J Multidisciplinary Scientific Journal

MDPI

Article

Dual-Optimized Genetic Algorithm for Edge-Ready IoT Intrusion Detection on Raspberry Pi

Khawlah Harasheh [1], Satinder Gill [2,*], Kendra Brinkley [2], Salah Garada [2], Dindin Aro Roque [1], Hayat MacHrouhi [1], Janera Manning-Kuzmanovski [1], Jesus Marin-Leal [2], Melissa Isabelle Arganda-Villapando [1] and Sayed Ahmad Shah Sekandary [1]

[1] Department of Information Systems, J. Sargeant Reynolds Community College, Richmond, VA 23228, USA; hm84691@email.vccs.edu (H.M.); jm20386@email.vccs.edu (J.M.-K.)
[2] Department of Electrical Engineering, J. Sargeant Reynolds Community College, Richmond, VA 23228, USA; sgarada@reynolds.edu (S.G.)
* Correspondence: gills4@vcu.edu

Abstract

The Internet of Things (IoT) is increasingly deployed at the edge under resource and environmental constraints, which limits the practicality of traditional intrusion detection systems (IDSs) on IoT hardware. This paper presents two IDS configurations. First, we develop a baseline IDS with fixed hyperparameters, achieving 99.20% accuracy and ~0.002 ms/sample inference latency on a desktop machine; this configuration is suitable for high-performance platforms but is not intended for constrained IoT deployment. Second, we propose a lightweight, edge-oriented IDS that applies ANOVA-based filter feature selection and uses a genetic algorithm (GA) for the bounded hyperparameter tuning of the classifier under stratified cross-validation, enabling efficient execution on Raspberry Pi-class devices. The lightweight IDS achieves 98.95% accuracy with ~4.3 ms/sample end-to-end inference latency on Raspberry Pi while detecting both low-volume and high-volume (DoS/DDoS) attacks. Experiments are conducted in a Raspberry Pi-based real lab using an up-to-date mixed-modal dataset combining system/network telemetry and heterogeneous physical sensors. Overall, the proposed framework demonstrates a practical, hardware-aware, and reproducible way to balance detection performance and edge-level latency using established techniques for real-world IoT IDS deployment.

Keywords: IoT security; intrusion detection system (IDS); genetic algorithm; hyperparameter optimization; feature selection; Raspberry Pi; edge computing; DoS; man-in-the-middle (MitM); data injection; ensemble learning; latency-aware ML

Academic Editor: Georgios Kambourakis

Received: 26 November 2025
Revised: 19 January 2026
Accepted: 23 January 2026
Published: 25 January 2026

1. Introduction

Internet of Things (IoT) systems are expanding rapidly in homes, industries, healthcare, and more, but this growth also widens the attack surface, especially when considering the weaknesses of the IoT caused by environmental limitations. Resource-limited environments are exposed to different types of attacks, mainly Denial of Service (DoS) and Distributed Denial of Service (DDoS). Traditional intrusion detection systems (IDSs) struggle in this context because they need more power, space, and heavy-duty hardware to run, which is not one of the IoT's characteristics, where IoT devices are heterogeneous, and strict latency and energy restrictions are placed at the edge.

To address accuracy and efficiency at the same time, many studies have been performed for optimization, particularly using genetic algorithms (GAs), either to select

important feature sets or to tune model hyperparameters [1–5]. GA-aided IDSs have shown very good results on common benchmarks using some popular datasets such as NSL-KDD and CICIDS. The GA improves detection rates while reducing dimensionality and training costs [2–4,6,7]. There is a lot of recent research where researchers extend the GA and combine it with k-nearest neighbors, deep networks, or ensembles, and this combination increases the accuracy of detecting specific threats such as DDoS or port scans [1,3,8,9]. Another researcher has shown how parallel efforts explore edge orientation, reporting on-device inference latencies on the millisecond scale to demonstrate feasibility on microcontroller-class hardware and smart home gateways [10–12]. Data augmentation paradigms such as federated learning and WGAN-based synthesis have been used to cope with scarce or siloed IoT data while maintaining strong detection performance [13,14].

With all this existing research where authors try to find the best way to combine GAs with IDSs to make them more compatible with the IoT environment, we still have three major practical hurdles to clear. A lot of the current research relies on older or fake datasets [2,4,6,8], while other researchers build GA-based systems that are only designed to handle one type of attack, usually DDoS, or can detect only high-volume attacks and cannot handle low-volume ones, as in the Slowloris attack, which limits their usefulness as a general-purpose, multi-attack detector in actual environments [1,3,9]. Researchers often do not consider the speed of the runtime. Only a few studies try to improve both accuracy and speed at the same time, especially when working with IoT edge devices that have limited computing power or energy [8,10].

This paper targets those gaps with a lightweight, edge-ready IDS trained on a newly collected Raspberry Pi-based dataset that contains sensor readings with system metrics such as CPU, memory, and network usage. In addition, this paper presents a lightweight IDS that detects high- and low-volume attacks at the same time, not only DDoS but also DoS and Slowloris attacks. This paper is designed to create an IDS compatible with IoT deployments that require high detection accuracy under millisecond latency and fit the resource limitations. To overcome these limitations, this work proposes a dual-optimized IDS leveraging feature selection and GA hyperparameter tuning, designed specifically for latency-sensitive IoT deployments. This work makes the following contributions:

- A real, mixed-modal IoT dataset collected on Raspberry Pi-class hardware, combining system and network metrics with heterogeneous physical sensor readings under controlled multi-attack scenarios.
- A leakage-free learning pipeline integrating ANOVA-based feature selection with hardware-aware GA hyperparameter tuning, explicitly bounded by edge device inference constraints.
- A latency-first IDS design capable of detecting both low-volume and high-volume attacks (DoS, DDoS, and Slowloris) with millisecond-level inference on edge hardware.
- A deployment-oriented and reproducible framework, including fixed random seeds, bounded GA search spaces, and device-level runtime measurements.
- A clear path toward multi-objective optimization, extending the framework to jointly consider accuracy, latency, model size, feature sparsity, and energy efficiency in future IoT deployments.

The remainder of this paper is organized as follows. Section 2 reviews related work, comparing our approach with existing genetic algorithm-based and hybrid deep learning intrusion detection systems, and provides a summary table highlighting differences in datasets, attacks covered, accuracy, latency, and real-world deployability. Section 3 presents the methodology in detail, including the data pipeline, Raspberry Pi configuration, attack scenarios, preprocessing procedures, and our dual optimization strategy combining an ANOVA with a genetic algorithm, along with the architecture of the final model. In the

same section, the dataset is described, summarizing feature distributions and correlations with supporting figures for context. Section 4 presents the results and discussion, including overall accuracy, per-class performance metrics, confusion matrices, and system latency, with comparisons against several strong baselines. Finally, Section 5 outlines future work, and Section 6 provides the conclusion, including planned multi-objective extensions and broader deployment strategies.

2. Related Work

Early GA-enhanced IDS research largely concerns legacy benchmarks such as NSL-KDD and CICIDS, either using the GA for feature selection only or pairing the GA with heavyweight models that are difficult to deploy at the edge. For example, a GA-DBN pipeline achieved strong accuracy on benchmark traffic but optimized primarily for the detection rate rather than device-level latency or resource limits [3]. A deep hybrid industrial IoT detector likewise emphasized accuracy with complex stacks, again without on-device validation or millisecond inference guarantees [15–17]. More recent work has improved data handling, class balancing, and ensembles on public corpora [10] yet still evaluates off-device and without explicit runtime budgets.

Several GA-based studies do report notable gains but remain threat-specific, such as DDoS-only, or omit edge validation. OMEGA uses a GA to optimize an ensemble for DDoS detection and reports high accuracy; however, it targets a single attack family and provides no concrete on-device latency figures [4]. GA-FS approaches reduce dimensionality and increase accuracy on medical IoT traffic, but they neither test multiple diverse attacks nor demonstrate edge-class inference at millisecond scales [8]. A classic deep learning IDS on IoT traffic shows high benchmark accuracy as well [1] yet similarly lacks truly mixed-sensor streams and edge deployment constraints.

A smaller stream explores edge orientation directly. A smart home IDS demonstrates on-device performance with ~3.51 ms per-sample latency using compact tree/boosted models; accuracy is ~97.59% on-device and ~99.5% in the cloud [18]. While exemplary in runtime reporting, that work narrows its scope to a specific device/domain and does not combine heterogeneous sensors with system metrics nor optimize via a GA under multi-attack conditions. Finally, privacy-aware or augmentation-centric studies pursue distributional robustness but evaluate primarily on public datasets without device-level latency trade-offs [2].

Our approach differs along four practical axes: it is trained and validated on a new, real, mixed-modal dataset collected on a Raspberry Pi; it detects multiple attacks rather than a single threat and for both low-volume and high-volume attacks; it employs dual optimization using a fast filter FS combined with GA hyperparameter tuning that is leakage-free and hardware-aware; and finally, it achieves great edge latency and low running time together with excellent accuracy, taking into account IoT constraints.

Other research has focused on new IDS evaluation using new wireless network intrusion datasets, such as AWID2 and AWID3, to study and improve machine learning-based detection performance in wireless environments. These datasets were developed to capture rich 802.11 (Wi-Fi) traffic under both normal and attack conditions. Several recent studies use AWID3; for example, Dashtifard et al. use AWID3 with a hybrid ensemble learning framework that integrates multiple classifiers to achieve high detection accuracy across several attack categories, demonstrating the effectiveness of combined models on wireless intrusion benchmarks.

