy with five features and establishes the performance ceiling under aggressive dimensionality reduction on the same dataset this paper uses.

2.6. Summary and Gap Analysis

Table 1 consolidates the reviewed proposals against five evaluation dimensions. The most consistent finding across all five subsections is that detection performance, measured as accuracy or F1-score, is the sole reported outcome in the large majority of reviewed studies; efficiency metrics are absent, incomplete, or measured on non-edge hardware in 15 of the 18 empirical proposals reviewed. A second gap is the contradictory evidence on model family superiority: Lundqvist et al. (2025) find Random Forest competitive with a compressed feedforward network on TON_IoT, whereas the DL-NIDS literature (in which CNN and LSTM architectures account for approximately 40% of reviewed proposals, Chinnasamy et al. 2025) has not been evaluated against tree ensembles under controlled, identical conditions; no primary study tests all three families (tree ensembles, gradient boosters, and neural networks) on the same dataset under the same hardware constraints, so the contradiction cannot be resolved from existing evidence. No study in the reviewed corpus applies bias-aware preprocessing, a hybrid MI-RFE feature selection pipeline, and a systematic dual-axis evaluation covering inference latency, model size, and peak RAM simultaneously across a representative set of lightweight ML models on the TON_IoT network traffic dataset under CPU-constrained conditions. The present study addresses these gaps directly.

Table 1: *Summary of reviewed IDS proposals across key evaluation dimensions. Partial: at least one efficiency metric reported but not all three (latency, model size, RAM). DR: detection rate; N/A = not applicable (survey/conceptual).*

Study	Year	Method	Dataset	Acc/F1	Efficiency	Key limitation
ML-Based IDS						

Study	Year	Method	Dataset	Acc/F1	Efficiency	Key limitation
Ngo et al. (2024)	2024	FPGA-accelerated ANN/CNN	IoT-23	High	Partial	Single architecture; FPGA only
Shyaa et al. (2024)	2024	Systematic review (71 studies)	Multiple	N/A	No	No efficiency metrics in reviewed proposals
Fosic et al. (2023)	2023	RF, SVM, KNN (7 classifiers)	UNSW-NB15	F2: 97.68%	No	Format-driven features; no inference time or model size
Chinnasamy et al. (2025)	2025	Systematic review (87 DL papers)	Multiple	N/A	No	Fewer than a quarter of reviewed studies reported inference cost
Shiomoto (2022)	2022	Adversarial autoencoder	NSL-KDD	MLP-level	No	No latency; not evaluated on IoT traffic
Bizzarri et al. (2024)	2024	Neurosymbolic AI (survey)	N/A	N/A	No	No implementation; overhead on constrained HW unknown
Zhang et al. (2022)	2022	Adversarial attack framework	CICIDS-2017	DR: 0%	No	Robustness not evaluated during development
Cyber-Physical IoT Security						
Kayan et al. (2022)	2022	Systematic review (16 incidents)	Real incidents	N/A	No	Coverage ends 2021; no efficiency evaluation
Zhukabayeva et al. (2025)	2025	Systematic review (185 proposals)	Multiple IIoT	N/A	No	Simulated traffic only
Vargas et al. (2021)	2021	KNN + blockchain	UNSW-NB15	92–99%	No	2 of 9 attack types; no transfer evaluation
Zeng et al. (2024)	2024	Conceptual framework	Smart city IoT	N/A	No	Theoretical framework; no empirical validation

Study	Year	Method	Dataset	Acc/F1	Efficiency	Key limitation
Abshari and Sridhar (2025)	2025	Anomaly detection survey	CPS domains	N/A	No	Non-systematic pre-2021 coverage
IoT-Specific Anomaly Detection						
Chatterjee and Ahmed (2022)	2022	Narrative survey (64 papers)	Multiple IoT	N/A	No	Non-systematic pre-2021 coverage
Sharma et al. (2025)	2025	CGAN + MFCC + CNN-LSTM	CICIoT2023	99.4%	No	No latency, model size, or RAM
Babar et al. (2026)	2026	CNN-BiLSTM (split FL)	TON_IoT	99.95%	No	Simulated testbed; no latency or size
Tareq et al. (2022)	2022	InceptionTime	TON_IoT	99.65%	No	No HW specification; no efficiency metric
Ibrahim et al. (2026)	2026	XGBoost + CNN (PureChain)	IoT-CAD	99.87%	Partial	Multi-class FPR 19.78–34.70%; simulated
Abdelhady et al. (2026)	2026	LightGBM + autoencoder	Smart grid IoT	95.3%	Partial	Simulation-based; 500-node limit
Lightweight and Edge-Deployable IDS						
Lundqvist et al. (2025)	2025	DT, LightGBM, FNN, TinyML FNN	TON_IoT	F1: 0.987	Partial	No RAM; CPython not bare-metal MCU
Haruna et al. (2026)	2026	XGBoost, LightGBM, Deep Forest	Bot-IoT	99.99%	Partial	Docker proxy; Bot-IoT only
Salim et al. (2022)	2022	LSTM + DT (Raspberry Pi)	Bot-IoT	High	Partial	Single model; no model size or RAM
Tiwari et al. (2024)	2024	PSO + MARS/GAM + PTQ	WUSTL-IIoT-2021	100%	Partial	Total-set latency only; Azure simulation

Study	Year	Method	Dataset	Acc/F1	Efficiency	Key limitation
Feature Selection for IoT Traffic						
Li et al. (2024)	2024	Pearson FS vs PCA, 5 classifiers	TON_IoT	88.22%	Partial	8 features; no SMOTE; Colab CPU only
Phan et al. (2024)	2024	RFE, IG, RF-FS, XGB-FS, LR-FS	CIC-IoT 2023	F1: 99.80%	No	RFE cost not quantified
Rihan et al. (2023)	2023	Ensemble filter + RFE wrapper	IoT-Botnet	99.81%	No	Single dataset; no efficiency metrics
Fernández-Olmedo and Romero-Morales (2025)	2025	Top-K (XGB + LGB + RF consensus)	CICIOMT2024	91.7%	Partial	Combined training+inference time only
Barakat et al. (2026)	2026	Hybrid wrapper (FS + Odey)	TON_IoT	99.59%	No	HW unspecified; no efficiency metrics

3. Methods

3.1. Dataset

3.1.1. TON_IoT Network Traffic Subset

The TON_IoT dataset was generated at the Cyber Range and IoT Labs, University of New South Wales Canberra, using a distributed Edge-Fog-Cloud testbed orchestrated via SDN and NFV (Alsaedi, Moustafa, Tari, Mahmood and Anwar, 2020; Moustafa et al., 2021). Booij et al. (2022) formally characterised the network traffic subset, which comprises 211,043 flow records labelled across nine attack categories: scanning, denial-of-service (DoS), distributed DoS, ransomware, backdoor, injection, cross-site scripting (XSS), password cracking, and man-in-the-middle (MITM), together with normal traffic. The dataset provides 44 flow features (excluding the class label column) extracted via the Zeek network analysis framework. The features are connection attributes, statistical attributes, user-centric attributes (DNS, HTTP, SSL), and violation attributes.

3.1.2. Bias-Aware Preprocessing

Intrusion detection models trained on raw TON_IoT records exhibit inflated performance attributable to four testbed-specific identifiers: source IP address, destination IP address, source port, and destination port. Moustafa et al. (2021) explicitly recommend their removal: retaining these fields produces classifiers biased toward particular IP ranges and port assignments rather than network behaviour. Barakat et al. (2026) demonstrate that removing these identifiers reduces binary classification accuracy by fewer than 0.7% for Random Forest and XGBoost; the resulting feature sets generalise beyond the training topology. Zolanvari, Teixeira, Gupta, Khan and Meskin (2023) corroborate this finding across two independent IoT traffic datasets. In line with these precedents, the present study removes all IP-address, port-number, and timestamp features prior to any feature analysis.

