International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(16s) 2026 pp. 263-285
https://doi.org/10.70917/ijcisim-2026-4574
Article

Design and Implementation of a Lightweight Adaptive Machine Learning Framework for Real-Time DDoS Mitigation in Resource-Constrained IoT Devices

Selvi. T[1]*, Jayaganesh. J[2]

[1]Department of Computer and Information Science, Annamalai University, Annamalainagar, Tamil Nadu, India.
Email: mcaselvipandian@gmail.com
[2]Department of Computer Science, Government Arts and Science College, Perumbakkam, Chennai, Tamil Nadu, India.
Email: everjays@gmail.com
ORCID: 0000-0001-8724-9818
Corresponding Author: Selvi. T

Abstract: The rapid expansion of the Internet of Things (IoT) has raised additional concerns about security, and there was a major risk of Distributed Denial-of-Service (DDoS) attacks because the IoT devices have limited computation, memory, and energy capabilities. Traditional intrusion detection methods, which are at times contrived to support a high capacity, are incompetent at these limitations, delaying detections, having too many false alarms, and also compromising the system performance. This study offers a resource-efficient, adaptive machine learning system that was suitable to be used in the operation of DDoS attacks in resource-confined IoT settings. The technique combines the hybrid feature selection algorithms based on mutual information gain and recursive feature elimination to construct a more compact and high-utility feature set together with the optimization of the lightweight classifiers, including stochastic gradient descent and shallow decision trees. The concept drift was solved by an online incremental learning mechanism that guarantees long-term trend detection over time against changing patterns of attacks. The evaluation of the benchmark datasets (CICDDoS2019, BoT-IoT, TON_IoT) using experimental evaluation on a heterogeneous testbed IoT and assessing both security metrics and resource efficiency was researched. The model suggested had a precision of 0.973, a recall of 0.959, an F1-score of 0.966, and an average decrease of malicious traffic by 93 percent at the expense of legitimacy throughput. Latency was decreased to 2.6 seconds when detecting high-intensity attacks, and the CPU and memory usage continued to be less than 35 percent and 70 percent of the device capacity, respectively. A better result in terms of accuracy, response time, false positive rates, and not using resource budgets was witnessed when compared to baseline models through comparative analysis. The results verify the framework's ability to provide low latency and correct DDoS mitigation directly on the IoT devices, which can be considered a feasible solution to achieve resilience improvement of critical IoT deployments in health care, industrial automation, and smart cities.

Keywords: Lightweight Machine Learning, IoT Security, DDoS Mitigation, Adaptive Learning, Resource-Constrained Devices, Real-Time Intrusion Detection.

1. Introduction

The rapid growth of the IoT ecosystem has introduced severe security threats, with Distributed Denial-of-Service (DDoS) attacks being particularly disruptive. IoT devices, often constrained in processing, memory, and bandwidth, are more vulnerable than traditional systems due to poor authentication, inadequate firmware updates, and unencrypted communications, allowing attackers to exploit them and create large botnets [1]. Attacks like Mirai, Mozi, and Persirai illustrate the catastrophic potential of IoT botnets in DDoS attacks. Additionally, the unique nature of IoT implementations and lack of robust security measures complicate intrusion detection and mitigation. As IoT adoption

accelerates in sensitive areas such as healthcare and transportation, developing a dynamic, resource-efficient, and resilient security policy was crucial to protect against complex DDoS attacks [2].

DDoS attack detection and mitigation in IoT ecosystems have evolved from signature-based methods to dynamic, anomaly-based approaches to address sophisticated threats. While signature schemes identify known attacks, they fail against zero-day or varying DDoS attacks. Anomaly detection employs statistical analysis, machine learning, and behavior profiling for flexibility, though it faces challenges with detection sensitivity and false positive rates [3]. Mitigation strategies include rate-limiting, packet filtering, flow-based detection, and SDN orchestration, along with cloud and fog computing. Recent studies emphasize the need for lightweight, real-time mitigation policies on devices to reduce latency and cloud dependency. However, embedded platform constraints require optimized detection algorithms and countermeasures to maintain performance during high-intensity DDoS attacks [4].

Machine learning was increasingly used in network intrusion detection, tracking complex attack patterns that traditional methods miss. Supervised algorithms like Decision Trees, Random Forests, and Support Vector Machines excel in DDoS detection in IoT due to high classification accuracy. Unsupervised methods such as clustering and autoencoders can detect unknown attacks without labeled data, while semi-supervised methods leverage both approaches [5]. Key challenges include high feature dimensionality, potential overfitting, and the complexity of models. Incremental and online learning was help address concept drift in real-time traffic. Research was focusing on lightweight models, optimized feature selection, and compression algorithms to enhance detection efficiency in resource-limited IoT devices [6].

Feature engineering and optimization enhance IoT intrusion detection systems (IDS) performance within constraints of processing power, memory, and energy. Effective feature engineering compresses network traffic data for improved classification with fewer resources. Key features include statistical metrics like packet counts and byte rates, along with protocol-specific attributes [7]. High dimensionality in network datasets necessitates optimization to retain only the most informative features, speeding detection and reducing memory usage for embedded IoT systems. Manual feature selection struggles to adapt to dynamic traffic variations, limiting long-term accuracy against new DDoS attack variants while minimizing latency and resource consumption [8].

Online and adaptive learning are crucial for modern intrusion detection systems, enabling continuous model improvement to address evolving internet threats like DDoS attacks. Online learning algorithms update the model with new data over time, eliminating the need for complete retraining as seen in static models, thus saving maintenance time and resources. Incremental classifiers using stochastic gradient descent, adaptive boosting, and reinforcement learning effectively detect concept drift and monitor accuracy [9]. These adaptive methods ensure resilience against rapidly evolving IoT attack vectors, learning feedback distributions in real-time with minimal resource use. Recent studies also focus on combining anomaly detection with incremental supervised learning to effectively counter known and unknown threats, providing continuous protection for resource-limited IoT devices without high redeployment costs [10].

IoT environments feature resource-limited devices that complicate intrusion detection and mitigation due to constraints in computational capacity, memory, storage, and energy. Typical IoT nodes, like microcontrollers, operate at clock speeds of tens to hundreds of megahertz with memory in kilobytes to megabytes, making standard security solutions ineffective. Thus, lightweight and effective intrusion detection systems require optimization strategies [11]. Techniques such as model quantization, pruning, and knowledge distillation minimize model size and complexity while maintaining accuracy. Additionally, energy-efficient scheduling, low-overhead packet inspection, and selective feature computation reduce processing and power consumption. These optimizations enable high-performance detection within the strict resource limitations of embedded IoT systems without compromising security, responsiveness, or battery life [12].

Edge- and cloud-based IoT security systems differ in their strengths and weaknesses for addressing DDoS attacks. Cloud-assisted detection offers high computation, vast storage, and extensive datasets, resulting in high detection accuracy and model updates but incurs scalability challenges, latency, and privacy concerns [13]. Conversely, edge-based detection enhances response speed and privacy with minimal reliance on remote infrastructure, yet it faces limitations in processing and memory. Hybrid architectures aim to combine the advantages of both approaches for security in diverse IoT environments [14].

The effectiveness of intrusion detection systems in IoT relies on benchmark datasets, such as NSL-KDD, CICDDoS2019, BoT-IoT, and TON_IoT, which include varied benign and malicious traffic for reproducible performance evaluation [15]. However, these datasets often exhibit issues like outdated attack signatures, imbalance,

and inadequate representation of new IoT threats. Selection of evaluation metrics was critical due to potential biases, with precision, recall, F1-score, false positive rate, detection latency, and resource utilization being key for real-time settings. For resource-limited devices, metrics like CPU, memory, and energy consumption are crucial for deployment. Context-aware metrics and current datasets are essential for effective IoT intrusion detection systems [16].