Similarly, Yonbawi et al. evaluate feature transferability with deep learning and machine learning models using AWID and AWID3, highlighting that while high performance is achieved on individual datasets, generalization across different data sources remains

challenging. Other studies, as listed in Table 1, have implemented lightweight convolutional neural network architectures on AWID3 data, achieving competitive accuracy and inference performance suitable for practical deployments in wireless contexts.

Table 1. Comparison of related work and proposed approach.

Ref.	Dataset	Attacks	Accuracy	Latency	IoT-Compatible	Limitations
Ours	New and up-to-date dataset 2025	DoS, DDoS, and Slowloris	98.9%	~4.3 ms/sample	Yes	-
[3]	NSL-KDD	DOS, Probe, R2L, and U2R	97.78–99.45%	Not reported	No	Older data Heavy model No device latency
[4]	HL-Iot, ToN-IoT, CICIDS 2017	DDoS	90%	Not reported	Unclear	Single-threat No device latency
[6]	IoMT traffic	Unclear	92.98–96.08%	Not reported	Unclear	Narrow domain No device latency
[7]	NSL-KDD	DOS, Probe, R2L, and U2R	99%	Not reported	Unclear	Old data No device latency
[8]	NSL-KDD, UNSW-NB15	DOS, Probe, R2L, and U2R	96.92–99.77%	Not reported	No	Old data No device latency
[10]	TON-IoT	DDoS, Ransomware, Backdoor, Injection, XSS, and Password cracking	76.49–100%	Not reported	No	Complex DL Heavy model No device latency
[11]	TON-IoT	DoS, MitM, scanning, and Ransomware	97.59%	~3.51 ms/device	Yes	Narrow device/domain; Not mixed-modal
[12]	AWID3	Multi-class Wi-Fi attacks	99.75%	Not reported	No	Wi-Fi only, no device latency
[13]	AWID2, AWID3	Wi-Fi attack classes	98–99%	Not reported	No	Transferability issues
[14]	AWID3	Wi-Fi intrusion categories	Competitive	Not reported	No	Not multi-modal IoT
[15]	AWID	Injection, Impersonation, and Flooding	99.4%	Not reported	No	Wi-Fi only, no device latency

Despite these advancements, it is important to note that AWID2/AWID3 are wireless IDS benchmarks for Wi-Fi traffic, not specifically tailored to heterogeneous IoT system behavior or datacenter IoT environments. Their focus on wireless frames and network traffic patterns distinguishes them from multi-modal IoT sensor datasets that combine physical, system, and network metrics. This paper complements these efforts by addressing real IoT sensor and network data in a datacenter simulation, optimizing for both multi-attack detection and hardware-aware deployment, aspects not covered by existing Wi-Fi-centric studies.

Table 1 provides a contextual overview of related IDS studies that use different datasets and deployment assumptions. Because cross-dataset comparisons cannot isolate the contri-

bution of the proposed GA–ANOVA framework, we validate the "dual-optimization" claim using within-dataset baselines on our 2025 mixed-modal dataset (see Table 2), including GA tuning versus standard hyperparameter search methods under identical preprocessing and evaluation settings.

Table 2. Baseline vs. GA-optimized IDS performance.

Model	Number of Features	Accuracy (%)	Optimization
Baseline (no GA)	10	99.20	None
Proposed (GA-optimized)	10	98.85	Genetic Algorithm
Grid Search (within-dataset baseline)	10	94.28	Grid Search
Random Search (within-dataset baseline)	10	94.28	Random Search

3. Methodology

This study develops and evaluates a lightweight IDS pipeline designed for edge-constrained IoT devices. As a first step, a baseline IDS was designed and tested for securing desktops and regular devices, and then it was enhanced and transformed into a new lightweight IDS. The main goal is to achieve high detection accuracy with millisecond-scale latency using real, mixed-modal data recorded on a Raspberry Pi-class platform. The methodology includes a reproducible experimental setup that combines network and system telemetry with heterogeneous physical sensors as seen in Figure 1. It also includes controlled attack campaigns covering low-volume DoS, high-volume DoS/DDoS, and Slowloris attacks and implements a leakage-free learning pipeline that first stabilizes the feature space and then applies a hardware-aware GA to tune model hyperparameters under stratified cross-validation. The final performance is evaluated on a held-out test split and reported in terms of accuracy, per-class metrics, confusion matrices, and end-to-end inference time (ms/sample) measured on the device.

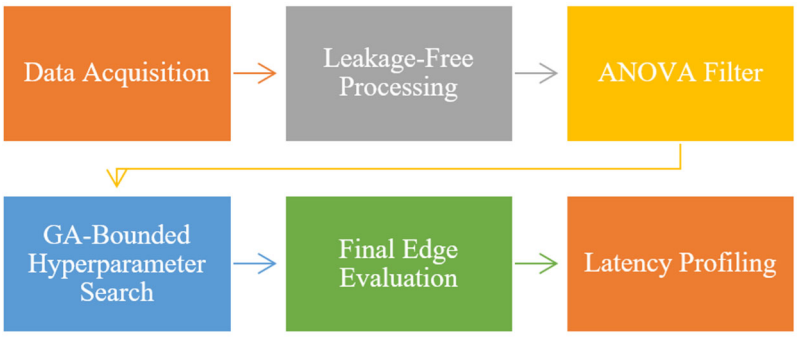

Figure 1. Proposed dual-optimization framework.

3.1. Dataset Description

This study uses a dataset collected at a large mid-Atlantic community college over 17 days, with an actual data collection time of around 336.7 h from an IoT-enabled datacenter simulation on a Raspberry Pi-class edge node that logs both cyber and physical sensors at the same time. Each row is a timestamp-aligned snapshot of system/network load, such as CPU, memory, and inbound/outbound network throughput, concatenated with heterogeneous ambient sensors like motion/PIR, temperature, humidity, smoke, water, IR, distance/ultrasonic, and sound sensors, yielding a compact tabular representation $x_t \in \mathbb{R}^d$. The combined file contains 1,464,239 records and 14 numeric features after preprocessing, with four ground-truth classes: normal, low-volume DoS, high-volume DoS/DDoS, and Slowloris. Class counts show a realistic long-tailed structure, where normal and Slowloris

dominate, while DoS and DDoS are less frequent, which motivates stratified splitting and macro-averaged metrics in evaluation. Table 3 reports the record counts by class.

Table 3. Record counts by class.

Class	Records
Normal	756,918
DoS	59,265
Slowloris	557,472
DDoS	90,584
Total	1,464,239

Figure 2 shows the feature correlation heatmap across numeric features, showing blocks of redundancy among system counters and certain physical sensors. This supports the use of a filter stage before model training to reduce variance and accelerate optimization.

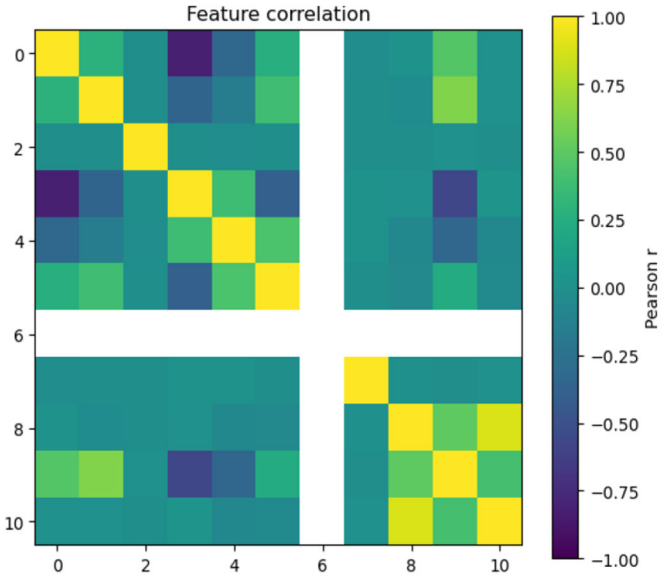

Figure 2. Feature correlation heatmap.

Figure 3 illustrates a t-SNE projection of the sampled training data, showing clear clustering between normal and attack classes, confirming that feature separation is achievable in the reduced-dimensional space.

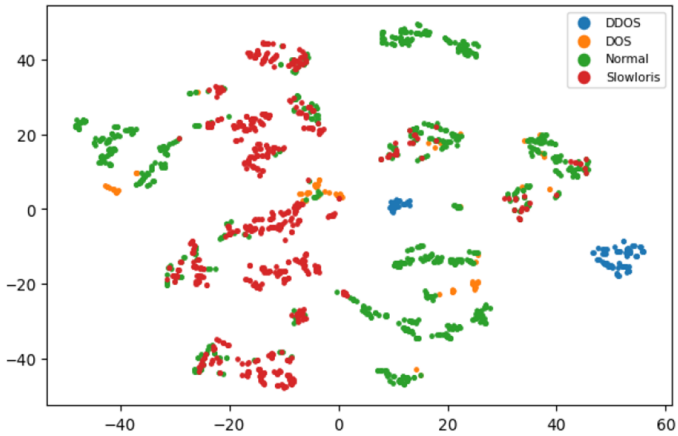

Figure 3. t-SNE (sampled) of training set.