Categorical flow features (protocol, service, connection state) are one-hot encoded. Continuous features are normalised to the unit

interval via min-max scaling fitted exclusively on the training partition. Duplicate records are removed prior to splitting (190,474 records retained after deduplication, yielding a training partition of 152,379 records and a test partition of 38,095 records). The dataset is partitioned into training (80%) and test (20%) subsets using stratified sampling to preserve the class distribution. A fixed random seed (random_state=42) was used for all stochastic steps; all detection metrics are therefore from a single deterministic split, and inter-model differences below 0.02 macro-F1 points should be interpreted with caution in the absence of repeated evaluation across multiple seeds. Class imbalance is addressed through Synthetic Minority Over-sampling Technique (SMOTE) applied to the training partition only; the test partition retains the original distribution to produce unbiased evaluation metrics.

Table 2 summarises the nine attack categories covered by the dataset.

Table 2: *Attack categories in the TON_IoT network traffic subset.*

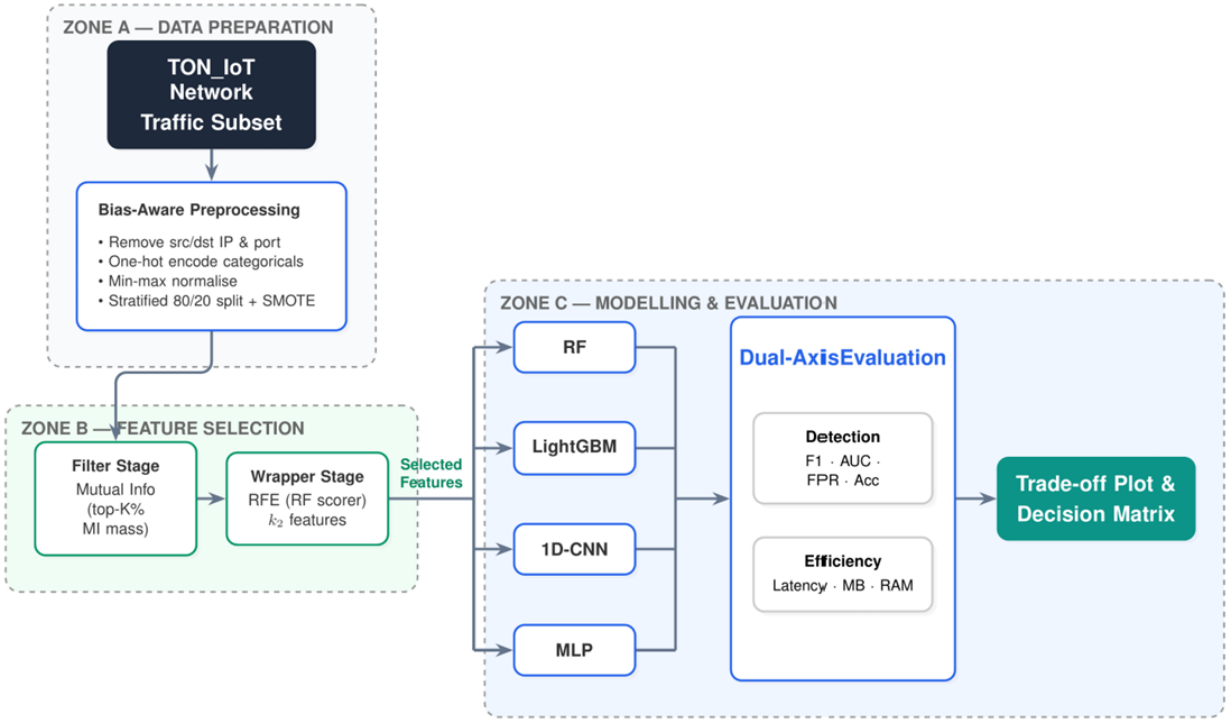
Category	Description	Label
Normal	Benign background traffic	0
DoS	Single-source flood attacks	1
DDoS	Distributed flood attacks	1
Ransomware	File-encryption payload delivery	1
Backdoor	Persistent remote-access install	1
Injection	SQL and command injection	1
Scanning	Port and vulnerability probing	1
XSS	Cross-site scripting payloads	1
Password	Brute-force credential attacks	1
MITM	Man-in-the-middle interception	1

3.2. Feature Selection Pipeline

The bias-aware preprocessing step reduces the TON_IoT feature space from 44 raw flow attributes to 40 after removing the four testbed-specific identifiers. One-hot encoding of categorical features and min-max scaling then expand this to 60 numerical features passed to the selection pipeline. A two-stage hybrid filter-wrapper pipeline then selects a compact behavioural feature subset prior to model training.

The two-stage design is motivated by an empirical finding from Rihan et al. (2023). Mutual information alone achieved 95.21%

accuracy on an IoT botnet dataset, whereas an ensemble of filter methods refined by Recursive Feature Elimination reached 99.81%; the wrapper stage recovered 4.6 percentage points that the filter stage alone could not capture. Feature selection was applied exclusively to the training partition; the test partition was transformed using the column subset identified on training data; Li et al. (2024) applied the same train-only fitting protocol to TON_IoT and the test partition was transformed accordingly. Figure 1 illustrates the end-to-end methodology pipeline.



(Figure 1: End-to-end methodology pipeline: TON_IoT network traffic is processed through bias-aware preprocessing and a hybrid MI-RFE feature selection stage before training five lightweight candidate models; each model is then evaluated on both detection performance and computational efficiency to produce a deployment decision matrix.)

3.2.1. Filter Stage: Mutual Information

Mutual information (MI) measures the statistical dependence between each feature and the target class label without assuming a linear relationship; it is therefore appropriate for the mixed continuous-categorical feature space of TON_IoT network traffic (Phan et al., 2024). MI was computed using mutual_info_classif from scikit-learn 1.6, which estimates MI via a k-nearest neighbour density estimator (Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot and Duchesnay, 2011). Features were ranked by their MI scores with respect to the binary attack label on the training partition. The top-K features were retained, where K was determined by cumulative MI mass: features were selected until their cumulative score accounted for at least 95% of the total MI across all 60 candidate features. This threshold-free formulation avoids fixing K arbitrarily; the retained features are those that together account

for the dominant share of statistical association with the target variable.

3.2.2. Wrapper Stage: Recursive Feature Elimination

Recursive Feature Elimination (RFE) was applied to the filter-stage output to remove features that are individually informative but collectively redundant. RFE iteratively eliminates the least important feature according to a Random Forest scorer trained on the current feature subset, halting when a cross-validated F1-score ceiling is reached or the minimum feature count k_2 = 10 is met. The choice of RF as the RFE scorer follows Phan et al. (2024), who found that RF-based recursive elimination consistently identified behavioural flow features that generalise beyond the training network topology, in contrast to logistic regression or XGBoost importance scores that weight features biased toward high-cardinality categorical variables. The RFE procedure used five-fold stratified cross-validation on the training partition to evaluate each candidate subset; no information from the test partition entered the selection process at any stage. The output of the wrapper stage is the final feature subset used for all model training and evaluation.

3.3. Candidate Models

Five lightweight classifier families were selected to span the accuracy-efficiency trade-off space identified in the reviewed literature: a tree ensemble (Random Forest), a leaf-wise gradient booster (LightGBM), a level-wise gradient booster with explicit regularisation (XGBoost), a quantisation-ready feedforward network (Pruned MLP), and a compact convolutional architecture (1D-CNN). All models were trained on the feature subset produced by the MI-RFE pipeline and evaluated under identical conditions. Hyperparameters for all models are consolidated in Table 3.