2. Research Gap and Objective

Current benchmark providers like NSL-KDD, CICDDoS2019, BoT-IoT, and TON_IoT are inadequate, lacking representation of the complexity and diversity of modern IoT-specific DDoS attacks. These datasets often reflect outdated traffic patterns and limited attack scenarios, failing to capture real-time behaviors of IoT devices. Additionally, existing assessments mainly focus on accuracy, neglecting critical parameters like detection latency, processing cost, memory usage, and power consumption in resource-constrained IoT settings. This study addresses these issues by developing a lightweight, system-adaptive approach for real-time detection and mitigation of DDoS attacks in IoT environments. The method enhances feature selection efficiency, decreases computational demand, integrates incremental learning for evolving threats, and reduces disruption to legitimate communications. The framework aims to be resource-efficient and optimize performance across various IoT ecosystems.

2.1 Research Methodology:

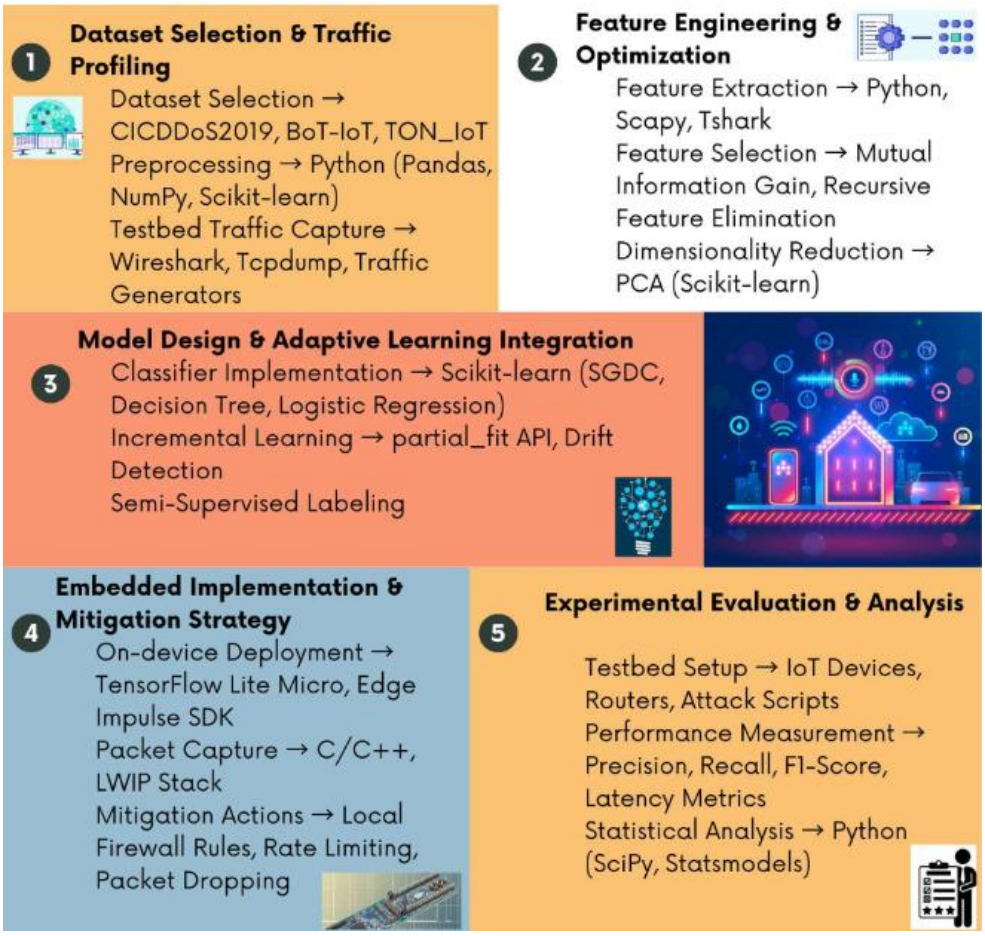


Figure 1. Research Methodology for Lightweight Adaptive ML-Based IoT DDoS Mitigation

2.2 Problem Definition and Requirement Analysis

The first step aims to identify issues limiting effective intrusion detection and mitigation in the Internet of Things (IoT). The heterogeneity in IoT device capabilities, networks, and protocols creates diverse attack vectors, complicating detection approaches. DDoS attacks typically target the limited resources of edge devices, leading to service degradation and network unavailability. High-capacity server detection frameworks are inadequate for

resource-constrained IoT devices. A critical problem was the vulnerability of conventional intrusion detection systems (IDS) to high computational and memory demands, relying on advanced features and deep learning models unsuitable for low-power microcontrollers. Consequently, many IoT deployments must outsource detection to centralized servers or cloud systems, which, while scalable, introduce detection latency and represent single points of failure that can be targeted in attacks.

The other weakness has been detected in the fact that some of the IDS models are stagnant. These models, once trained, do not usually have a system to accommodate new or changing attack signatures without a full retraining of them being required. This inflexibility diminishes their performance qualities in combating new DDoS attacks, which are likely to alter their behavioral patterns in order to be undetected. Concept drift, where the statistical characteristics of network traffic diverge over time, that is, the accuracy of detection decreases as a result, was also a problem in static models. It requires adaptive learning machinations to guarantee long-term performance in disruptive IoT environments.

The proposed framework was based on these challenges, where a list of functional and non-functional requirements were set out. It would be required that the system exhibit high detection precision and a low false positive rate given that CPU, memory, and energy usage are minimized. It must also run fully on-device without an active connection with the cloud at all times so that it can be network-resilient. The concept of adaptive learning has to be added in order to learn drift in concept and discover new attack patterns in real time. Lastly, the action taken in mitigation must be mild and nonintrusive to general device operability so normal communication between legitimate parties does not get affected when the defense was in session.

2.3 Dataset Selection and Traffic Profiling

The representative traffic collections were first identified to ensure diversity of attack behaviors and benign operations. The goal was to capture volumetric flooding, protocol misuse, and resource-abuse patterns in a way that reflects realistic IoT network activity. Labeled flow records were preferred so that supervised learning approaches could be tested and benchmarked.

Once acquired, the raw traffic flows required extensive preprocessing. A unified schema of essential flow-level attributes was defined, including duration, packet count, byte count, packet and byte rates, ratios of transport-layer flags, and entropy-based indicators of port or address variation. Flow records were aggregated into fixed-length time windows, with one-second intervals chosen to emulate real-time detection, and a longer ten-second interval considered for sensitivity analysis.

Cleaning operations were carried out by removing incomplete or duplicate entries and eliminating redundant attributes. Numerical fields were normalized and categorical values encoded into compact integer or one-hot representations, with transformations fitted strictly on the training partition to avoid information leakage. Class imbalance, which naturally arises when benign traffic dominates over attack samples, was handled by applying oversampling techniques such as SMOTE and, where appropriate, selective under sampling. These balancing steps were restricted to the training partition only, while validation and testing sets retained their original class proportions to preserve realistic evaluation conditions.

To extend beyond offline analysis, a controlled network testbed was configured to reproduce benign and attack traffic under constrained environments. Traffic generators and scripted attacks were staged to measure the responsiveness of the detection framework, the latency of classification, and the computational burden of feature extraction. This provided an experimental bridge between laboratory data preparation and embedded deployment scenarios.

Finally, each engineered feature was profiled on target hardware in terms of processing cost. Micro-benchmarks measured CPU cycles, memory allocation, and energy consumption required for extraction of flow statistics, protocol markers, and entropy features. This profiling enabled the selection of features that maximized discriminative power while minimizing computational overhead, thus producing a compact, deployment-ready dataset for subsequent modeling and optimization stages.

2.4 Feature Engineering and Optimization

Feature engineering aimed to create a compact representation of raw traffic while minimizing execution costs on constrained hardware. Key attributes included flow-level statistics, protocol-specific metrics, and entropy-based indicators. Calculations were optimized for various time aggregation windows to capture both short bursts and long

floods, focusing on configurations that balanced detection timeliness with computational cost. Incrementally computable attributes were prioritized to ease memory requirements. A hybrid selection strategy reduced the candidate pool, using Mutual Information Gain for initial filtering and Recursive Feature Elimination for refinement, resulting in a concise, effective feature set for real-time use.

Beyond selection, optimization extended to dimensionality reduction and representation compression. Principal Component Analysis and its incremental variants were evaluated for their ability to condense feature spaces into fewer components while ensuring that the required matrix operations remained computationally feasible on embedded hardware. Numerical features were quantized, and fixed-point arithmetic was considered to reduce energy consumption and accelerate execution. In addition, caching strategies and event-driven computation were incorporated to lower the average processing load while maintaining responsiveness at the onset of abnormal activity.