3.1.1. Experimental Setup

The experimental setup was designed to simulate a real datacenter IoT device environment in which the IDS must operate under tight latency and resource constraints while observing heterogeneous IoT records. The full pipeline was deployed on multiple Raspberry Pi 5 units, serving as central processing and logging devices, equipped with various sensors, including temperature, humidity, smoke, water leakage, motion, infrared, distance, and sound detectors. This arrangement grounds model development in device-level constraints such as CPU, memory, and network bandwidth usage rather than desktop assumptions, and it ensures that the resulting dataset reflects the mixed-modal behavior characteristic of real deployments, including low-volume and high-volume perturbations, as well as low-and-slow connection exhaustion. The following subsections detail the hardware/software environment, the data acquisition protocol and feature composition, and the construction of attack scenarios with precise timing for ground-truth labeling.

3.1.2. Raspberry Pi Environment

All experiments were run on a Raspberry Pi 5-class edge device configured to mirror the resource constraints expected in practical IoT deployments and to simulate its environment and limitations. The IDS operates directly on the edge to eliminate external network issues, and latency is measured as wall-clock time per test sample using high-resolution timers. This device-level timing is central to this study because calculating accuracy alone is insufficient without taking it into account. Let T_pred denote the total wall-clock time to score N test samples; the per-sample latency (ms/sample) is given in Equation (1).

$$\text{Latency} = \frac{\text{T}_\text{p}\text{red}}{\text{N}\left(\frac{\text{ms}}{\text{sample}}\right)} \tag{1}$$

where ms/sample denotes the end-to-end inference latency of the deployed pipeline, i.e., applying the already-fitted preprocessing transforms (imputation if needed, scaling, and ANOVA projection learned offline on the training set) followed by model prediction. This metric excludes training and excludes fitting the scaler/ANOVA.

3.1.3. Sensors and Data Collection

The Raspberry Pi simultaneously records the system and network metrics, in addition to the physical sensor readings. This mixed-modal design captures the cyber footprint of attacks alongside contextual/operational signals from the surrounding environment (e.g., ambient conditions and activity), which are not interpreted as causal physical effects of cyberattacks. Each record at time t is a concatenation of system/network metrics and physical sensor readings, as shown in Equation (2). Ground-truth labels y_t ∈ {Normal, DoS_low, DoS/DDoS_high, Slowloris} are assigned using attack script start and stop times.

$$\begin{aligned} \text{x\_t} = [&\text{CPU\_t, MEM\_t, NET\_in, t, NET\_out, t} \mid \text{motion\_t, temp\_t, humid\_t, smoke\_t,} \\ &\text{water\_t, IR\_t, dist\_t, sound\_t}] \in \mathbb{R}\text{ˆd} \end{aligned} \tag{2}$$

3.1.4. Attack Simulations

While the sensors were recording normal traffic, as described in Section 3.1.3, several attacks were executed using a Linux laptop (Dell, Richmond, USA), including low-volume DoS, high-volume DoS/DDoS, and Slowloris. Finally, the records were manually labeled as normal, DoS, DDoS, or Slowloris.

3.2. Learning Pipeline

This study presents a comparison between two distinct IDS designs: a baseline model and an enhanced final version. The baseline design, shown in the first flowchart in Figure 4, serves as an initial proof of concept, using a boosted tree model with fixed hyperparameters and unoptimized settings. This model achieved an impressive accuracy of around 99% during testing on a standard computer, but it was not specifically tailored for use on an IoT device. The process followed a straightforward sequence: first, the database was loaded; then the data was label-encoded; and finally, it was split into 80% for training and 20% for testing. During the data preprocessing stage, three main steps were performed. First, missing data were handled by imputing values using the mean. Next, the features were standardized using z-scores to ensure they were on the same scale. Finally, the most relevant attributes were selected through an ANOVA. In the training stage, the boosted tree module with fixed hyperparameters was used, and during testing, the model's performance was assessed by generating metrics such as accuracy, a Classification Report, a confusion matrix, and inference time.

Figure 4. Baseline design flowchart.

The enhanced final version, shown in the flowchart in Figure 5, takes a direct approach to addressing resource limitations in the IoT. This design incorporates a learning pipeline that merges efficient data preparation with hardware-aware model selection, ensuring that detection quality and edge deployment are optimized together. The process begins by stabilizing the representation through integrity checks, stratified splitting, standardization and imputation for training only, and ANOVA-F filter selection. This step produces a compact feature space that reduces variance and accelerates the subsequent search (Equations (3)–(7)).

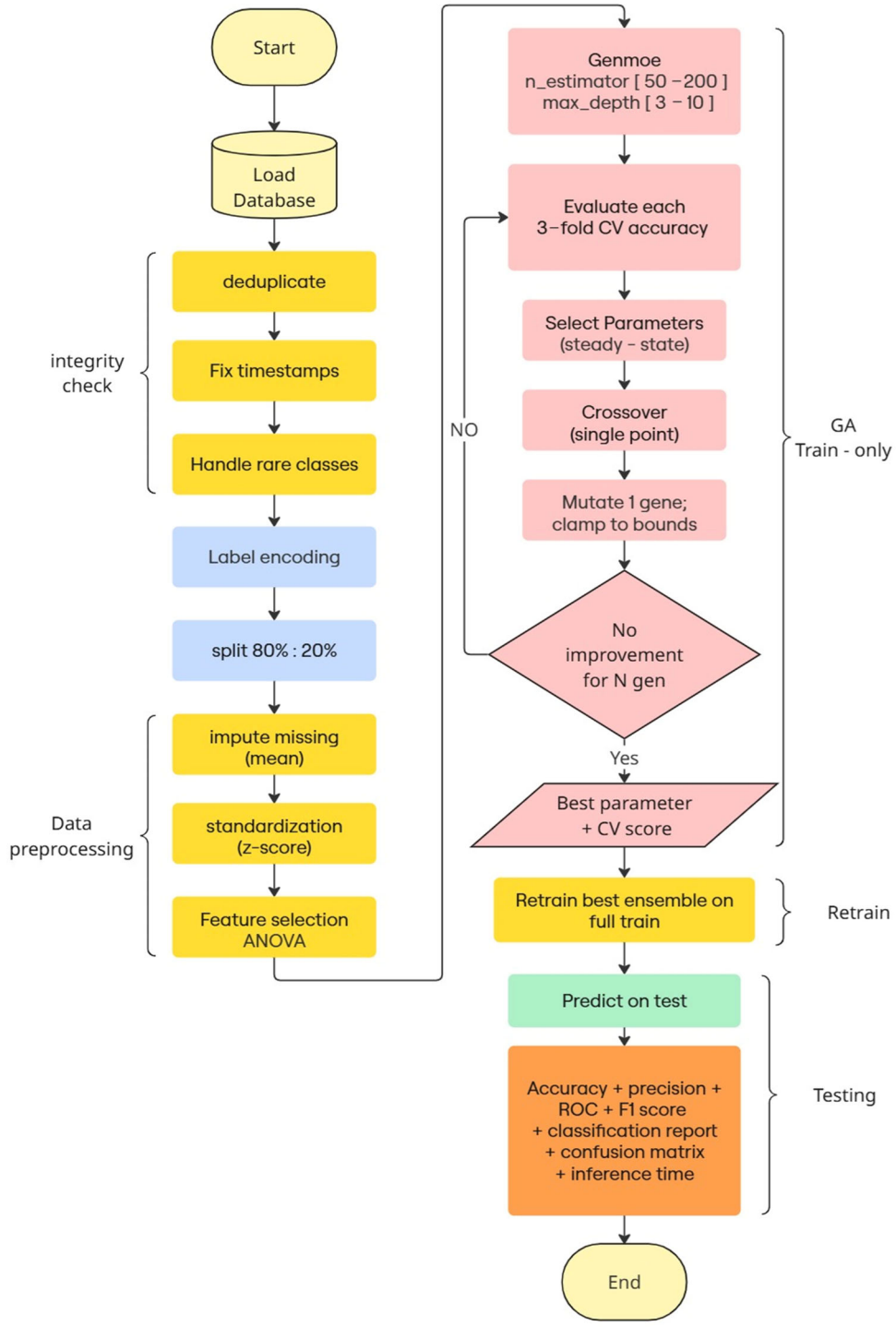

Figure 5. The enhanced design flowchart.

On this stabilized training set, a genetic algorithm explores discrete, bounded hyperparameters for a boosted tree ensemble while maximizing cross-validated accuracy (Equations (8)–(11)). An optional wrapper balances accuracy with feature sparsity and latency through a composite objective (Equation (12)). Once the best configuration is found, it is retrained on the entire training set and evaluated once on the held-out test split, with the results reported in terms of accuracy, per-class metrics, and a confusion matrix (Equations (13)–(15)). Device-level latency is measured in milliseconds per sample to ensure that the selected model meets edge timing requirements (Equation (1)), and dispersion is summarized to assess runtime stability (Equation (16)). Finally, hyperparameter bounds are linked to the theoretical inference cost of depth-bounded ensembles (Equation (17)),

demonstrating that the integer search space is both statistically sound and computationally feasible on Raspberry Pi-class hardware (Vilros, Lakewood, USA).