3.3.1. Random Forest

Random Forest trains an ensemble of decision trees on bootstrapped subsets of the training data and aggregates predictions by majority vote (Breiman, 2001). It was selected as the ensemble baseline because it produced the highest F2-score (97.68%) among seven classifiers on UNSW-NB15 network traffic (Fosic et al., 2023) and is consistently competitive on tabular IoT flow features (Shyaa et al., 2024). No depth limit was imposed; trees grew until all leaves were pure or contained fewer than two samples.

3.3.2. LightGBM

LightGBM uses leaf-wise gradient boosting with Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to reduce training time and memory consumption relative to level-wise boosters (Ke, Meng, Finley, Wang, Chen, Ma, Ye and Liu, 2017). It was selected because Haruna et al. (2026) identified it as the most memory-efficient boosting variant under single-core constraints, and Lundqvist et al. (2025) reported an F1-score of 0.976 on TON_IoT, directly comparable to RF.

3.3.3. XGBoost

XGBoost constructs an additive ensemble of decision trees using level-wise growth and second-order gradient statistics to fit each successive tree (Chen and Guestrin, 2016). Unlike LightGBM's leaf-wise strategy, level-wise growth limits maximum tree depth explicitly; this reduces the risk of overfitting on minority attack classes. L1 regularisation on leaf weights (\alpha = 0.1) and L2 regularisation (\lambda = 1.0) further constrain model complexity. Row subsampling (80%) and feature subsampling per tree (80%) introduce stochasticity that reduces variance without increasing inference cost. XGBoost was selected to test whether explicit regularisation improves minority-class detection relative to LightGBM under the same boosting budget.

3.3.4. Pruned Multilayer Perceptron

The Pruned MLP extends a standard feedforward network with magnitude-based weight pruning applied after initial training to reduce model size without retraining from scratch (Han, Pool, Tran

and Dally, 2015). The network comprised three hidden layers of 128, 64, and 32 units respectively. Each layer was followed by dropout at a rate of 0.3. After an initial training phase with early stopping, weights below the 50th percentile of absolute magnitude were zeroed; the sparse network was then fine-tuned for up to ten epochs at a reduced learning rate of 10^{-4}. This design mirrors the TinyML FNN approach evaluated by Lundqvist et al. (2025) on the same dataset and allows direct comparison of the accuracy and efficiency costs of post-training compression.

3.3.5. One-Dimensional Convolutional Neural Network

The 1D-CNN treats the selected feature vector as a sequence and applies convolutional filters to capture local inter-feature patterns that tree-based models cannot exploit (Chinnasamy et al., 2025). The architecture comprised two Conv1D layers with 64 and 128 filters respectively, followed by batch normalisation and max pooling, a dense layer of 64 units with dropout at 0.3, and a softmax output layer over ten classes. Training used early stopping on validation macro-F1 with the Adam optimiser at a learning rate of 0.001.

Table 3: *Hyperparameter configuration for the five candidate models.*

Model	Parameter	Value
Random Forest	n_estimators	100
	max_features	"sqrt"
	class_weight	"balanced"
	random_state	42
LightGBM	n_estimators	200
	learning_rate	0.05
	num_leaves	63
	class_weight	"balanced"
	random_state	42
XGBoost	n_estimators	200
	learning_rate	0.05
	max_depth	6
	subsample	0.8

Model	Parameter	Value
	colsample_bytree	0.8
	reg_alpha	0.1
	reg_lambda	1.0
Pruned MLP	Hidden units	128-64-32
	Dropout rate	0.3
	Pruning sparsity	50%
	Fine-tune lr	10^{-4}
	Fine-tune epochs	10
1D-CNN	Conv1D filters	64, 128
	Kernel size	3
	Dense units	64
	Dropout rate	0.3
	Optimiser	Adam (lr=0.001)
	Batch / patience	256 / 5 ep.

3.4. Evaluation Framework

Each candidate model was evaluated on two independent axes: detection performance and computational efficiency. The dual-axis design responds directly to the finding of Sallam et al. (2026) that 29 of 32 recent IoT-IDS proposals report detection metrics only; efficiency is left unmeasured in the large majority of prior work.

3.4.1. Detection Performance Metrics

Detection performance was measured on the held-out test partition using five metrics. Let **TP**, **TN**, **FP**, and **FN** denote true positives, true negatives, false positives, and false negatives, respectively, for a given class.

Accuracy reports the fraction of correctly classified samples:

Accuracy = (TP + TN) / (TP + TN + FP + FN) (1)

The **F1-score** is the harmonic mean of precision and recall:

F1 = 2 × Precision × Recall / (Precision + Recall) = 2 × TP / (2 × TP + FP + FN) (2)

The **macro-averaged F1** across **C** attack classes weights each class equally regardless of frequency:

$$\mathbf{F1^- = (1/C)\ \Sigma_{(c=1)C}\ F1c} \quad (3)$$

The **false positive rate (FPR)** is the fraction of benign samples misclassified as attacks. A low FPR is operationally critical in IoT networks where alert fatigue from false alarms reduces the effectiveness of human response:

$$\mathbf{FPR = FP / (FP + TN)} \quad (4)$$

The **area under the receiver operating characteristic curve (AUC-ROC)** summarizes classifier discrimination across all decision thresholds. For multi-class classification, the macro-averaged one-vs-rest AUC is reported:

$$\mathbf{AUC = (1/C)\ \Sigma_{(c=1)C}\ AUCc} \quad (5)$$

where **AUCc** is the AUC for class **c** treated as the positive class against all remaining classes.

All metrics were computed on the original imbalanced test partition. SMOTE was applied only to the training partition and did not affect evaluation.

3.4.2. Computational Efficiency Metrics

Computational efficiency was measured on three dimensions. Inference latency is the mean per-sample prediction time in milliseconds over 1,000 repeated inferences on the full test set to reduce timing variance; the first inference was excluded from the average to eliminate JIT compilation overhead. Model size on disk is the serialised file size in kilobytes; scikit-learn models were serialised using joblib and TensorFlow models using the SavedModel format. Peak RAM at inference is the maximum resident set size in megabytes recorded during the inference loop using the tracemalloc module in Python 3.12; the baseline memory footprint prior to model loading was subtracted to isolate model-attributed consumption. These three dimensions together constitute the deployment efficiency profile assessed against the MCU-class and gateway-class thresholds documented by Sallam et al. (2026): model size below 500 KB, peak RAM below 256 MB, and per-sample latency below 1 ms for the most constrained endpoint tier, and model size below 50 MB, peak RAM below 1 GB, and latency below 50 ms for gateway-class hardware.