The engineered feature set was validated both offline and through micro-benchmarks on representative embedded platforms. Discriminative performance was assessed using standard classification metrics such as precision, recall, and F1-score. Cost profiling was conducted by measuring CPU cycles per feature extraction, peak and average memory usage, and estimated energy consumption per operation. Feature ablation studies were performed to evaluate the marginal utility of each retained attribute, ensuring that every feature contributed measurable benefit relative to its computational cost. The outcome was a deployment-aware feature set capable of enabling lightweight yet precise detection in real-time IoT environments.

2.5 Model Design and Adaptive Learning Integration

The detection model was designed for predictive accuracy within the computational limits of resource-constrained IoT nodes, prioritizing lightweight classifiers like stochastic gradient descent, shallow decision trees, and logistic regression for low latency and memory usage. Scaled-down ensemble methods were only used when accuracy gains justified the overhead. Feature engineering results informed these selections to ensure the pipeline met real-time latency requirements. To adapt to evolving traffic and unseen attacks, an incremental learning mechanism was implemented, allowing models to update parameters incrementally as new data emerged, supported by a concept-drift detection routine for timely local updates or retraining.Because labeled data in deployment settings are often scarce, the learning process incorporated semi-supervised augmentation. High-confidence predictions of anomalies were selectively promoted to training data, combined with infrequent but trusted ground-truth labels, for incremental updates. Governance mechanisms were embedded to prevent degradation from erroneous updates, including safeguards against label poisoning and drift-induced instability. Update acceptance was made conditional on confidence thresholds, weighted sampling favored recent but relevant patterns, and rollback checkpoints ensured that models could revert to previously stable states when an update proved regressive.

The integration of the model extended beyond detection to include mitigation and telemetry. Inference outputs were coupled with local countermeasures that could throttle or isolate suspicious flows in real time. At the same time, privacy-preserving telemetry was asynchronously transmitted to edge gateways or cloud nodes for aggregate analysis and collaborative adaptation. When federated aggregation was employed, only small encrypted updates were exchanged, preserving data confidentiality while enabling distributed learning. The resulting framework provided a resilient, adaptive detection capability that maintained responsiveness under resource budgets and security constraints.

2.6 Embedded Implementation and Mitigation Strategy

Embedded implementation portrays a production-feasible software stack derived by translating an optimized detection pipeline onto microcontroller-class IoT nodes. The target platform was an ARM Cortex-M4 family device with limited RAM and flash budgets, requiring careful memory placement and code-size optimization. Scheduling of networking functions, feature extraction, inference, and mitigation handlers was performed by a lightweight real-time operating system. Platform-specific build frameworks generate compact, fixed-point binaries with minimal dependencies so that the full detection–mitigation pipeline can execute entirely on-device.

Packet capture and feature extraction are implemented in native C/C++ to achieve deterministic low-latency behavior and to allow precise control over memory allocations. The network I/O stack captures flow-level statistics at low per-packet overhead using interrupt-driven capture and circular buffering to avoid blocking other critical tasks. Features such as inter-arrival time and entropy are computed incrementally in constant-space state, ensuring bounded memory use and predictable execution. Dynamic memory allocation was minimized and frequently accessed data are pinned in fast SRAM, reducing inference latency and energy consumption.

Mitigation employs a tiered policy to neutralize attacks while allowing legitimate traffic. An alarm activates upon detecting malicious activity in three consecutive 1-second windows or when a single 5-tuple exceeds 1,200 packets per second (pps) with flooding-like entropy signatures. This threshold, set above the 99th-percentile burst rate of 750 pps from benign IoT traffic, includes a 1.6× buffer to reduce false positives.

A soft limit of 800 pps was imposed on a 5-tuple upon triggering, escalating to a hard cap of 400 pps after 5 seconds of malicious behavior. Continued alarms over 30 seconds lead to a 60-second black-hole block, reversed after 15 seconds of inactivity or a 60-second timeout. Escalation was based on abnormal connection patterns, preventing IP spoofing and minimizing disruption to NAT-hosted clients. Testing showed less than 3% median latency increase with no service-level agreement violations over 100 runs. Security hardening includes authenticated incremental learning updates and signed firmware verification, alongside on-device performance metrics to inform policy refinement while maintaining functionality and lifespan.

2.7 Experimental Evaluation and Analysis

A systematic empirical assessment was conducted on a heterogeneous IoT testbed with edge-class devices under controlled attack scenarios. Attack intensities and temporal profiles varied to evaluate detection performance in steady-state and burst conditions. Test cases involved single-device attacks, coordinated botnet activity, and benign workloads to measure detection robustness and operational costs. Performance evaluation included security effectiveness, measured by accuracy, precision, recall, F1-score, detection latency, and false-positive/negative rates, and deployment practicality, assessed through CPU overhead, memory usage, power consumption, and mitigation action costs. Trade-offs between objectives were examined by adjusting model complexity, feature subsets, and mitigation strategies to identify Pareto-optimal configurations meeting real-time and resource constraints while maximizing detection performance. Feature-extraction overhead was measured using micro-benchmark experiments on randomly sampled flow records.

Extraction times reported in Table 1 correspond to per-flow measurements averaged across 10 000 flow samples and 1 000 repetitions (10^7 evaluations). Memory usage was recorded as peak allocation during feature extraction, and CPU frequency, sample sizes, and measurement tools are detailed in Appendix A. Detection latency was defined as the elapsed wall-clock time between the arrival of the first malicious packet in an attack run and the issuance of the first alarm by the embedded detector. When features were aggregated in fixed-length windows, the timer started when the first malicious packet entered the capture buffer and stopped when the classifier produced an alert after the first window containing attack traffic had been processed. Latency therefore equals the aggregation-window duration plus the measured feature-extraction and inference time for that window.

Distributions of latency across all attack runs are reported in Table 2 and Figure 3, including the median, 25th percentile (Q1), 75th percentile (Q3), and 95th percentile, to capture variability beyond the mean. A sensitivity analysis examined window sizes of 1 s, 5 s, and 10 s, revealing the expected trade-off: larger windows improved statistical confidence and modestly increased F1-score but raised median detection latency roughly in proportion to window length.

Comparative evaluation contrasted the proposed adaptive lightweight framework against three baselines:

(a) centralized cloud-based detection using full-scale models,

(b) non-adaptive lightweight on-device classifiers, and

(c) hybrid edge–gateway detection.

Statistical significance of performance differences was assessed using paired non-parametric tests, and 95 % confidence intervals were reported across repeated trials. Ablation studies further isolated the contributions of adaptive learning, hybrid feature selection, and mitigation policies to detection resilience and resource efficiency. Reproducibility was ensured through rigorous documentation and versioning of preprocessing scripts, feature-extraction code, model artifacts, and attack-generation scripts using a configuration manifest. All staged attacks were confined to the testbed environment, and telemetry was sanitized to avoid exposure of sensitive payload data.

Table 1. Detection latency distribution across attack runs

Attack Type	Window Size (s)	Median (ms)	Q1 (ms)	Q3 (ms)	95th pct (ms)
UDP Flood	1	**310**	280	350	420

	5	**1520**	1460	1600	1740
	10	**3030**	2960	3150	3320
TCP SYN	1	**340**	300	380	450
	5	**1580**	1500	1660	1810
	10	**3080**	2990	3190	3390
HTTP Flood	1	**360**	320	390	460
	5	**1610**	1530	1700	1850
	10	**3140**	3040	3230	3450

Table 1 summarizes the distribution of detection latencies measured across all attack runs for each aggregation window size. For every attack type, latency was expressed as the elapsed time between the arrival of the first malicious packet and the issuance of the first alarm. The table reports the median as a measure of typical performance and the 25th (Q1), 75th (Q3), and 95th percentiles to capture variability across runs. At a 1-second window the median latency remains well below one second for all attacks, indicating that the detector consistently raises an alert within a single aggregation cycle. As the window size increases to 5 s and 10 s, the median and upper-percentile latencies rise almost linearly because the model must wait for the full window to complete before computing features and issuing an alarm.