3.2.1. Data Preprocessing

The process begins by loading the dataset and performing integrity checks such as duplicate removal, timestamp normalization, and verification of the presence of required fields. Rare classes are addressed before splitting: if any label occurs fewer than two times, it is either merged or removed to enable stratified sampling. The dataset is then partitioned using a stratified train/test split, typically 80 to 20, with a fixed random seed. From the training portion only, a StandardScaler is fitted to homogenize feature scales, and a mean imputer is applied to resolve missing entries; the learned transformations are subsequently applied to both training and test partitions. To stabilize modeling and reduce variance, the filter-based feature selection stage ANOVA is applied again, fitted strictly on the training split, and then used to project both partitions.

Let D = {(x_i, y_i)}_{i = 1}^n. If the least frequent class contains fewer than two samples, it is merged or removed to enable stratified sampling. With a test proportion α (e.g., 0.2), indices are partitioned by class, as in Equation (3).

$$I\_c = \{i{:}y\_i = c\},\ I\_c\hat{}test \subset I\_c,\ |I\_c\hat{}test| = \lfloor \alpha \cdot |I\_c| \rfloor,\ I\_c\hat{}train = I\_c \backslash I\_c\hat{}test \tag{3}$$

Standardization is fitted on the training split only (Equations (4) and (5)) and applied to both the training and test sets. Mean imputation is likewise fitted on the training data (Equation (6)). Filter-based feature selection uses ANOVA-F, computed on the training split only (Equation (7)), after which both partitions are projected onto the top-K features.

$$\begin{gathered} \mu\_j = (1/|I\_train|) \cdot \sum\_\{i \in I\_train\}\ x\_\{ij\}, \\ \sigma\_j = \mathrm{sqrt}((1/|I\_train|) \cdot \sum\_\{i \in I\_train\}\ (x\_\{ij\} - \mu\_j)\hat{}2 + \varepsilon) \end{gathered} \tag{4}$$

$$\tilde{x}_{ij} = \frac{(x_{ij} - \mu_j)}{\sigma_j} \tag{5}$$

$$\begin{gathered} m\_j = (1/|\{i \in I\_train{:}x\_\{ij\} \neq NaN\}|) \cdot \sum\_\{i \in I\_train,\ x\_\{ij\} \neq NaN\}\ x\_\{ij\};\ x\_\{ij\} \\ x\_\{ij\} \text{ if not NaN else } m\_j \end{gathered} \tag{6}$$

$$\begin{aligned} F &= MS_{(between)}/MS_{(within)} \\ &= \left(\left(sum_{(i=1)}^{k} n_i(\overline{x}_i - \overline{x})^2\right)/(k-1)\right) \Big/ \left(\left(sum_{(i=1)}^{k} sum_{(j=1)}^{(n_i)}\left(x_{(ij)} - \overline{x}_i\right)^2\right)/(N-k)\right) \end{aligned} \tag{7}$$

where k is the number of groups/classes, n_i is the number of samples in group i, $N = \sum_i n_i$ is the total number of samples, $\overline{x}_i$ is the mean of feature x in group i, and $\overline{x}$ is the overall mean.

Latency definition and protocol: We report latency as end-to-end inference time per sample for the deployed pipeline (transform + predict). At inference time, we apply the already-fitted preprocessing transforms (scaling and ANOVA projection learned offline on the training set) followed by model prediction. Training and fitting preprocessing steps are excluded. For desktop timing, latency is computed using batch timing (timing N samples and dividing by N) to avoid microsecond-level timer granularity artifacts.

3.2.2. Genetic Algorithm Hyperparameter Optimization

To mitigate data leakage and temporal artifacts, we apply a leakage-free training protocol. Timestamps are used only to align sensor streams and define labeled attack windows and are not included as predictive features. All preprocessing operations, mean imputation, z-score standardization, and ANOVA-F feature selection, are fit exclusively on

the training partition and then applied to the held-out test partition. Beyond the default stratified 80/20 split, we additionally perform a time-blocked evaluation, where training uses earlier days and testing uses later days, to reduce the risk that temporally correlated patterns inflate ROC/PR performance. This evaluation better reflects deployment scenarios in which the IDS is trained on historical behavior and tested on future unseen periods.

Integer hyperparameters for a boosted tree ensemble are optimized using a hardware-aware GA. The chromosome encodes θ = [n_estimators, max_depth] with bounded, discrete ranges (Equation (8)). Post-mutation clamping enforces these bounds (Equation (9)). Fitness is defined as the mean 3-fold stratified cross-validation accuracy on the training partition (Equation (10)). Steady-state selection, single-point crossover, and single-gene mutation with probability p_m are used (Equation (10)), with early stopping applied when fitness saturates. An optional wrapper mode augments the chromosome with a binary feature mask z and uses a composite objective (Equation (12)) to favor accurate, sparse, and fast models.

In the lightweight IDS, an ANOVA is applied as a filter-based feature selection method to retain the top-k features (trained on the training split only and then applied to validation/test data). The genetic algorithm (GA) is not used for feature selection; instead, it performs bounded hyperparameter tuning for the lightweight classifier to balance detection quality under edge constraints. Chromosome encoding and bounds: Each chromosome represents a candidate hyperparameter vector θ, e.g.,

$$\theta = [n\_estimators,\ max\_depth]$$

with bounded ranges defined in advance. The GA's fitness is computed using stratified cross-validation and macro-averaged metrics (macro-F1) to reduce bias toward majority classes.

$$\theta = [\text{n\_estimators, max\_depth }] \in \mathbb{Z}\text{^}2,\ \text{n\_estimators} \in [50, 400]\_\Delta 10,\ \text{max\_depth} \in [2, 12]\_\Delta 1 \tag{8}$$

$$\theta\_j \leftarrow \min\{\max\{\theta\_j, L\_j\}, U\_j\} \tag{9}$$

$$\text{fit}(\theta) = (1/|F|)\cdot\sum_{(T,V)\in F}(1/|V|)\cdot\sum_{i\in V} 1(f(x_i;\theta) = y_i) \tag{10}$$

$$\theta\_\text{child} = [\theta\_\{1{:}k\}(p1)\|\theta\_\{k+1{:}d\}(p2)],\ \theta\_j\text{^child} \leftarrow \theta\_j\text{^child} + \delta \text{ with prob. p\_m else } \theta\_j\text{^child (then clamp)} \tag{11}$$

$$\text{fit}_\lambda(z,\theta) = \text{CVAcc}(X_train, sel\cdot z, \theta) - \lambda^1\cdot(K_z/K) - \lambda^2\cdot\frac{\widehat{\ell(z,\theta)} - \ell_min}{\ell_max - \ell_min} \tag{12}$$

3.2.3. Final Ensemble Model and Evaluation

After GA convergence to $\theta\star$, the ensemble is retrained on the full training set and evaluated once on the held-out test set. Overall accuracy (Equation (13)), the confusion matrix, and per-class precision, recall, and F1 (Equation (14)) are reported. In addition to accuracy and per-class precision/recall/F1, we report balanced accuracy, Matthews correlation coefficient (MCC), and false alarm rate/false positive rate (FAR/FPR) on the held-out test set. For the multi-class setting, FAR/FPR is computed in a one-vs.-rest manner and macro-averaged across classes. Latency is measured using Equation (1), and dispersion is summarized by the sample standard deviation (Equation (15)). For depth-bounded trees,

the prediction cost scales approximately as shown in Equation (16), justifying the use of bounded integer hyperparameters for edge deployment.

$$\mathrm{Acc} = \frac{1}{\mathrm{n_{test}}} \cdot \sum_{i \in I_t est} 1(f(x_i; \theta\star) = y_i) \tag{13}$$

$$\begin{gathered}\text{Prec\_c = C\_\{cc\}/(}\textstyle\sum\text{\_b C\_\{bc\} + }\varepsilon\text{), Rec\_c = C\_\{cc\}/(}\textstyle\sum\text{\_b C\_\{cb\} + }\varepsilon\text{), F1\_c =} \\ \text{2·Prec\_c·Rec\_c/(Prec\_c + Rec\_c + }\varepsilon\text{)}\end{gathered} \tag{14}$$

$$\mathrm{s}_\ell = \sqrt{\frac{1}{\mathrm{N}-1} \cdot \sum_{i=1}^{N} \left(\ell_i - \overline{\ell}\right)^2} \tag{15}$$

$$\text{T\_infer} \approx \text{O(n\_estimators} \cdot \text{max\_depth)} \tag{16}$$

To handle class imbalance in GA optimization, our dataset is long-tailed (normal/Slowloris dominate relative to DoS/DDoS). Therefore, the GA fitness function is defined to be imbalance-aware by optimizing macro-F1 (computed across classes) under stratified cross-validation, rather than optimizing raw accuracy alone. This discourages solutions that overfit to majority classes and promote balanced detection quality across minority attacks. The GA's fitness for a candidate hyperparameter set θ is computed as follows:

$$Fitness(\theta) = (1/K) sum_{(k=1)}^{K} MacroF1_k(\theta) \tag{17}$$

where K is the number of stratified CV folds.

Optionally, we report latency separately and do not include it in the GA's fitness unless explicitly stated; thus, the fitness focuses on balanced classification quality, while latency is assessed during deployment profiling.

4. Results and Discussion

This section reports model performance on the held-out test set and discusses deployment-relevant behavior. Standard classification metrics such as accuracy, precision, recall, and F1-score are presented; the confusion matrix is analyzed; and accuracy is compared against processing time relative to representative GA-based and deep/hybrid IDS baselines from the literature. In addition to accuracy and per-class precision/recall/F1, we report balanced accuracy, Matthews correlation coefficient (MCC), and false alarm rate/false positive rate (FAR/FPR) on the held-out test set. For the multi-class setting, FAR/FPR is computed in a one-vs.-rest manner and macro-averaged across classes.