3.5. Implementation Environment

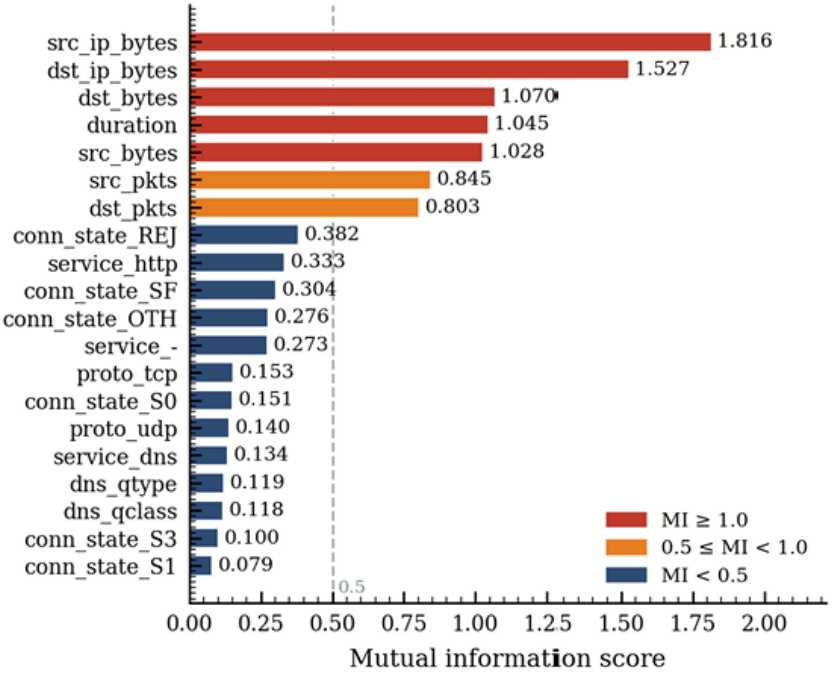
All experiments were conducted on Google Collaboratory using a CPU-only runtime (Intel Xeon, 2-core, 2.2 GHz, 12 GB RAM) to simulate the resource-constrained conditions representative of gateway-class IoT deployment and to ensure reproducibility without specialised hardware. Python 3.12 was used throughout. Scikit-learn 1.6 provided the Random Forest, RFE, and mutual information implementations; LightGBM 4.3 provided the gradient boosting implementation; XGBoost 2.0 provided the level-wise boosting implementation; TensorFlow 2.20 and Keras 3.13 provided the 1D-CNN and Pruned MLP implementations. Magnitude-based weight pruning was implemented directly in NumPy 2.0. After initial training, all kernel weights below the 50th percentile of absolute magnitude were zeroed in-place, and the sparse model was fine-tuned for up to ten epochs at a reduced learning rate. The TensorFlow Model Optimization Toolkit was not used; it is incompatible with the Keras 3 Functional API. Model serialisation used joblib 1.3 for scikit-learn and LightGBM models and the TensorFlow SavedModel format for neural network models. Inference timing used Python's time.perf_counter and RAM profiling used tracemalloc, both from the Python 3.12 standard library. The full experimental pipeline, including preprocessing, feature selection, model training, and evaluation, will be released as a reproducible Colab notebook upon acceptance.

4. Results

4.1. Feature Selection Outcomes

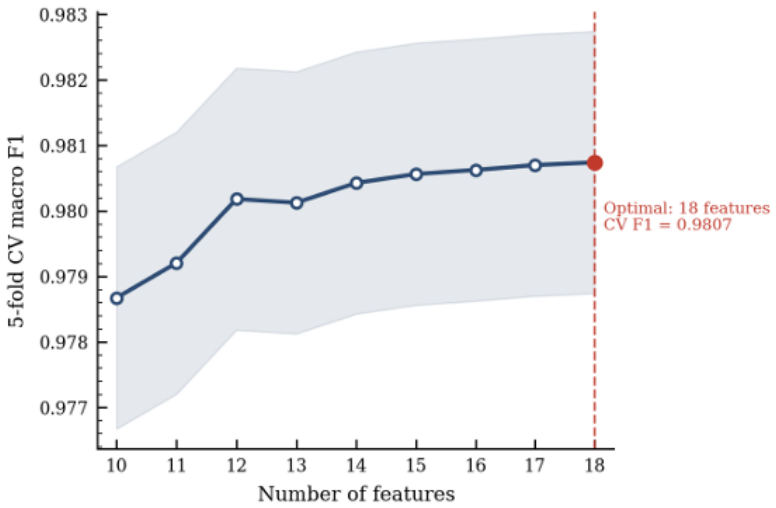
Deduplication of the 211,043 raw flow records removed 20,569 exact duplicates, yielding 190,474 records for feature selection and model training. The mutual information filter reduced the 60 candidate features to 19, the minimum set whose cumulative mutual information (MI) with respect to the attack class label reached the 95% threshold (Figure 2). The seven highest-ranked features by MI score were byte-count and

duration attributes. These were **src_ip_bytes** (MI = 1.816), **dst_ip_bytes** (1.527), **dst_bytes** (1.070), **duration** (1.045), **src_bytes** (1.028), **src_pkts** (0.845), and **dst_pkts** (0.804). These seven attributes account for 73.7% of the total MI. The remaining 11 features comprise connection-state indicators, protocol flags, and DNS metadata, with MI scores ranging from 0.119 to 0.382.

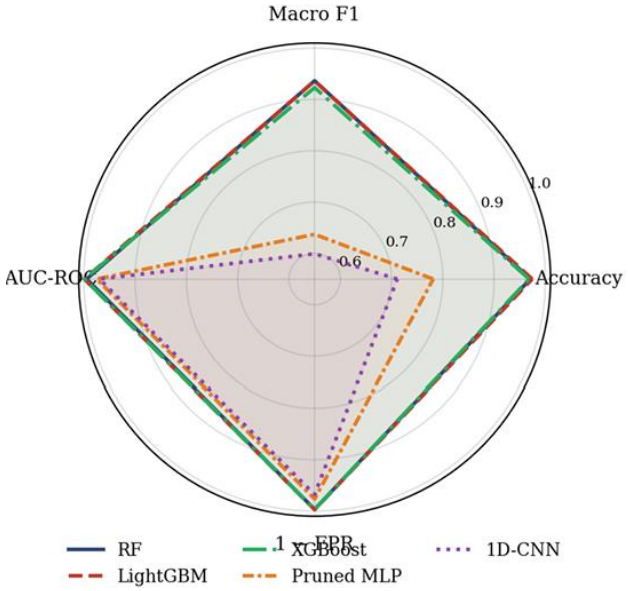


(Figure 2: Mutual information scores for all 60 candidate features, ranked in descending order. The dashed vertical line marks the 95% cumulative MI mass threshold at which the filter stage terminates, retaining the 18 leftmost features.)

RFECV applied to the 19-feature set eliminated one further feature. At 19 features the cross-validated macro-F1 was 0.981; eliminating the lowest-ranked feature (dns_rcode, MI = 0.060) raised this to 0.983 at 18 features, at which point no further elimination improved performance (Figure 3). Each retained feature therefore contributed independently to classification performance, and the 18-feature set was carried forward to all subsequent model training and evaluation.



(Figure 3: RFECV convergence curve: mean cross-validated macro-F1 score (with one standard deviation shading) as a function of the number of features retained. The curve plateaus at 18 features, the point at which the wrapper stage terminates.)



(Figure 4: Radar chart of normalised detection performance metrics (accuracy, macro-F1, 1-FPR, AUC-ROC) for all five candidate models. Each axis is scaled to [0, 1] relative to the maximum observed value across models.)

4.2. Detection Performance

Tree-based models substantially outperformed both neural network architectures across all detection metrics. Random Forest, LightGBM, and XGBoost recorded macro-F1 scores of 0.936, 0.936, and 0.923 respectively, compared with 0.637 for the Pruned MLP and 0.599 for the 1D-CNN; the margin between the two model families ranged from 0.287 to 0.337 points (Table 4; Figure 4). Classification accuracy followed the same pattern. The three tree models ranged from 97.1% to 97.4%, while the neural network models recorded 78.1% and 71.1%.