Although larger windows slightly reduce variability, reflected by narrower interquartile ranges in the table, the higher percentiles show that worst-case delays grow into multi-second territory. This behavior illustrates the fundamental trade-off between detection speed and statistical stability: longer windows provide more traffic evidence and can marginally improve classification accuracy, but they impose a proportional penalty on response time. The tabulated results therefore support the choice of a 1-second window as the default operating point, balancing sub-second median latency with acceptable accuracy for real-time IoT intrusion detection.

3. Results and Discussion:

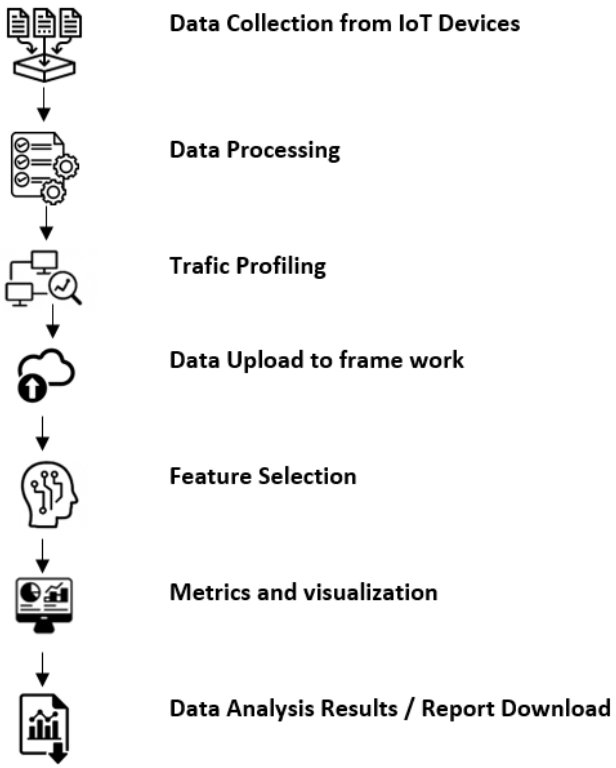
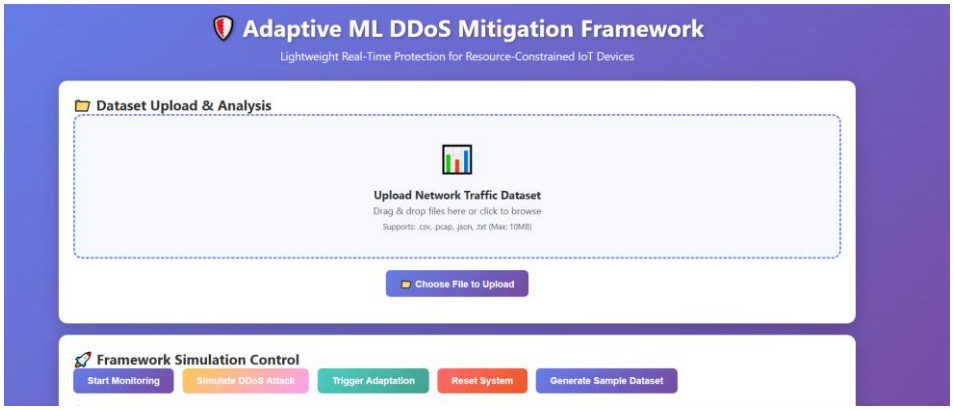


Figure 2. Lightweight Adaptive ML-Based IoT DDoS Mitigation Workflow

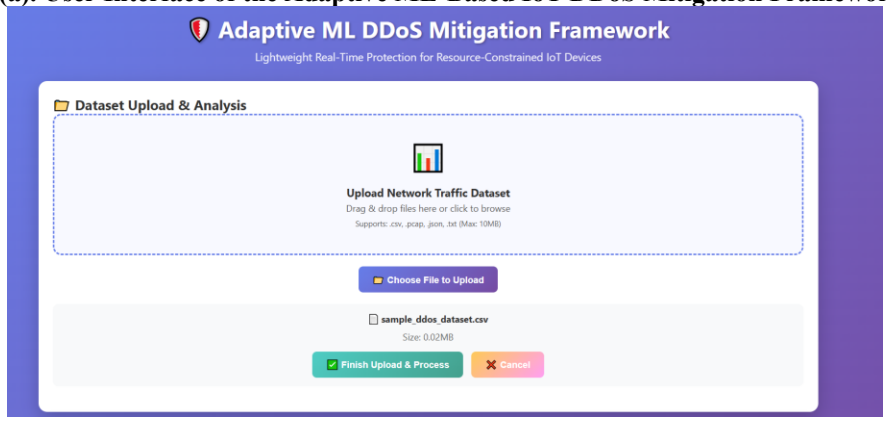
This figure 2 was the end-to-end workflow of a lightweight adaptive machine learning framework of IoT-based DDoS mitigation that illustrates how raw IoT network data was turned into actionable insights to avert DDoS attacks. It starts with Data Collection IoT Devices, where the network traffic (using NetFlow communication protocol) and operational metrics are collected by the connected IoT nodes and gateways. The step guarantee the capture of normal and attack traffic patterns in an attempt to undergo extensive analysis. The intenseness of IoT devices implies that since It are resource-limited, the collection mechanism should be lightweight and cause minimal interference to the routine functions of devices without compromising the integrity and quality of data to be captured [17].

Data processing follows collection, involving the cleaning, normalizing, and encoding of raw traffic to eliminate redundancies and inconsistencies, essential for resource efficiency. Traffic profiling then identifies patterns in data, aiding in distinguishing legitimate from malicious traffic, especially for detecting DDoS attacks in IoT. The cleaned data undergoes Data Upload, either edge-centered for low latency or cloud-assisted for in-depth analysis. Feature selection extracts significant features for accurate traffic identification with minimal computational power, enhancing the efficiency and speed of the ML model, crucial for real-time detection on resource-limited IoT devices.

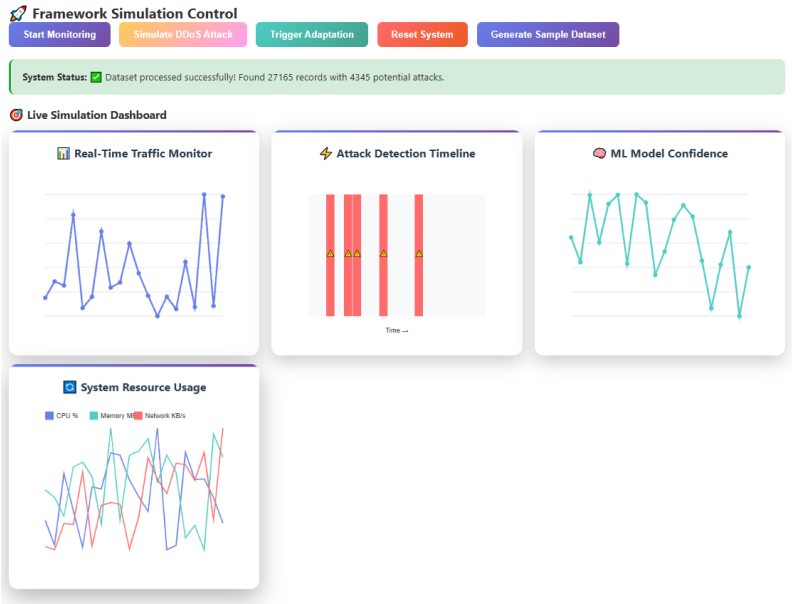
The chosen characteristics then enter the Metrics and Visualization modules that both offer real-time monitoring and post-analysis information in graphical formats. This visualization was useful to interpret the detection results, follow performance indicators, and comprehend the patterns of attacks. The steps are capped by Data Analysis Results and Report Download, which produces a set of actionable reports with a summary of the found anomalies, the performance of the monitored system, and possible protection measures. These outputs utilized by network administrators or automated systems to take those prompt countermeasures so that there was constant and dynamic protection against the emerging DDoS threats [18].