4.1. Overall Performance on the Test Set

On regular desktops and other devices, the fixed-parameter boosted tree baseline achieves strong results, with 99.20% test accuracy and an end-to-end inference time of around 0.002 ms per sample. Desktop latency is reported under the same end-to-end definition (transform + predict) using the same evaluation split; extremely small values should be interpreted as microsecond-level timing on high-performance hardware. Its per-class precision and recall are uniformly high, at approximately 0.99. However, this baseline is designed for regular machines; it does not enforce hardware-aware hyperparameter bounds, and its latency is not validated on the IoT or other edge devices. As a result, despite the headline metrics, the baseline is not a reliable choice for Internet of Things (IoT) deployments, where CPU, memory, power, and thermal limits materially affect runtime behavior.

To make the baseline methodology compatible with IoT hardware, an edge-optimized variant was developed that retains the same leak-free preprocessing but replaces fixed settings with a genetic algorithm-based search over bounded, device-aware integer spaces

for the number of estimators and tree depth. Fitness is computed via stratified 3-fold cross-validation on the training set, with early stopping and a time budget to prevent over-searching. Evaluated on Raspberry Pi-class hardware, this GA-tuned ensemble achieves 98.9% accuracy with an average on-device inference latency of around 4.3 ms per sample, reflecting robust generalization under the device's resource constraints.

Class-level behavior aligns with attack mechanics: DDoS and high-volume DoS yield the strongest precision and recall due to pronounced shifts in CPU and network counters. Slowloris shows slightly lower recall because of its "low-and-slow" pattern, and low-volume DoS occasionally overlaps with normal over short horizons. However, macro-averaged precision, recall, and F1-score remain high because of ANOVA filtering and GA-bounded model size.

Because the class distribution is long-tailed, we report macro-averaged metrics and optimize GA fitness using macro-F1 under stratified CV. This ensures that improvements reflect performance on minority attacks (DoS/DDoS) rather than gains driven primarily by the majority classes.

Table 4 presents the precision, recall, and F1-score for each class in the test set. The model demonstrates balanced detection performance across all attack types, with slightly lower recall for low-volume DoS and Slowloris due to their subtle, low-rate behavior. This confirms that while the IDS is robust overall, temporal dependencies could further improve sensitivity to slow attacks. Also, Table 4 compares the proposed GA-optimized IDS against a baseline model trained on the same dataset using identical preprocessing and feature selection but without genetic algorithm optimization. The baseline achieves slightly higher overall accuracy, 99.20, compared to the GA-optimized model, 98.85. This difference is expected, as the baseline is configured to maximize accuracy without considering deployment constraints.

Table 4. Precision, recall, and F1-score results across normal and attack classes.

Class	Precision	Recall	F1-Score
Normal	0.99	0.98	0.985
DoS (Low Volume)	0.97	0.95	0.96
DoS (High Volume)	0.99	0.99	0.99
Slowloris	0.96	0.94	0.95

These results indicate that while the IDS is robust across all attack categories, temporal dependencies could further improve sensitivity to low-rate attacks such as Slowloris.

As shown in Figure 6, residual confusion between normal and low-volume DoS reflects short-term temporal overlap in CPU and network usage. This suggests that incorporating temporal features or short-window history in future versions of the IDS could help distinguish between normal fluctuations and low-rate denial-of-service behavior. In contrast, high-volume DoS/DDoS attacks show clear separation, and Slowloris misclassifications occur only during transition windows, confirming the model's overall robustness across multiple attack intensities.

Figures 7 and 8 display ROC and PR curves computed from probability outputs on the held-out evaluation split. We acknowledge that very high AUC/AP values can be a red flag in IDS studies; therefore, we explicitly enforce leakage controls in the evaluation pipeline. Specifically, timestamps are used only for alignment/labeling and are never included as model inputs, and all preprocessing steps (imputation, scaling, and ANOVA feature selection) are fitted on the training split only and then applied to the test split. In addition to the standard stratified split, we include a time-blocked evaluation (train on

earlier days, test on later days) to reduce temporal correlation artifacts. We note that strong separability can occur in controlled lab deployments because the dataset includes direct system/network telemetry (e.g., CPU and network counters) recorded on-device during scripted attack windows; nonetheless, the time-blocked results provide a more conservative estimate of generalization across days.

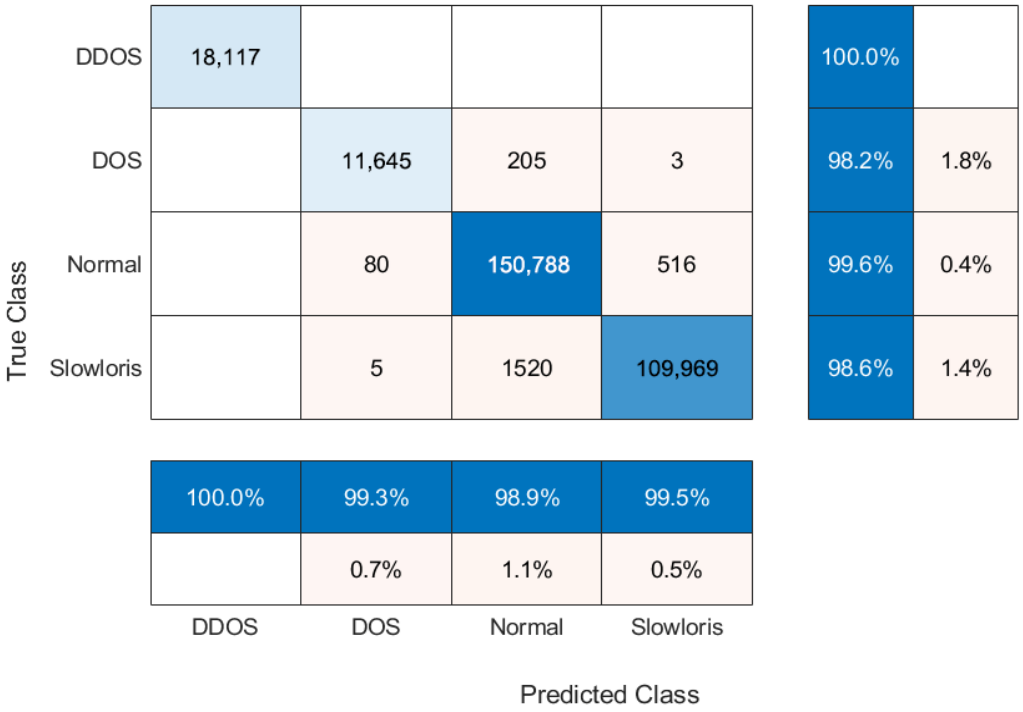

Figure 6. Confusion matrix.

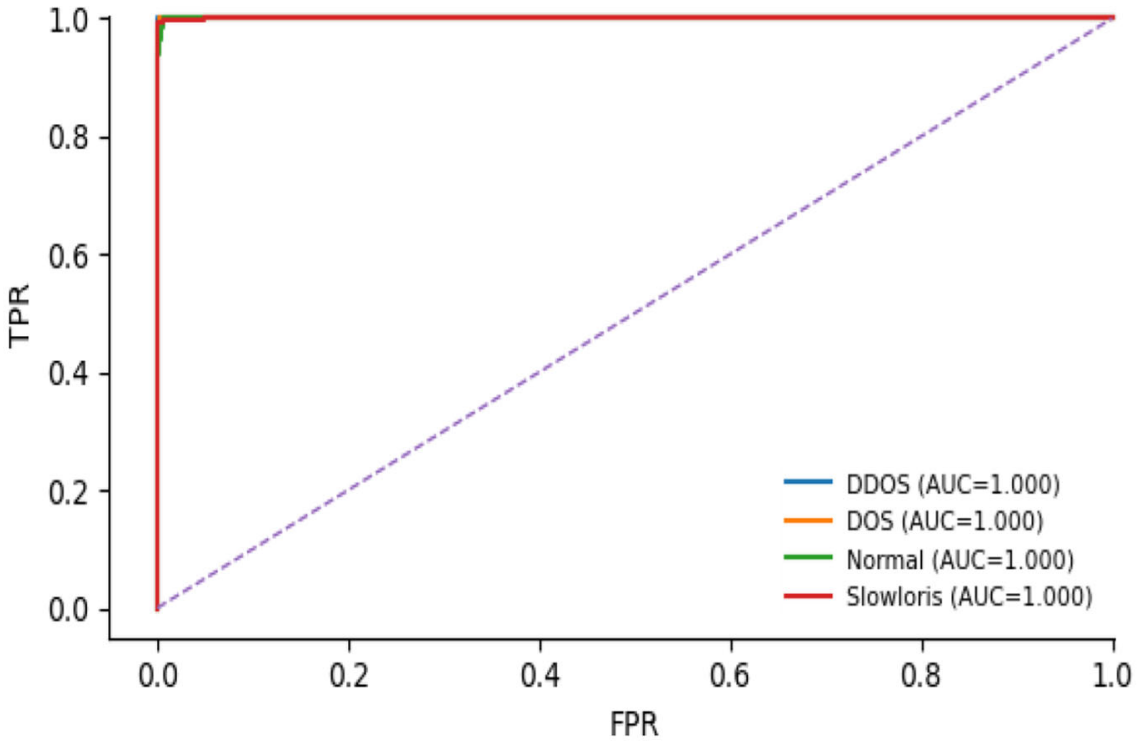

Figure 7. ROC curve.