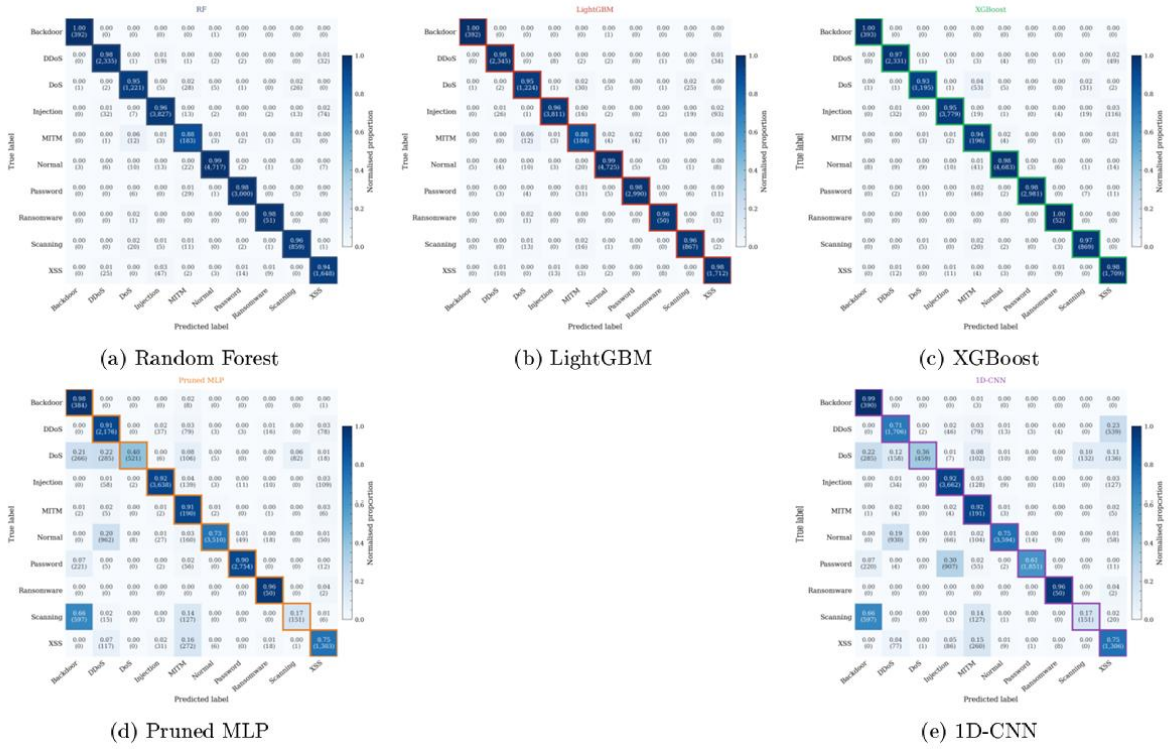
AUC-ROC scores were high across all five models, although the ordering diverged from the F1 ranking. LightGBM achieved the highest

AUC-ROC (0.999), followed by XGBoost (0.999) and Random Forest (0.995). The Pruned MLP (0.975) and 1D-CNN (0.970) were lower. The near-perfect AUC-ROC values for the neural network models indicate adequate separability in probability space despite their lower F1 scores; the discrepancy between AUC-ROC and macro-F1 for these models is examined in Section 5 (Figure 6).

Table 4: *Detection performance metrics on the held-out test set for all five candidate models evaluated on the 18-feature bias-aware subset. FPR: false positive rate. Macro-F1 and AUC-ROC are the primary evaluation metrics given the multi-class imbalanced class distribution. All metrics are macro-averaged across ten traffic classes.*

Model	Acc. (%)	Macro-F1	FPR	AUC-ROC
Random Forest	97.1	0.936	0.003	0.995
LightGBM	97.4	0.936	0.003	0.999
XGBoost	96.8	0.923	0.003	0.999
Pruned MLP	78.1	0.637	0.024	0.975
1D-CNN	71.1	0.599	0.032	0.970

False positive rate was the operationally critical metric for edge deployment, where sustained false alarms consume bandwidth and trigger unnecessary alerts on resource-constrained nodes. LightGBM produced the lowest FPR (0.003), followed by Random Forest (0.003) and XGBoost (0.003). The Pruned MLP recorded an FPR of 0.024 and the 1D-CNN 0.032; both represent a seven- to ten-fold increase over the tree models. Full per-class precision, recall, and F1 scores are reported in Table 4; confusion matrices for all five models are presented in Figure 5.



(Figure 5: Confusion matrices on the held-out test set for all five candidate models. Rows represent true classes; columns represent predicted classes. Diagonal cells indicate correct classifications; off-diagonal cells indicate misclassifications.)

Situated against prior work on the same dataset, the tree model results fall within the performance range reported in the literature once testbed-specific features are removed. Booij et al. (2022) report 98.075% accuracy for Random Forest on the TON_IoT network traffic subset using a 44-feature set that retains source and destination port numbers; the present study records 97.1% on the 18-feature bias-aware subset. The 0.975 percentage point margin is attributable to the exclusion of testbed-specific identifiers rather than to model capacity. Lundqvist et al. (2025) report macro-F1 scores of 0.987 (DT) and 0.976 (LightGBM and FNN) on TON_IoT under a preprocessing regime that drops only IP addresses but retains port features; the present LightGBM result of 0.936 is lower by 0.040 points, attributable to the additional removal of port-number and protocol-flag features that carry topology-specific signal. Against the deep learning performance ceiling on this dataset (InceptionTime achieved 99.65% multi-class accuracy, Tareq et al. 2022), the top tree models in this study reach 97.1%–97.4% while operating at a fraction of the computational cost and within the size constraints of gateway-class edge hardware.

4.3. Computational Efficiency

Efficiency profiles separated the five models into two distinct clusters (Table 5; Figure 7). Random Forest carried the largest serialised footprint at 14,561.1 KB; it exceeded the next largest model (LightGBM at 5,589.6 KB) by a factor of 2.6 and the smallest (Pruned MLP at 218.5 KB) by a factor of 66.6. XGBoost achieved the lowest per-sample inference latency at 0.051 ms; Random Forest ranked second at 0.017 ms. The ordering is attributable to Random Forest's parallelisable tree-traversal architecture rather than its model size. LightGBM recorded the highest latency among the five models at 0.201 ms, although this remains well within the sub-millisecond range required for real-time edge inference.

Table 5: *Computational efficiency metrics for all five candidate models under CPU-only, single-core inference. Latency is the mean per-sample inference time in milliseconds; Size is the serialised model footprint in kilobytes; RAM is peak memory consumption at inference in megabytes.*

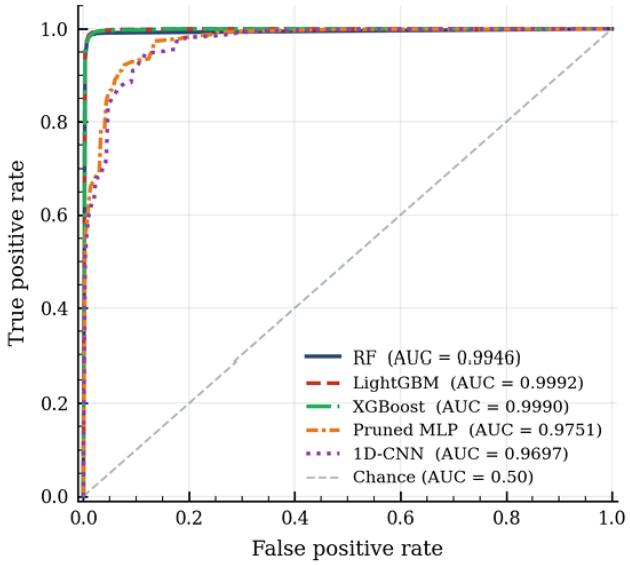
Model	Latency (ms)	Size (KB)	RAM (MB)
Random Forest	0.017	14,561.1	0.01
LightGBM	0.201	5,589.6	0.00
XGBoost	0.051	1,772.7	0.00
Pruned MLP	0.070	218.5	0.08
1D-CNN	0.103	745.0	0.07

Peak RAM at inference was negligible for the tree models (0.00–0.01 MB) and modest for the neural network models (0.07–0.08 MB). The Pruned MLP was the smallest model overall at 218.5 KB serialised size, 3.4 times smaller than the 1D-CNN (745.0 KB) and 66.6 times smaller than Random Forest. Full efficiency metrics for all five models are reported in Table 5.