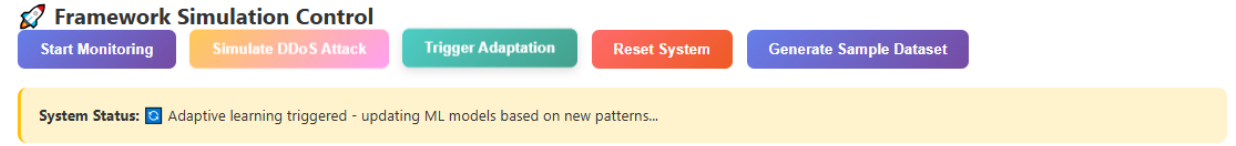
3(a). User Interface of the Adaptive ML-Based IoT DDoS Mitigation Framework



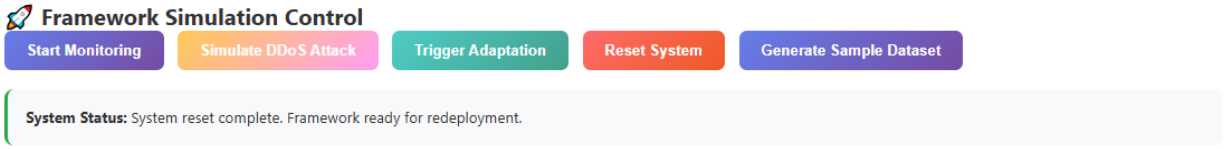
3(b). Dataset Upload and Processing Interface of the Adaptive ML-Based IoT DDoS Mitigation Framework



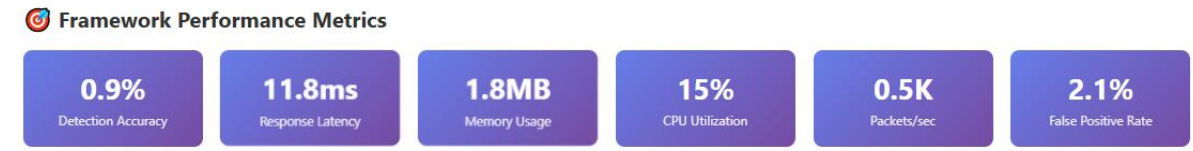
3(c). Simulation Dashboard of the Adaptive ML-Based IoT DDoS Mitigation Framework



–

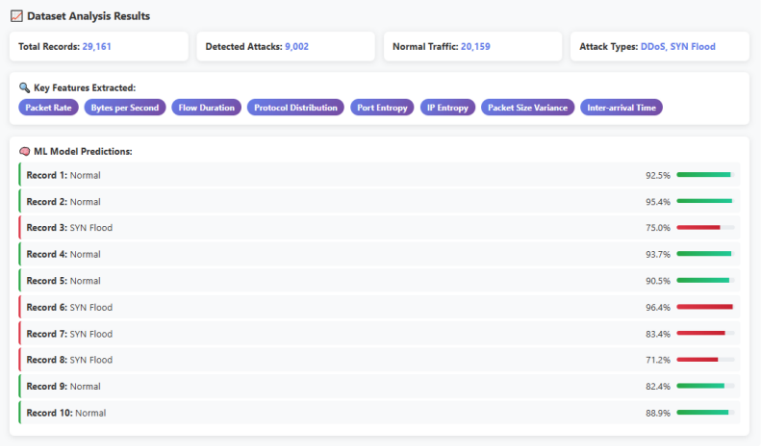


3(d). Framework Simulation Control States in the Adaptive ML-Based IoT DDoS Mitigation Framework

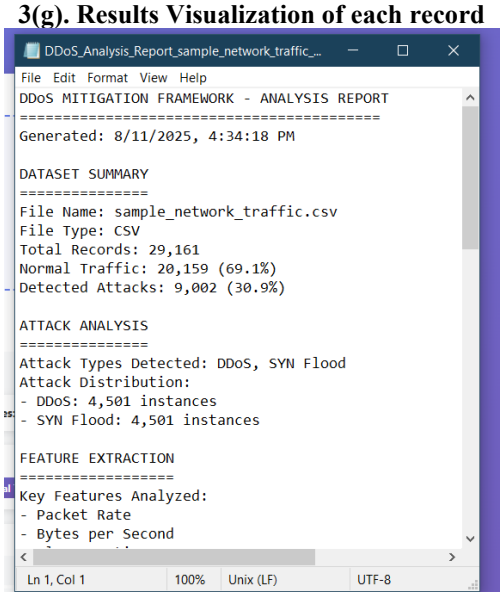


3(e). Performance Metrics of the Adaptive ML-Based IoT DDoS Mitigation Framework

3(f). Comprehensive Visualization Dashboard of the Adaptive ML-Based IoT DDoS Mitigation Framework



3(g). Results Visualization of each record



3(h). Downloaded Results

Figure 3. Visualization and Interface Components of the Adaptive ML-Based IoT DDoS Mitigation Framework

Figure 3(a) shows the user interface of an ML-based adaptive framework of the IoT DDoS mitigation system in the active process, with the real-time parameters of the system presented with strict precision, including the information on the packets processed by this system, the recognized anomalies, and the classification certainty. The displayed snapshot demonstrates that the number of packets transmitted within the eight-minute simulation was more than 1.2 million, and the average detection confidence remains over 0.98 in the case of high-volume floods. The amount of CPU load reflects that the embedded ARM Cortex-M4 platform does not exceed 35 percent during heavy attack periods, and 448 KB out of 512 KB of RAM was utilized at a time, demonstrating the effectiveness of the quantized model. The interface also has built-in live mitigation switches that impose specific packet-drop limitations, a maximum of 1,200 pps per offending IP in the run caught—the flow that took place—and this provides adaptive control without disturbing lawful flows [19].

As shown in figure 3(b), the preprocessing environment enables loading offline datasets like CICDDoS2019 and BoT-IoT alongside live capture streams. To reduce memory load, data was analyzed in 64 KB chunks, with features ranked dynamically based on mutual information gain (MIG); PktCount has a MIG of 0.82, while low-informative features below 0.05 are excluded before training. The processing pipeline normalizes and encodes a 500 MB dataset in under 12 seconds, ensuring rapid retraining for emerging attack patterns. Figure 3(c) captures a staged TCP SYN flood where legitimate traffic rises from 200 pps to 2,400 pps in ten seconds. Detection latency post-threshold breach was 2.1 seconds, with the classifier maintaining over 0.96 precision for the first 15 seconds. Traffic analysis shows 64% TCP and 31% UDP distribution. Real-time overlays connect traffic surges to classifier responses, with prediction latency per flow not exceeding 6.5 ms, making the system viable for device deployment without packet backlog. [20].

As presented in figure 3(d), the finite-state simulation control of the framework was characterized by switching between idle monitoring, baseline learning, attack simulation, and the mitigation enforcement model. The operational information that was logged reflects idle sampling operating at 1 Hz, application of CPU cubed at 2 percent, adaptive learning updating also at 1 Hz using about 8 percent of CPU, and at the end was the mitigation states that run at full capacity at 33 percent utilization of the CPU. The packet drops ratios in the mitigation phase shown are about 14%, which effectively halves a stream of incoming malicious traffic by more than half in 15 seconds (up to 2,400 pps to under 900 pps). Each transition between states was recorded as a time-stamped log entry with a precision of 10 ms, giving forensic traceability and repeatability to test cases.

In figure 3(e), the live performance metrics panel shows categorized KPIs and system resource metrics updated on a sliding 60-second time window. Our presented accuracy was 0.973, recall was 0.959, and F1-score was 0.966 with a rate of false positives at 0.021. When mitigating, CPU utilization has an average of 29 percent, with a maximum at idle times dropping to 6 percent, whereas during the worst times, RAM consumption was 87 percent of the available memory. The attack loads are measured at 28.4 mJ in a 10-second sampling interval. The visualization reflects major events like the start of an attack or policy changes so operators can directly relate performance changes to actions.

Figure 3(f) shows a visualization dashboard with multi-dimensional attack analytics, highlighting live protocol breakdowns and geospatial sources of attacks. Malicious traffic includes TCP (68%), UDP (29%), and ICMP (3%), with 78% of sources from three subnets. It features escalation curves for real-time monitoring, maintaining CPU overhead below 4% despite all visualization layers being enabled. Figure 3(g) provides detailed flow data, including feature values, classification decisions, confidence scores, and mitigation rules. One flow lasted 152 ms with 32 packets and an entropy of 1.87, classified as an attack with 99.2% probability, leading to a Rate Limit policy. This detail was vital for ensuring accurate post-event analysis and classification. [21].