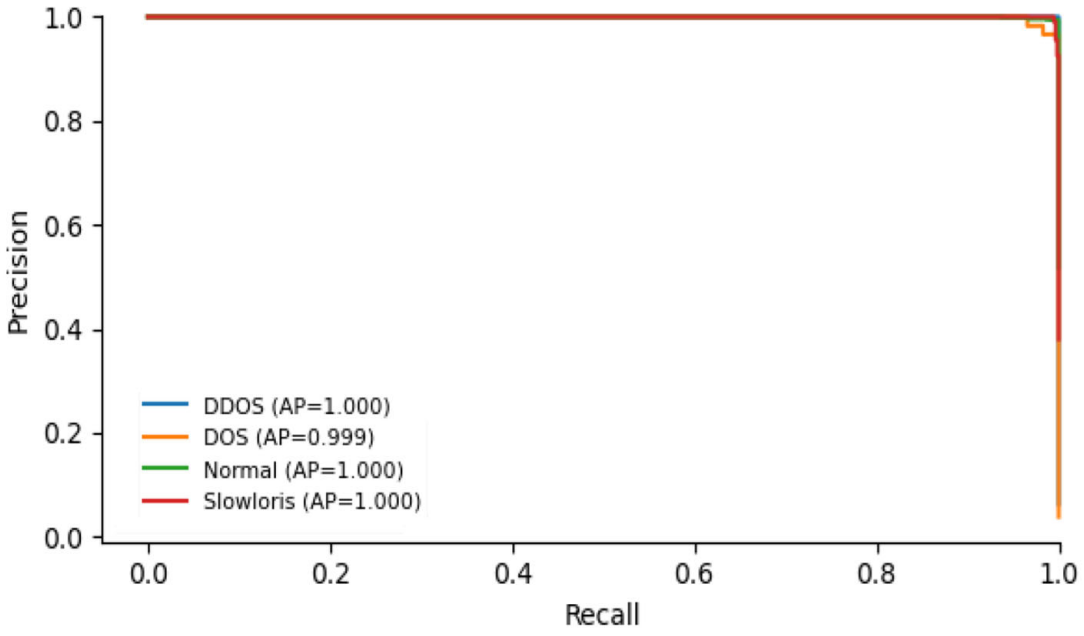

Figure 8. PR curve.

Figure 9 shows how the training and cross-validation curves track closely as the sample size increases, indicating that the selected capacity (number of estimators and tree depth) is well matched to the data scale; there is no sign of severe underfitting or overfitting.

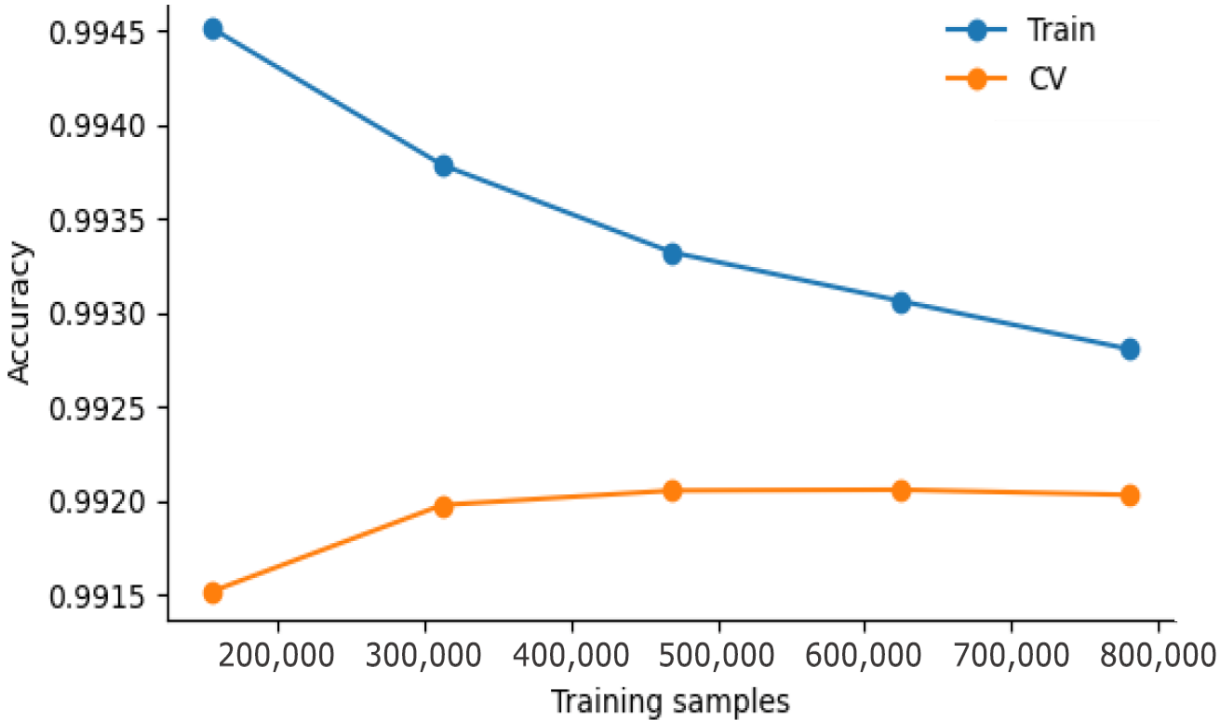

Figure 9. Learning curve (CV = 3, 5 points).

The histogram of per-sample latencies in Figure 10 shows a tight concentration around the mean with a narrow tail, satisfying sub-10 ms edge budgets by a wide margin. Reporting the full distribution, rather than a single mean, demonstrates the predictable behavior necessary for real-time controllers.

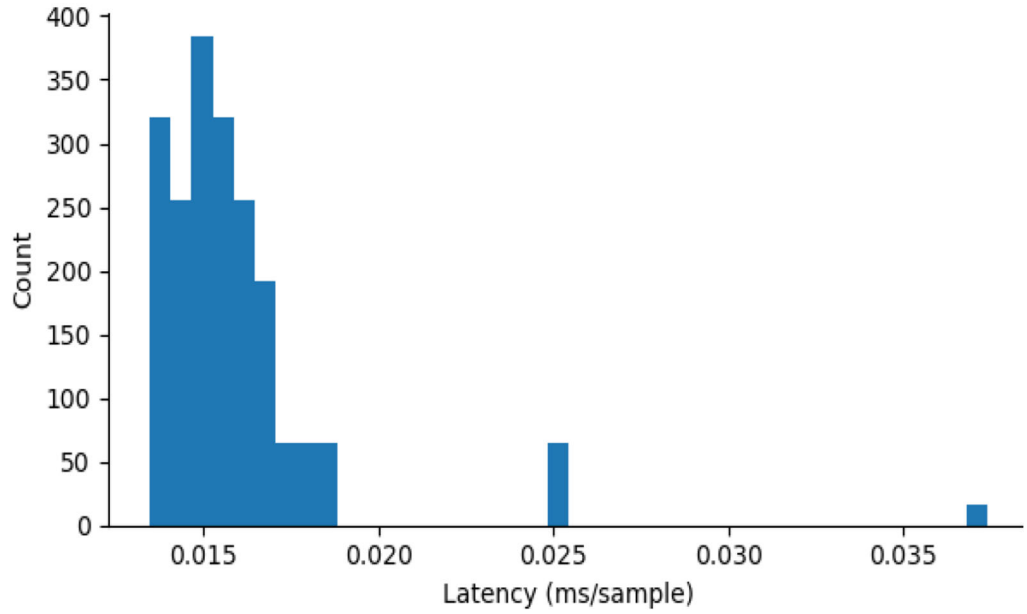

Figure 10. Latency distribution.

4.2. Feature Relevance and Interpretability

The train-only ANOVA highlights a compact subset of features with dominant discriminatory power, as shown in Figure 11. Notably, CPU (%) and Net (Mbps) rank highly, as expected for networked attacks. However, Sound, Water, and Smoke should be interpreted as contextual/operational signals in our lab deployment rather than causal indicators of cyberattacks. In particular, low-and-slow attacks (e.g., Slowloris) do not physically trigger smoke or water sensors; instead, these channels may co-vary with experiment conditions (e.g., ambient noise, human activity, or correlated device/environment states during scripted attack windows). Therefore, we treat them as potentially confounded features and validate their contribution through a dedicated ablation analysis.