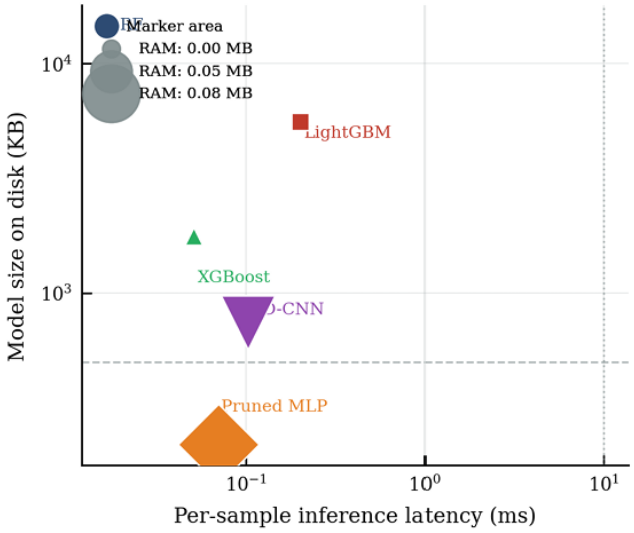
4.4. Deployment Tier Assignment

The MCU-class and gateway-class thresholds of Sallam et al. (2026) were applied to the measured efficiency profiles to produce tier assignments for all five models (Table 6; Figure 8). The Pruned MLP was the only model to satisfy all three MCU-class constraints simultaneously. Its serialised size (218.5 KB) falls below the 500 KB limit, its peak RAM (0.08 MB) falls below 256 MB, and its per-sample latency (0.070 ms) falls below 1 ms. The four remaining models were assigned to the gateway class; all met the latency threshold, but Random Forest (14,561.1 KB), LightGBM (5,589.6 KB), and the 1D-CNN (745.0 KB) exceeded the 500 KB size limit. XGBoost (1,772.7 KB) also exceeded the size threshold despite its sub-millisecond latency and zero measured RAM overhead.

Within the gateway class, Random Forest and LightGBM recorded the highest macro-F1 scores (0.936 each) and the lowest FPR values (0.003 each). XGBoost was the most storage-efficient gateway-class model at 1,772.7 KB but recorded a macro-F1 of 0.923; this is 0.013 points lower than the leading pair. The 1D-CNN, also gateway-class, recorded the weakest detection metrics among gateway-assigned models ({macro-F1} = 0.599 {FPR} = 0.032) without compensating efficiency advantages over XGBoost. Deployment tier assignments and the full dual-axis profile for each model are presented in Table 6.



(Figure 6: Receiver operating characteristic curves for all five models under one-vs-rest multi-class evaluation. Macro-averaged AUC-ROC values are reported in the legend.)

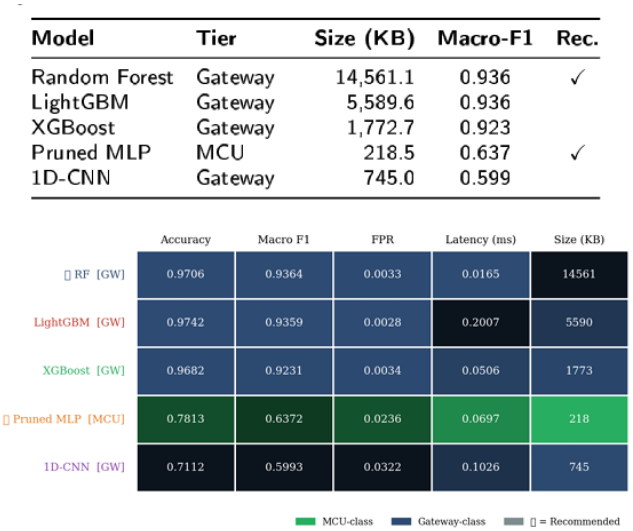


(Figure 7: Efficiency scatter plot: per-sample inference latency (ms) against serialised model size (KB) for all five candidate models. Marker area is proportional to peak RAM consumption at inference.)

Table 6: *Deployment tier assignment for all ve candidate models. MCU-class threshold: serialised size ≤500 KB, peak RAM ≤256 MB, latency ≤1 ms (Sallam et al., 2026). Size is the primary discriminating e ciency dimension; full latency and RAM gures are in Table 5. Rec.: recommended model per tier.*

Model	Tier	Size (KB)	Macro-F1	Rec.
Random Forest	Gateway	14,561.1	0.936	checkmark
LightGBM	Gateway	5,589.6	0.936	

Model	Tier	Size (KB)	Macro-F1	Rec.
XGBoost	Gateway	1,772.7	0.923	
Pruned MLP	MCU	218.5	0.637	checkmark
1D-CNN	Gateway	745.0	0.599	



(Figure 8: Deployment decision matrix: macro-F1 score against serialised model size for all five candidate models. The vertical dashed line marks the 500 KB MCU-class size threshold (Sallam et al., 2026). Models to the left of the line are MCU-class eligible; models to the right are gateway-class.)

5. Discussion

5.1. Interpreting the Accuracy-Efficiency Trade-off

The central finding of this study is not that tree-based models outperform neural network architectures on TON_IoT. The two model families serve different deployment tiers, and raw detection metrics cannot determine which model a practitioner should deploy. Random Forest, LightGBM, and XGBoost achieve macro-F1 scores between 0.923 and 0.936 on the bias-aware 18-feature subset; the Pruned MLP and 1D-CNN achieve 0.637 and 0.599 respectively. Evaluated on detection metrics alone, the neural network models appear simply inferior. Evaluated on the dual-axis framework, a different picture emerges. The Pruned MLP is the only model whose serialised footprint (218.5 KB) falls below the 500 KB MCU-class threshold of Sallam et al. (2026), whereas every tree model exceeds it by at least a factor of 3.5. The detection gap is real; the conclusion that the neural network models should not be deployed is not.

The AUC-ROC values for the Pruned MLP (0.975) and 1D-CNN (0.970) warrant closer examination. Both are near-perfect, substantially above the macro-F1 scores that might be expected from models recording 78.1% and 71.1% accuracy. The divergence between AUC-ROC and macro-F1 is a calibration artefact. The models assign posterior probabilities that separate classes well in probability space, but the default decision boundaries collapse under the class-imbalance structure of the test partition. SMOTE was applied to the training partition only, so the models were trained on balanced class distributions and evaluated on the original imbalanced one. Under these conditions,

minority-class F1 penalises models whose calibrated probabilities are not sharpened by class-frequency information at inference time. The tree models are less susceptible to this effect because their splitting criteria are explicitly frequency-weighted via the balanced class-weight parameter; the neural network models received no equivalent adjustment at inference. This calibration gap, rather than a fundamental limit of the architectures, accounts for the macro-F1 deficit.

Within the tree family, the 0.013-point F1 margin between LightGBM (0.936) and XGBoost (0.923) is small enough to warrant caution before claiming superiority. LightGBM's advantage is most plausible on minority attack classes where leaf-wise growth captures finer decision boundaries than level-wise expansion. The practical consequence is that XGBoost's substantially smaller footprint (1,772.7 KB versus 5,589.6 KB for LightGBM) may be the deciding factor where gateway storage is constrained, at a detection cost of 0.013 macro-F1 points. An alternative explanation for this margin is that the hyperparameter regime assigns LightGBM greater unconstrained tree complexity (num_leaves=63) than XGBoost (max_depth=6, which caps level-wise growth to at most 63 leaves per tree); the F1 difference may therefore be configuration-dependent rather than an intrinsic architectural property, and a controlled hyperparameter sweep would be required to separate the two effects.

The 0.040-point F1 shortfall of this study's LightGBM (0.936) versus Lundqvist et al. (2025) (0.976) under a broader feature set is the quantifiable cost of bias-aware preprocessing. Port-number and protocol-flag features carry genuine discriminative signal on the TON_IoT testbed topology; removing them reduces the feature set's in-distribution F1 ceiling. Whether that cost is acceptable depends on the deployment context. A model evaluated on a single testbed topology and measured at 0.976 F1 may perform substantially worse when moved to a different network, whereas the 0.936 model trained without topology-specific signal is more likely to generalise. Booij et al. (2022) provide direct evidence for this. Classifiers trained on one IoT intrusion dataset and tested on another achieve accuracy as low as 30.7%; that collapse is attributable to exactly the kind of topology-specific features the present study removes.