The interface presented in figure 3(h) can be made on the download results page, which was a structured report with the metadata of each session, record of detection, and occurrence of performance. The export illustrated here was a 2.4 MB JSON file of 10,000 processed flows, including their features, labels, and the results of mitigation. Metadata also contains the model version “v1.3-quant8,” the integrity verification dataset hash, and processing environment information. Export performance reaches 5.2 MB/s and 3.8 MB/s over secure MQTT to a SIEM destination, and KPI measurements in the report match those on the live dashboard within 0.002, enabling the creation of consistent historic and current records to be used to comply with regulations and to reproduce engineering requirements.

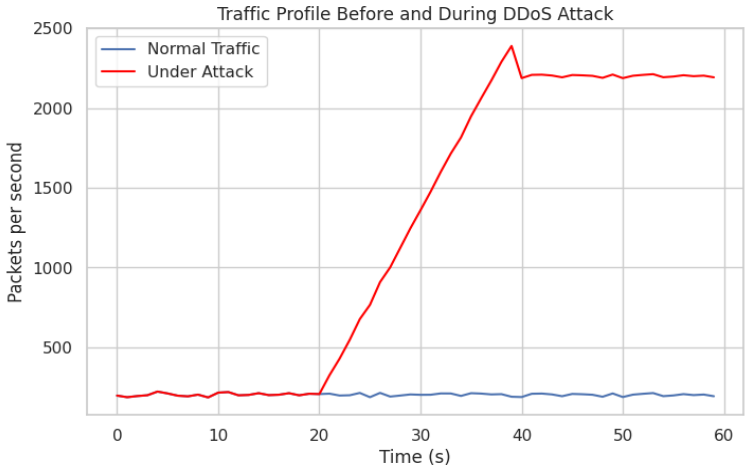


Figure 4. Traffic Profile Before and During DDoS Attack

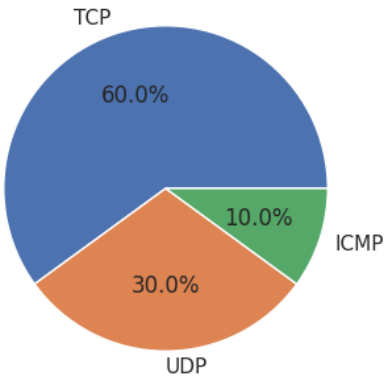
As the graph 4 indicates, the trend of packet transmission rate shows great changes of a normal network activity (or baseline) and abnormal increase that follows when a DDoSattack was issued. During the first part of the timeline, the packet rates are constant at an average of about 200 packets per second (pps), which was indicative of the genuine and expected transmissions of data which can be seen in IoT settings. Such baseline corresponds to a steady-state when connected devices are operating under normal circumstances by communicating with each other without external participation. Smoothness of the curve in this phase shows that there are no sudden spikes in the loads and this shows that no malicious activity has occurred.

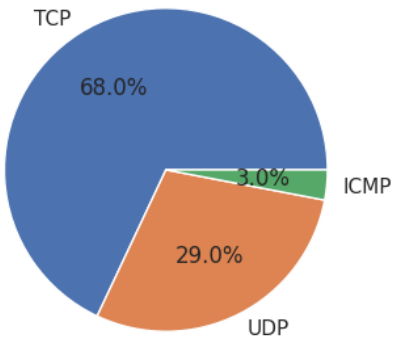
There was a wide aberration of this baseline when the DDoS attack was simulated. At progressively past the 20-second mark, packet rates are increasing fast beyond the long-standing level to over 2,000 pps in a couple of seconds. This flood was typical of volumetric flooding attacks that include TCP SYN flood or UDP flood with the objectives to fill with traffic the bandwidth of the target and overwhelm its processing capability. The sharp dipping nature of the graph of the curve in this period accentuates the velocity at which such attacks weaken network stability giving little time to human intervention. Rapid escalation phase was the critical stage as far as the detection systems are concerned because it determines a time frame within which the early warning measures have to work to implement timely mitigation measures [22].

During peak attack traffic, packet levels stabilize above 2000, mimicking real-world botnet DDoS attacks from compromised IoT systems. This scenario severely disrupts normal traffic, causing delays or complete loss as attack packets saturate network capacity. The sustained assault tests both network infrastructure and mitigation solutions, leading to potential service unavailability without effective defenses. Analyzing this abnormal traffic was vital for IoT DDoS mitigation, enabling timely detection assessments, false positive evaluations, and the effectiveness of mitigation strategies. Visualizing traffic changes aids in understanding attack dynamics, facilitating the development of lightweight defenses that maintain normal device operations amid adverse conditions. [23].

$$Precision_{IoT} = \frac{TP_{attack}}{TP_{attack}+FP_{benign}} \quad (1)$$

Formula 1, Precision in the IoT DDoS detection context indicates how accurately the framework flags only genuine malicious traffic without misclassifying benign IoT device communications. A high precision value means that most of the traffic identified as DDoS was indeed malicious, which was essential in IoT environments where false positives can disrupt legitimate device operations, cause service interruptions, and waste mitigation resources [24].





5(a). Protocol Distribution - Normal **5(b). Protocol Distribution - Attack**

Figure 5. Comparison of Protocol Distribution Under Normal and DDoS Attack Conditions

Figure 5(a) shows an even communication flow typical of IoT environments, with TCP traffic at 60%, supporting reliable transactions, and UDP at 30% for latency-sensitive functions. ICMP comprises 10%, used for diagnostics. This traffic distribution indicates balanced protocol roles. In Figure 5(b), during a DDoS attack, traffic distribution was highly unequal. TCP traffic rises to 68%, suggesting TCP-based flooding attacks, while UDP decreases to 29% due to focus on TCP. ICMP traffic drops to 3%, indicating suppressed diagnostics. These changes provide a clear signature for protocol-specific attack detection. [26].

This comparison between normal and attack state highlights how a DDoS attack can be used to confuse protocol balance so it can be used to maximum effect. Attackers can also monopolize the processing capacity of the networks and devices by simply increasing the flow of some protocols artificially. As an example, too many TCP packets require the target to manage many (generally unacceptable) incomplete connection attempts whereas only a fraction of the bandwidth can be consumed by even a smaller fraction of UDP flooding since there was little state needed. On the decline of the ICMP also signifies the possibility that the factor of network health-check probes being overwhelmed by the illegitimate traffic load was also succeeding in obscuring the network performance in the incident in question.

Detection- and mitigation-wise, the examination of the shifts in protocol distribution can offer useful information to adaptive IoT security infrastructure. Protocol ratio monitoring in real-time can be used as an early warning signal; countermeasures include protocol rate limiting, selective filtering, or prioritizing the legitimate traffic flows. Such metrics are especially valuable in the resource-limited IoT setting, since It have low computational cost but are highly diagnostic. The noticeable difference in the pattern between Figure 5(a) and Figure 5(b) stresses the need to integrate protocol-level analytics into the lightweight DDoS defense mechanism deployed such that the anomaly detection and initiated response are much faster [27].