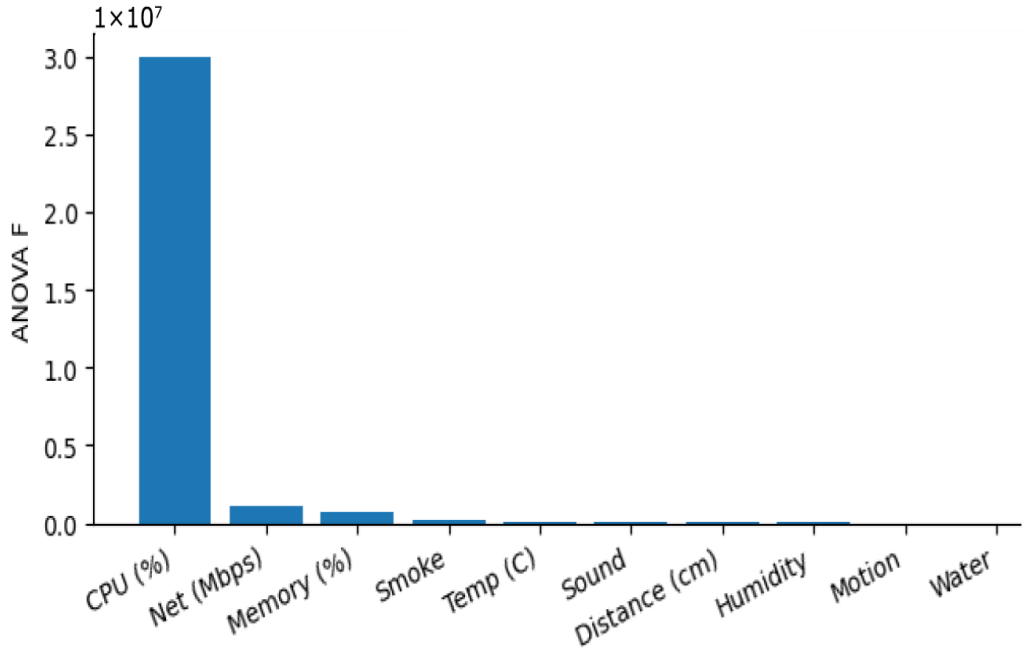

Figure 11. Top-K features using ANOVA.

In our dataset, physical sensors are synchronized with system and network telemetry at the edge node and are included to reflect the mixed-modal operational context typical

of IoT deployments. While this can enrich detection, it also introduces the risk of artifacts (e.g., environmental fluctuations correlated with attack execution timing). To mitigate this, the timestamp is used only for alignment and labeling and is not used as an input feature. Moreover, all preprocessing steps and ANOVA feature selection are fit strictly on the training split and then applied to the held-out test split. We further quantify the impact of physical channels via ablation experiments reported in Table 5.

Table 5. Ablation study results across feature groups (cyber/system vs. physical vs. mixed-modal).

Feature Set	Features Included	Accuracy	Marco F1	Marco AUC	Notes
(A) Cyber + system only	CPU (%), Memory (%), Net (Mbps)	0.9381	0.9447	0.9920	Tests whether cyber telemetry alone explains separability
(B) Physical sensors only	Sound, Smoke, Water, Temp (°C), Distance (cm), Humidity, IR, Motion	0.8202	0.5372	0.9445	Checks confounding/artifact risk
(C) Mixed-modal (full)	System/Network Telemetry + Physical Sensors	0.9616	0.9642	0.9955	Expected best if physical context adds useful signal

Post hoc permutation importance on a sampled test subset supports the ANOVA ranking and indicates that the ensemble depends on a small set of features, as shown in Figure 12. The alignment between the filter (ANOVA) and the model-based permutation ranking reinforces the argument for using the ten-feature configuration in edge scenarios.

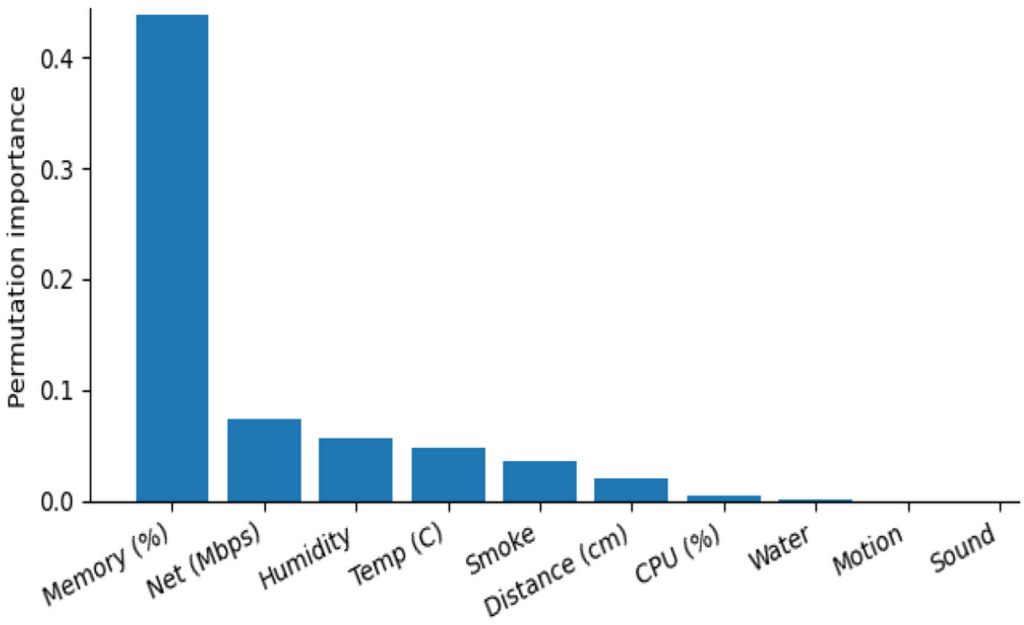

Figure 12. Permutation importance.

Because some contextual channels may correlate with experimental timing, we report the time-blocked evaluation results in addition to the stratified split to reduce temporal correlation artifacts and to provide a more conservative estimate of generalization.

4.3. GA Tuning and Stability

The GA employs a focused search over (n_estimators, max_depth) using 3-fold stratified CV to measure performance. Within a 15 min timeframe, the GA settles on n_estimators

= 110 and max_depth = 7, achieving a CV accuracy of about 0.992. It saturates within a few generations, suggesting a consistent and hardware-aware optimization landscape instead of a fragile, overly complex search. The convergence of the GA is documented in notebook logs, showing its progress generation by generation. Ultimately, the test accuracy remains around 99.2% whether or not the GA-selected hyperparameters are used, indicating a flat optimum. This flatness is advantageous for real-world deployment, as it suggests that the model can handle minor changes in configuration without a significant drop in performance.

Figure 13 shows that accuracy remains high for K ∈ {5,10,15,20}, highlighting that a small feature budget is sufficient. This effectively leads to reduced memory bandwidth and improved real-time predictability on microcontrollers and gateways.

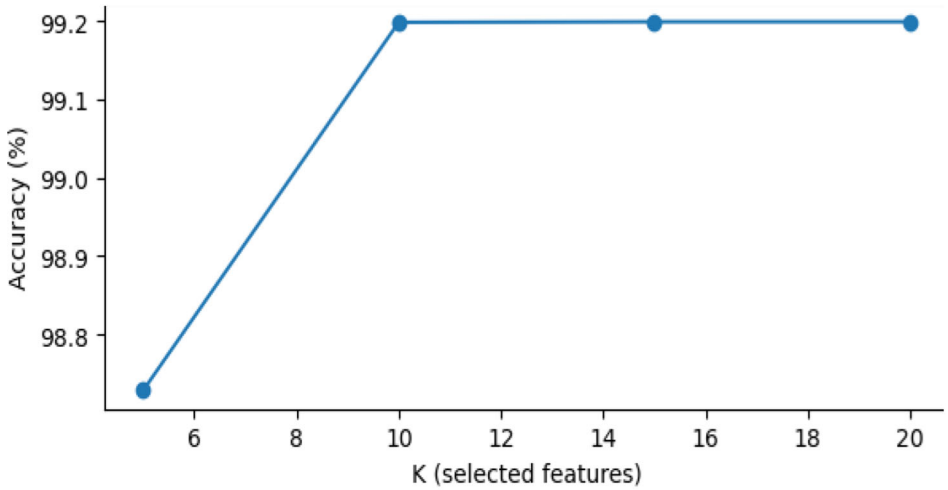

Figure 13. Sensitivity to number of features.

Table 6 summarizes on-device runtime profiling. The proposed IDS achieves 4.3 ms/sample while maintaining moderate resource usage (CPU: 29.29% avg, 91.5% peak; memory: 20.80% avg, 40.5% peak), which remains acceptable for Raspberry Pi-class edge deployment.

Table 6. Runtime profiling on edge device.

Metric	Value
Number of test samples	1000
Number of selected features	10
Inference latency (ms/sample)	4.3
CPU utilization (%)	29.29 (avg), 91.5 (peak)
Memory overhead (%)	20.80 (avg), 40.5 (peak)
Deployment platform	Raspberry Pi 5 (8 GB), Raspberry Pi OS 64-bit

Note: CPU and memory values reflect measured utilization during inference; they are reported as averages and peaks over inference window.

The ablation results confirm that physical sensors should be interpreted as contextual signals rather than direct physical effects of cyberattacks. Physical-only performance is comparatively weak, whereas combining physical context with cyber telemetry improves overall detection, supporting a mixed-modal but cautionary interpretation.

Table 7 reports additional IDS-relevant evaluation metrics on the held-out test set to complement accuracy-based reporting. In addition to macro-averaged precision, recall, and F1-score, we include balanced accuracy and the Matthews correlation coefficient (MCC),

which provide a more robust assessment under class imbalance and long-tailed attack distributions. We also report the false alarm rate/false positive rate (FAR/FPR) to quantify the tendency of the IDS to raise false alerts, an operationally critical aspect of intrusion detection. For the multi-class setting, FAR/FPR is computed in a one-vs.-rest manner and macro-averaged across classes. Overall, the results in Table 7 confirm that the proposed approach maintains strong detection performance while keeping false alarms low.