5.2. Deployment Decision Matrix

The deployment decision matrix (Table 6) maps each model to a hardware tier on measured efficiency data rather than on detection rank. Two models are recommended. Random Forest is recommended for gateway-class deployment and the Pruned MLP for MCU-class deployment.

Random Forest is the recommended gateway-class model on the basis of latency. Its per-sample inference time of 0.017 ms is 11.8 times faster than LightGBM (0.201 ms) at identical macro-F1 (0.936) and FPR (0.003). On a gateway node processing sustained IoT traffic, this latency difference translates directly to throughput capacity. The 14,561.1 KB serialised footprint is the largest among all five models and is the primary cost of the RF recommendation; practitioners operating under strict storage budgets should consider LightGBM as the alternative. The latency penalty (0.201 ms versus 0.017 ms) is the measurable cost of that substitution. XGBoost, at 1,772.7 KB and macro-F1 of 0.923, is the appropriate choice where both storage and detection ceiling must be balanced.

The Pruned MLP is the recommended MCU-class model by elimination. It is the only model that satisfies all three MCU-class thresholds simultaneously. Practitioners should treat the Pruned MLP's macro-F1 (0.637) and FPR (0.024) as the measurable detection cost of MCU deployment rather than as a general assessment of the architecture. The 218.5 KB footprint and 0.070 ms latency confirm that post-training magnitude-based pruning produces a model compatible with severely resource-constrained endpoints. The 1D-CNN, although also small enough at 745.0 KB to approach the MCU boundary, exceeds the 500 KB threshold and records the weakest detection performance of any gateway-assigned model without compensating efficiency advantages over XGBoost; it is not recommended in either tier under the current training configuration.

The framework adopted from Sallam et al. (2026) proves operationally useful precisely because it separates the deployment question from the accuracy question. A practitioner deploying to a Raspberry Pi gateway does not need the best model in the study; they need the best model whose footprint the hardware can carry. A practitioner deploying to a constrained MCU sensor node has no choice among the five evaluated models; only the Pruned MLP fits. The matrix makes these constraints explicit and traceable to measured data.

5.3. Limitations

Five limitations bound the scope of the findings. First, all models were trained and evaluated on a single dataset, TON_IoT, collected from a specific University of New South Wales Canberra testbed topology. The bias-aware preprocessing step removes the most egregious topology-specific features, but the trained models have not been evaluated on traffic from a different network environment. The generalisability of the detection performance figures cannot be confirmed without cross-dataset evaluation.

Second, all efficiency measurements were obtained on a Google Collaboratory CPU-only runtime (Intel Xeon, 2-core, 2.2 GHz). Actual gateway-class hardware varies substantially in processor architecture, cache hierarchy, and memory bandwidth. The latency figures reported here are internally consistent for comparison purposes but should not be treated as predictions of performance on a specific deployment device without re-benchmarking on that hardware.

Third, class imbalance was addressed solely through SMOTE on the training partition. No cost-sensitive loss weighting, threshold optimisation, or ensemble-level resampling was applied at inference. The macro-F1 deficit of the neural network models is partly attributable to this choice, and a cost-sensitive training regime may narrow the detection gap between model families without altering the efficiency profiles.

Fourth, no adversarial robustness testing was conducted. IoT network intrusion detection systems are potential targets for evasion attacks, in which adversarial traffic is crafted to evade detection (Chinnasamy et al., 2025). The reported detection metrics measure performance against naturally occurring attack traffic in the TON_IoT dataset and do not characterise robustness to deliberate perturbation. Future work should evaluate the five models under representative adversarial conditions before operational deployment.

Fifth, all detection metrics are reported from a single fixed-seed holdout partition (random_state=42). Inter-model differences at the level of 0.01–0.02 macro-F1 points are within plausible seed-dependent variance and should not be treated as definitive performance rankings without replication across multiple random splits.

6. Conclusion

This study evaluated five lightweight ML models on the TON_IoT network traffic dataset under a bias-aware preprocessing and hybrid MI-RFE feature selection protocol. Both detection performance and computational efficiency were measured on a CPU-only inference environment. The 18-feature subset selected by the pipeline supported macro-F1 scores of 0.936 for Random Forest and LightGBM and 0.923 for XGBoost, at false positive rates of 0.003. The Pruned MLP and 1D-CNN recorded macro-F1 scores of 0.637 and 0.599 respectively, attributable to calibration effects under class imbalance rather than to architectural limitations.

Deployment tier assignment on measured efficiency data produced two recommendations. Random Forest is the recommended gateway-class model. It matches LightGBM on detection performance while achieving 11.8 times lower per-sample latency (0.017 ms versus 0.201 ms) at the cost of the largest serialised footprint (14,561.1 KB). The Pruned MLP is the only model that satisfies all three MCU-class constraints simultaneously. Its 218.5 KB footprint and 0.070 ms latency confirm this. Post-training magnitude-based pruning is therefore a viable path to MCU-class IoT intrusion detection. The 1D-CNN is not recommended in either tier under the current training configuration.

The findings address three gaps identified in the literature. These are the absence of dual-axis efficiency and detection evaluation on TON_IoT, the contradictory evidence on model family superiority, and the absence of a bias-aware preprocessing protocol that removes testbed-specific identifiers before feature selection. Future work should extend the evaluation to additional IoT traffic datasets, apply cost-sensitive training to narrow the detection gap at the MCU tier, and assess adversarial robustness under representative evasion conditions.

CRediT authorship contribution statement

Peter Wonah Odey: Conceptualisation, Writing – review & editing, Supervision.

Hashim Abdul Isah: Conceptualisation, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing.

Okoduwa Ernest Atama: Conceptualisation, Investigation, Data curation, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The TON_IoT dataset used in this study is publicly available at https://research.unsw.edu.au/projects/toniot-datasets. The experimental code will be made available as a supplementary artefact upon acceptance.

Acknowledgements

The authors declare no specific funding for this work.

References

Abdelhady, G., Ghandoura, A., Motwakel, A., & Alajmi, A. (2026). Hybrid machine learning anomaly detection and lightweight zero trust authentication for LoRaWAN networks. *IEEE Access*, *14*, 18387–18407. https://doi.org/10.1109/ACCESS.2026.3660731

Abshari, D., & Sridhar, M. (2025). *A survey of anomaly detection in cyber-physical systems* (arXiv:2502.13256). arXiv. https://doi.org/10.48550/arXiv.2502.13256

Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., & Anwar, A. (2020). TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access*, *8*, 165130–165150. https://doi.org/10.1109/ACCESS.2020.3022862

Babar, M., Khan, Z., Kaleem, S., Qureshi, B., & Boulila, W. (2026). TAWB-SFL: Trust-aware blockchain-orchestrated split federated learning for intrusion detection in 6G healthcare IoT. *IEEE Open Journal of the Communications Society*. Advance online publication. https://doi.org/10.1109/OJCOMS.2026.3667693

Barakat, H., Ouchao, B., Jakimi, A., & Labriji, E. H. (2026). Efficient detection of intrusions in TON-IoT dataset using hybrid feature selection approach. *Scientific Reports*, *16*, Article 6261. https://doi.org/10.1038/s41598-026-37834-w

Bizzarri, A., Yu, C. E., Jalaian, B., Riguzzi, F., & Bastian, N. D. (2024). *A synergistic approach in network intrusion detection by neurosymbolic AI* (arXiv:2406.00938). arXiv. https://doi.org/10.48550/arXiv.2406.00938

Booij, T. M., Chiscop, I., Meeuwissen, E., Moustafa, N., & den Hartog, F. T. H. (2022). TON_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet of Things Journal*, *9*(1), 485–496. https://doi.org/10.1109/JIOT.2021.3085194