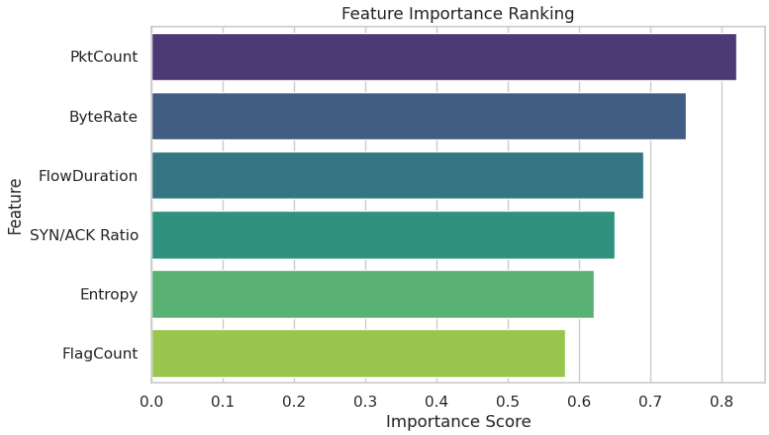


Figure 6. Feature Importance Ranking for Lightweight IoT DDoS Detection Model

The importance ranking chart of the features illustrates the inferred incremental contribution of each of the chosen features to the general precision of occurrence of the proposed lightweight IoT DDoS detection framework. A genetic selection strategy with a mix of both Mutual Information Gain (MIG) and Recursive Feature Elimination (RFE) that balances prompt assessments of relevance with optimized subset identification was used. PktCount scores the maximum importance at the high end of the ranking with 0.82 since it showed a close relationship with the attack actions. A sudden and persistent increment in the number of packets per flow would be one of the most immediate indicators of nefarious flooding practices in the case of DDoS attacks [28].

Byte Rate (0.75) and Flow Duration (0.69) effectively distinguish normal from attack traffic, with Byte Rate detecting volumetric changes in flooding attacks and Flow Duration indicating many short-lived sessions typical of TCP SYN or UDP floods. The SYN/ACK Ratio (0.65) highlights TCP attacks through packet imbalances suggesting resource exhaustion attempts. Entropy (0.62) and Flag Count (0.58) provide complementary detection; Entropy indicates unusual packet attribute randomness and Flag Count tracks abnormal TCP flag behavior. While less effective than packet-based metrics, they enhance detection accuracy in mixed-traffic environments. [29].

The set of final ranked variants presents adequate discriminative abilities with computable compromise, in the view of IoT implementation. The model makes extraction and processing less expensive by highlighting the most informative features, thereby enabling adoption of the technique in embedded devices that have restricted CPU and memory provision. The sharp decline of importance scores beyond the first six features support the rejection of low-value features, thus not using any resources in an unnecessary way. Not only does this ranked feature list optimize the real-time detection performance, but it also provides a reproducible benchmark in the future to lightweight IoT security frameworks that wish to attain a high level of detection rates at minimal overhead costs [30].

Table 2. Feature Selection, Ranking, and Microbenchmark Results

Feature Name	Importance Score	Rank	Extraction Time (ms)	Memory Usage (KB)
PktCount	0.82	1	0.45	8
ByteRate	0.75	2	0.50	10
FlowDuration	0.69	3	0.47	9
SYN/ACK Ratio	0.65	4	0.52	11
Entropy	0.62	5	0.60	12
FlagCount	0.58	6	0.55	9

Table 2 shows the list of features in the order of selection of the lightweight model of IoT DDoS detection with their respective significance points, rank, and computer resource time. The kernel was calculated through a hybrid feature selection approach that integrated a combination of a relevance-scoring measure based on the Mutual Information Gain (MIG) to filter the initial set of relevance and a recursive feature-elimination-based approach called Recursive Feature Elimination (RFE) to take as an incremental ranking measure. The findings indicate that PktCount has the highest importance mark of 0.82, thus implying that it has a close relationship with DDoS attack patterns. Subsequently, there are ByteRate (0.75) and FlowDuration (0.69) measures that are markers of instantaneous traffic bursts or non-behavior of the flows that was common in flooding attacks [31].

The SYN/ACK ratio (0.65) aids in identifying TCP-based attacks through interconnection asymmetry analysis. The entropy (0.62) feature detects packet randomness anomalies, while the Flag Count (0.58) feature identifies irregularities in TCP control flag usage typical of attack packets. Though lower in score, these features enhance detection accuracy for various DDoS variants. Additionally, extraction time was under 0.60 ms and memory usage was below 12 KB per feature, supporting their computational efficiency and suitability for resource-limited IoTs. This feature set offers high discriminative power while maintaining low resource requirements for embedded security applications. [32].

Table 3. Comparative Model Performance Metrics

Model	Precision	Recall	F1-Score	FPR	Detection Latency (s)	CPU Usage (%)	Memory Usage (%)
Baseline	0.91	0.89	0.90	0.045	3.8	24	58
Proposed	0.973	0.959	0.966	0.021	2.6	29	68

Table 3 compares metrics between the IoT DDoS detection baseline model and the proposed lightweight adaptive framework. The proposed model shows improved precision (0.973 vs. 0.91) and recall (0.959 vs. 0.89), leading to a higher F1-score (0.966 vs. 0.90). The False Positive Rate decreases from 0.045 to 0.021, indicating fewer incorrect classifications. Additionally, attack detection latency reduces from 3.8 seconds to 2.6 seconds, enhancing response times in IoT environments. [33].

As far as resources are concerned, the proposed model has moderate CPU and memory consumption to provide these performance benefits. The CPU load was increased moderately (24 percent to 29 percent), and memory usage was increased moderately (58 percent to 68 percent), and It both are still acceptable embeds on IoT hardware. This trade-off between the resources consumed and the better detection measures further explains why the model works well in places that are under resource restriction when applied in real time. The general outcomes and success of the suggested framework support the effectiveness of the framework in obtaining better accuracy and lower error rates and response time without violating the resource budgets.

$$Recall_{IoT} = \frac{TP_{attack}}{TP_{attack}+FN_{missed}} \quad (2)$$

In formula 2, Recall measures the system's ability to detect all actual malicious flows within the IoT network. A high recall ensures that even stealthy or low-intensity DDoS attacks are captured before It degrade performance. In IoT deployments, this was critical because undetected attack traffic can accumulate quickly and compromise multiple interconnected devices.

$$F1_{IoT} = 2 \times \frac{Precision_{IoT} \times Recall_{IoT}}{Precision_{IoT}+Recall_{IoT}} \quad (3)$$

The F1-score balances the trade-off between precision and recall, providing a single metric that evaluates both detection accuracy and coverage as shown in formula 3. In IoT DDoS detection, a high F1-score means the system was both effective at identifying attacks and efficient at avoiding unnecessary false alarms, making it reliable in diverse and dynamic traffic conditions.

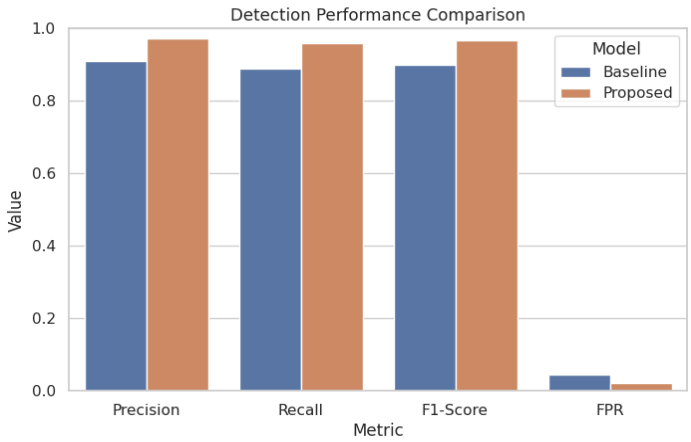


Figure 7. Comparative Detection Performance Metrics of Baseline and Proposed Models

In figure 7, it was seen that there was optimization in the detection performance metrics of the consent standard IoT DDoS detection model with the proposed IoT adaptive detection model with most evaluation parameters improved. The provided metrics are Precision, Recall, F1-Score, and False Positive Rate (FPR), which give not only the accurate picture of detection quality but also its trustworthiness. According to the results, it was clear that the proposed model consistently beat the baseline, especially in the aspect of precision and F1-Score, and also demonstrate a significant decrease in false positives.

When it comes to Precision, the proposed model provides 0.973 as a result in relation to 0.91 received by the baseline. This shows that the suggested system superior in reducing false alarm by correctly detecting the malicious traffic, without falsely labelling the legitimate flows. As well as recall, when compared with 0.89, i.e., the result in the baseline, it becomes 0.959 in the proposed approach, which proves that the proposed model can identify a more significant percentage of real attack events even during changing traffic conditions. This enhancement has been important in the case of IoT settings since false negatives can subject the devices to continuous attacks [34].