Table 7. Additional IDS-relevant evaluation metrics on the held-out test set.

Model	Balanced Accuracy	MCC	Macro-Precision	Macro-Recall	Macro-F1	FAR/FPR (Macro, OvR)
Baseline (No GA)	0.9715	0.9647	0.9896	0.9715	0.9801	0.0106
Proposed (GA-Optimized)	0.9698	0.9615	0.9884	0.9698	0.9787	0.0114
Grid Search	0.9680	0.9578	0.9869	0.9680	0.9770	0.0123
Random Search	0.9680	0.9578	0.9869	0.9680	0.9770	0.0123

5. Future Work

Future work will extend the proposed IDS toward scalable, collaborative, and interpretable edge deployments. On the learning side, federated learning will be integrated to enable site-specific model training without centralizing raw data while maintaining global coordination through secure aggregation. In addition, future work will include multi-site and multi-device deployments to evaluate cross-environment generalization beyond the current controlled setting. The optimization strategy will evolve into a multi-objective, deployment-oriented GA that jointly balances accuracy, latency, model size, feature count, and energy efficiency under strict real-time constraints. Explicit power and energy profiling on edge hardware will also be incorporated to complement the runtime latency, CPU, and memory measurements reported in this study.

The attack taxonomy will be expanded to include additional threats such as port scans, brute-force attempts, ARP/DNS spoofing, and data exfiltration, ensuring the broader coverage of both high- and low-volume attack behaviors. Future experiments will also evaluate the proposed framework on public IoT intrusion detection datasets to further assess generalizability. To sustain long-term performance, future iterations will incorporate online adaptation and drift detection to handle evolving network conditions and adversarial manipulation. Interpretability will also be enhanced by employing feature attribution techniques (e.g., SHAP values) and generating concise model cards that document GA provenance and data lineage.

Finally, the next phase will focus on the public release of the dataset and experimental scripts, supporting reproducibility and open collaboration. The framework will also be integrated into educational and research initiatives at the community college level to advance hands-on cybersecurity and IoT analytic training.

6. Conclusions

A lightweight, edge-ready IDS was presented, which was trained on a real, mixed-modal Raspberry Pi dataset. This system was fine-tuned using a unique, two-step approach: first, we selected the most relevant features using an ANOVA filter, and then we optimized the system's settings with a genetic algorithm that carefully adjusted parameters to fit within specific limits. When the final model was tested on a separate dataset, it achieved an impressive 98.9% accuracy, processing each sample in about 4.3 milliseconds. This means that it effectively identified various types of attacks, like low-volume DoS, high-

volume DoS/DDoS, and Slowloris attacks, with clear results shown in the confusion matrix and understandable errors during transition periods. This dual-optimized framework highlights that balancing accuracy and latency through GA tuning can transform IDS deployment on IoT devices from a research challenge into a practical, real-world solution.

Author Contributions: Conceptualization, K.H., K.B. and S.G. (Satinder Gill); methodology, K.H., K.B. and S.G. (Satinder Gill); software, K.H. and S.G. (Satinder Gill); investigation, K.H. and S.G. (Satinder Gill); resources, K.B.; data curation, K.H., K.B., S.G. (Satinder Gill), S.G. (Salah Garada), D.A.R., H.M., J.M.-K., J.M.-L., M.I.A.-V. and S.A.S.S.; writing—original draft preparation, K.H. and S.G. (Satinder Gill); writing—review and editing, K.H., K.B., S.G. (Satinder Gill), S.G. (Sarah Garada), D.A.R., H.M., J.M.-K., J.M.-L., M.I.A.-V. and S.A.S.S.; project administration, K.H. and S.G. (Satinder Gill); funding acquisition, K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Banaamah, A.M.; Ahmad, I. Intrusion Detection in IoT Using Deep Learning. *Sensors* **2022**, *22*, 8417. [CrossRef] [PubMed]
2. Bouzeraib, W.; Ghenai, A.; Zeghib, N. Enhancing IoT Intrusion Detection Systems Through Horizontal Federated Learning and Optimized WGAN-GP. *IEEE Access* **2025**, *13*, 45059–45076. [CrossRef]
3. Zhang, Y.; Li, P.; Wang, X. Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network. *IEEE Access* **2019**, *7*, 31711–31722. [CrossRef]
4. Saiyed, M.F.; Al-Anbagi, I. Optimized Ensemble Model with Genetic Algorithm for DDoS Attack Detection in IoT Networks. In *Proceedings of the 2024 IEEE International Conference on Communications Workshops (ICC Workshops), Denver, CO, USA, 9–13 June 2024*; IEEE: New York, NY, USA, 2024; pp. 433–438.
5. Jablaoui, R.; Cheikhrouhou, O.; Hamdi, M.; Liouane, N. Deep Learning Enabled Intrusion Detection System for IoT Security. *EURASIP J. Wirel. Commun. Netw.* **2025**, *2025*, 66. [CrossRef]
6. Gautam, D.; Bhadauria, S.; Trivedi, A. Malware Analysis Using Modified Genetic Algorithm in Cyber-Physical Systems. In *Proceedings of the 2022 IEEE 6th Conference on Information and Communication Technology (CICT), Gwalior, India, 18–20 November 2022*; IEEE: New York, NY, USA, 2022; pp. 1–5.
7. Li, J.; Chen, H.; Othman, M.S.; Salim, N.; Yusuf, L.M.; Kumaran, S.R. NFIoT-GATE-DTL IDS: Genetic Algorithm-Tuned Ensemble of Deep Transfer Learning for NetFlow-Based Intrusion Detection System for Internet of Things. *Eng. Appl. Artif. Intell.* **2025**, *143*, 110046. [CrossRef]
8. Swetha, R.; Senthilkumar, S. Genetic Algorithms-Based Feature Selection (GAFS-IDS) for Attack Detection in the Internet of Medical Things Network. In *Proceedings of the 2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0, Raigarh, India, 5–7 June 2024*; IEEE: New York, NY, USA, 2024; pp. 1–6.
9. Adegbite, R.; Kayode, A.A.; Olaniyan, O.O.; Arekete, S.A. Development of a Classification Model for Intrusion Attacks in Internet-Of-Things (IoT) Networks. *J. Sci. Technol.* **2025**, *30*, 1607–2073. [CrossRef]
10. Musthafa, M.B.; Huda, S.; Kodera, Y.; Ali, A.; Araki, S.; Mwaura, J.; Nogami, Y. Optimizing IoT Intrusion Detection Using Balanced Class Distribution, Feature Selection, and Ensemble Machine Learning Techniques. *Sensors* **2024**, *24*, 4293. [CrossRef] [PubMed]
11. Li, M.; Laiu, P.; Nichols, J.A.; Huettel, M.; Sikkema, I.; Mathur, M.; Hollifield, S.; Hankins, M. Cognitive IoT and Edge Computing for Intrusion Detection with Federated TinyML. In *Proceedings of the 2025 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 28–30 May 2025*; IEEE: New York, NY, USA, 2025; pp. 677–684.
12. Hasan, T.; Hossain, A.; Ansari, M.Q.; Syed, T.H. Enhanced Intrusion Detection in IIoT Networks: A Lightweight Approach with Autoencoder-Based Feature Learning. *arXiv* **2025**, arXiv:2501.15266.
13. Srivastava, A.; Sharma, H.S.; Rawat, R.; Garg, N. Detection of Cyber Attack in IoT Based Model Using ANN Model with Genetic Algorithm. In *Proceedings of the 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT), Greater Noida, India, 9–10 February 2024*; IEEE: New York, NY, USA, 2024; pp. 1198–1201.
14. Li, Z.; Shao, J.; Mao, Y.; Wang, J.H.; Zhang, J. Federated Learning with GAN-Based Data Synthesis for Non-IID Clients. In *Trustworthy Federated Learning*; Goebel, R., Yu, H., Faltings, B., Fan, L., Xiong, Z., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2023; Volume 13448, pp. 17–32, ISBN 978-3-031-28995-8.

15. Shahin, M.; Chen, F.F.; Hosseinzadeh, A.; Bouzary, H.; Rashidifar, R. A Deep Hybrid Learning Model for Detection of Cyber Attacks in Industrial IoT Devices. *Int. J. Adv. Manuf. Technol.* **2022**, *123*, 1973–1983. [CrossRef]
16. Kumar, D.; Pawar, P.P.; Addula, S.R.; Meesala, M.K.; Oni, O.; Cheema, Q.N.; Haq, A.U.; Sajja, G.S. AI-Powered Security for IoT Ecosystems: A Hybrid Deep Learning Approach to Anomaly Detection. *J. Cybersecur. Priv.* **2025**, *5*, 90. [CrossRef]
17. Mallidi, S.K.R.; Ramisetty, R.R. A Multi-Level Intrusion Detection System for Industrial IoT Using Bowerbird Courtship-Inspired Feature Selection and Hybrid Data Balancing. *Discov. Comput.* **2025**, *28*, 109. [CrossRef]
18. Javed, A.; Ehtsham, A.; Jawad, M.; Awais, M.N.; Qureshi, A.-H.; Larijani, H. Implementation of Lightweight Machine Learning-Based Intrusion Detection System on IoT Devices of Smart Homes. *Future Internet* **2024**, *16*, 200. [CrossRef]