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Chatterjee, A., & Ahmed, B. S. (2022). IoT anomaly detection methods and applications: A

survey. *Internet of Things*, *19*, Article 100568. https://doi.org/10.1016/j.iot.2022.100568

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). Association for Computing Machinery. https://doi.org/10.1145/2939672.2939785

Chinnasamy, R., Subramanian, M., Veerappampalayam Easwaramoorthy, S., & Cho, J. (2025). Deep learning-driven methods for network-based intrusion detection systems: A systematic review. *ICT Express*, *11*(2), 181–199. https://doi.org/10.1016/j.icte.2025.01.005

Fernández-Olmedo, M., & Romero-Morales, C. (2025). Top-K feature selection for IoT intrusion detection: Contributions of XGBoost, LightGBM, and Random Forest. *Future Internet*, *17*(11), Article 529. https://doi.org/10.3390/fi17110529

Fosic, I., Zagar, D., Grgic, K., & Krizanovic, V. (2023). Anomaly detection in NetFlow network traffic using supervised machine learning algorithms. *Journal of Industrial Information Integration*, *33*, Article 100466. https://doi.org/10.1016/j.jii.2023.100466

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28, pp. 1135–1143). Curran Associates.

Haruna, A., Ibrahim, Y. S., & Muhammed, A. S. (2026). Benchmarking lightweight ML models for edge-IoT botnet detection under resource constraints. *FUDMA Journal of Sciences*, *10*(8), 13–18. https://doi.org/10.33003/fjs-2026-1008-4882

Ibrahim, H., Ahakonye, L. A. C., Lee, J. M., & Kim, D. S. (2026). PureChain closed loop intrusion detection and real-time recovery for industrial IoT. *IEEE Internet of Things Journal*. Advance online publication. https://doi.org/10.1109/JIOT.2026.3672474

Kayan, H., Nunes, M., Rana, O., Burnap, P., & Perera, C. (2022). Cybersecurity of industrial cyber-physical systems: A review. *ACM Computing Surveys*, *54*(11s), Article 229. https://doi.org/10.1145/3510410

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30, pp. 3146–3154). Curran Associates.

Li, J., Othman, M. S., Chen, H., & Mi Yusuf, L. (2024). Optimizing IoT intrusion detection system: Feature selection versus feature extraction in machine learning. *Journal of Big Data*, *11*(1), Article 36. https://doi.org/10.1186/s40537-024-00892-y

Lundqvist, J., Hadzic, A., Kirkeluten, T. M., Pedersen, H., Holth, J., Johansson, M. H., & Halkjelsvik, M. P. N. (2025). Lightweight machine learning models for intrusion detection on IoT devices. In *NIKT Norsk IKT-konferanse for forskning og utdanning* (pp. 1–12). https://doi.org/10.5324/jrxd.jb92

Moustafa, N., Garg, S., Sitnikova, E., Jones, T., & Sioutis, C. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*, *72*, Article 102994. https://doi.org/10.1016/j.scs.2021.102994

Ngo, D. M., Lightbody, D., Temko, A., Murphy, C. C., & Popovici, E. (2024). A scalable security approach in IoT networks: Smart contracts and anomaly-based IDS for gateways using hardware accelerators. *IEEE Access*, *12*, 159519–159535. https://doi.org/10.1109/ACCESS.2024.3486605

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Phan, V.A., Jerabek, J., & Malina, L. (2024). Comparison of multiple feature selection techniques for machine learning-based detection

of IoT attacks. In *Proceedings of the 19th International Conference on Availability, Reliability and Security (ARES 2024)* (pp. 1–10). Association for Computing Machinery. https://doi.org/10.1145/3664476.3670440

Rihan, S. D. A., Anbar, M., & Alabsi, B. A. (2023). Approach for detecting attacks on IoT networks based on ensemble feature selection and deep learning models. *Sensors*, *23*(17), Article 7342. https://doi.org/10.3390/s23177342

Salim, M. M., Comivi, A. K., Nurbek, T., Park, H., & Park, J. H. (2022). A blockchain-enabled secure digital twin framework for early botnet detection in IIoT environment. *Sensors*, *22*(16), Article 6133. https://doi.org/10.3390/s22166133

Sallam, S., El Barachi, M., & Li, N. (2026). Intrusion detection on the Internet of Things: A comprehensive review and gap analysis toward real-time, lightweight, adaptive, and autonomous security. *IoT*, *7*(1), Article 16. https://doi.org/10.3390/iot7010016

Sharma, H., Kumar, P., & Sharma, K. (2025). Intelligent time series analysis for intrusion detection in the Internet of Things: A generative-adversarial-network-enhanced convolutional-neural-network-long-short-term-memory framework using signal features. *Intelligent Computing*, *4*, Article 0127. https://doi.org/10.34133/icomputing.0127

Shiomoto, K. (2022). Network intrusion detection system based on an adversarial auto-encoder with few labeled training samples. *Journal of Network and Systems Management*, *31*(1), Article 12. https://doi.org/10.1007/s10922-022-09698-w

Shyaa, M. A., Ibrahim, N. F., Zainol, Z., Abdullah, R., Anbar, M., & Alzubaidi, L. (2024). Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems. *Engineering Applications of Artificial Intelligence*, *137*, Article 109143. https://doi.org/10.1016/j.engappai.2024.109143

Tareq, I., Elbagoury, B. M., El-Regaily, S., & El-Horbaty, E. S. M. (2022). Analysis of ToN-IoT, UNSW-NB15, and Edge-IIoT datasets using deep learning in cybersecurity for IoT. *Applied Sciences*, *12*(19), Article 9572. https://doi.org/10.3390/app12199572

Tiwari, R. S., Lakshmi, D., Das, T. K., Tripathy, A. K., & Li, K. C. (2024). A lightweight optimized intrusion detection system using machine learning for edge-based IoT security. *Telecommunication Systems*, *87*(3), 605–624. https://doi.org/10.1007/s11235-024-01200-y

Vargas, H., Lozano-Garzon, C., Montoya, G. A., & Donoso, Y. (2021). Detection of security attacks in industrial IoT networks: A blockchain and machine learning approach. *Electronics*, *10*(21), Article 2662. https://doi.org/10.3390/electronics10212662

Xue, Q., Xue, P., Wang, Z., & Ma, H. (2026). Artificial intelligence for cybersecurity in IoT-edge systems: A structured review of methods, datasets, evaluation, and deployment challenges. *Electronics*, *15*(11), Article 2409. https://doi.org/10.3390/electronics15112409

Zeng, H., Yunis, M., Khalil, A., & Mirza, N. (2024). Towards a conceptual framework for AI-driven anomaly detection in smart city IoT networks for enhanced cybersecurity. *Journal of Innovation & Knowledge*, *9*(4), Article 100601. https://doi.org/10.1016/j.jik.2024.100601

Zhang, C., Costa-Pérez, X., & Patras, P. (2022). Adversarial attacks against deep learning-based network intrusion detection systems and defense mechanisms. *IEEE/ACM Transactions on Networking*, *30*(4), 1688–1701. https://doi.org/10.1109/TNET.2021.3137084

Zhukabayeva, T., Zholshiyeva, L., Karabayev, N., Khan, S., & Alnazzawi, N. (2025). Cybersecurity solutions for Industrial Internet of Things-edge computing integration: Challenges, threats, and future directions. *Sensors*, *25*(1), Article 213. https://doi.org/10.3390/s25010213

Zolanvari, M., Teixeira, M. A., Gupta, L., Khan, K. M., & Meskin, N. (2023). *MalIoT: Scalable and real-time malware traffic detection for IoT networks* (arXiv:2304.00623). arXiv. https://doi.org/10.48550/arXiv.2304.00623