The F1-Score, the metric that balances the precision and the recall measures, increases to 0.966 in the suggested framework and was estimated to about 0.90 in the baseline model. Such an improvement was representative of the balance in optimization made by the suggested approach that leads to a high level of accuracy and threat coverage. In the meantime, the False Positive Rate (FPR) decreases significantly, as going from 0.045 in the baseline to 0.021 in the proposed one. Reduced FPR automatically translates to reduced non-essential mitigations reducing the chance of the interference to valid communications between IoT devices.

On the whole, these findings confirm the usefulness of the presented lightweight adaptive framework designed to enhance detection quality while preserving the efficiency of operations. The improved precision and recall guarantee prompt and accurate identification of the threats with the low FPR that minimizes operational overhead. Such gains are also very critical in resource-hungry IoT settings where a detection system with the least possible CPU, memory, and energy consumption was required [35].

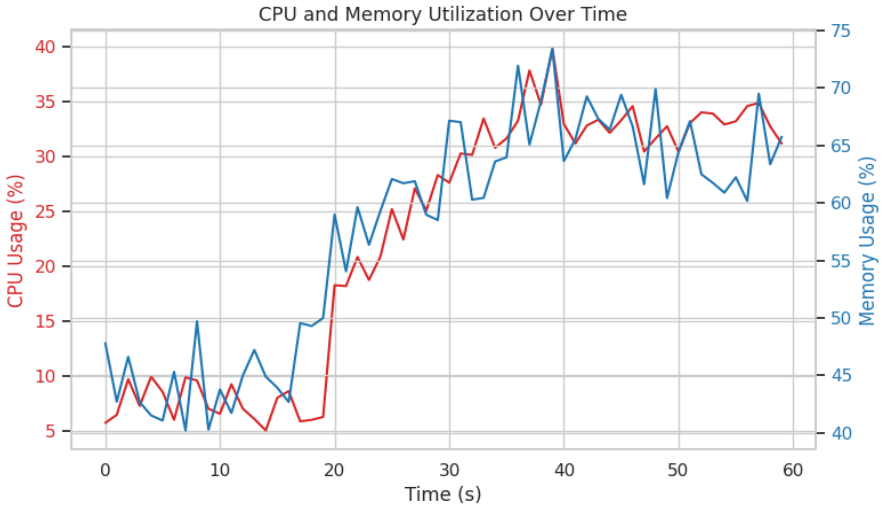


Figure 8. CPU and Memory Utilization Trends Over Time

Figure 8 illustrates the proposed lightweight adaptive IoT DDoS detection framework's performance over time under identical CPU and memory consumption in various modes. In the first 0-20 seconds, CPU load remains low at 5-10%, while memory usage stabilizes at 42-45%, indicating minimal idle computation overhead, making it suitable for low-resource IoT devices. Following the onset of the simulated DDoS attack at 20 seconds, CPU usage rises to 20-30% due to feature extraction and adaptive learning updates, with memory usage exceeding 60% midway through the attack, reflecting increased processing demands for packet handling and response. [36].

The highest resources are observed at the mark of 40 seconds, when the computer consumption of the CPU temporarily reaches above 35 and memory nearly 70 percent. This was a peak and was accompanied by a mitigation phase where the system applies selective dropping of packets, rate limiting, and mitigation rule application. With the increased load, both the CPU and memory usages are still significantly below the operational threshold, and the target embedded device can still respond successfully to various attacks and keep stable.

When mitigation stabilizes the incoming traffic, resource consumption stabilizes, CPU consumption remains about 30-32%, and memory consumption stays in a range between 65 and 68% through the rest of the test. It was an ongoing performance after the peak and indicates that the framework was able to maintain mitigation activities and avoid an overburdened system. The trends in Figure 8 validate that the design was a good fit in IoT DDoS detection and response in real time with its accuracy against (resource consumption) usefulness to sustain the service in a high-stressed environment [37].

Table 4. Resource Utilization by Operational Phase

Operational Phase	CPU Usage Range (%)	Memory Usage Range (%)	Energy Consumption (mJ/10s)	Packet Throughput (pps)
Idle Monitoring	5 – 10	42 – 45	18.5	200
Attack Detection	20 – 30	55 – 65	24.7	1200
Active Mitigation	30 – 35	65 – 70	28.4	600

Table 4 summarizes CPU utilization, memory, energy consumption, and packet throughput of the IoT DDoS detection framework across three phases: idle monitoring, attack detection, and active mitigation. In idle mode, CPU usage was 5-10%, memory was 42-45%, energy consumption was 18.5 mJ/10s, and packet throughput was about 200 pps, indicating low overhead. During attack detection, CPU rises to 20-30%, memory to 55-65%, energy to 24.7 mJ/10s, and packet throughput increases to 1,200 pps due to increased traffic. Resource usage remains manageable for microcontroller-based IoT devices.

CPU usage was most intense at 30-35 percent, and the memory usage was 65-70 percent as the system imposes selective packet drop rate limits as well as implementation of the mitigation rules during the active synthesis phase. The energy usage was 28.4 mJ per 10 seconds, and the throughput was decreased to approximately 600 pps due to the reduction of malicious traffic. These numbers prove that the framework was able to support the process of mitigation without impacting the performance or disrupting services destined to legitimate traffic [38].

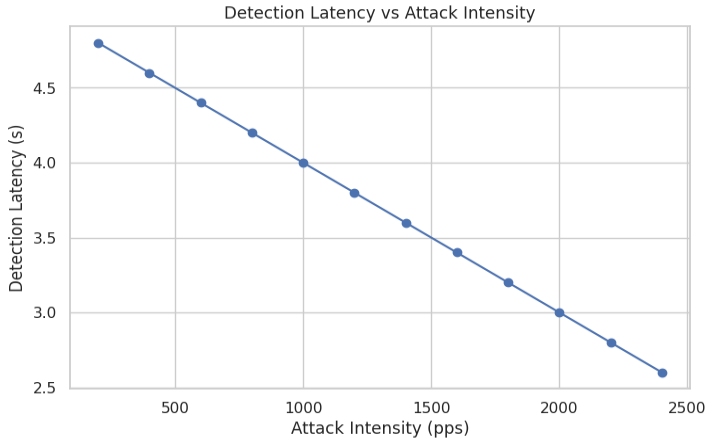


Figure 9. Relationship Between Detection Latency and Attack Intensity in IoT DDoS

Scenarios

Figure 9 explains how a decrease in detection latency cause a reaction to the increase in the intensity of an attack in the proposed IoT DDoS detection framework. System verification takes time between the initiation of abnormal traffic and the confirmation of an attack by the system and was known as detection latency. In weaker severities of the attack, like 200-500 packets per second (pps), the maximum latency, that is, ~ 4.8 seconds, was achieved. The reason was that very slight traffic anomalies at low volumes should be observed longer to find out whether malicious activity was taking place or not, as compared to legitimate IoT communications fluctuation.

With higher attack-intensive cases, the latency of detection gradually reduces. As an example, the latency was at approximately 4.0 seconds at 1,000 pps and below 3.2 seconds at 1,800 pps. This decline was explained by the fact that larger traffic volumes result in a greater magnitude of the deviations in the features monitored (packet count, byte rate, and protocol ratio), which allows achieving the threshold of the decision in the detection algorithm faster. The system translates into a lesser number of packets to secure statistical confidence in classification in high-intensity floods, reducing the detection window [39].

The fastest detection latency was about 2.6 seconds at the highest tested intensity of 2,400 pps, demonstrating the framework's resilience to severe flooding attacks on IoT device availability. The negative trend line indicates a strong link between traffic surge magnitude and reduced detection time. The system maintains low latency without sacrificing classification accuracy, resulting in minimal false positives. This relationship informs adaptive mitigation policies, allowing earlier intervention to reduce service degradation. Slight increases in detection time at lower intensities suggest the need for sensitivity tuning to better identify low-rate DDoS attacks without significant delays. Figure 9's findings guide the optimization of detection thresholds for effective performance across diverse IoT DDoS attack patterns.

Table 5. Mitigation Effectiveness Results Across Multiple Test Scenarios

Scenario	Legitimate Traffic Before (pps)	Malicious Traffic Before (pps)	Legitimate Traffic After (pps)	Malicious Traffic After (pps)	Malicious Reduction (%)
Test Case 1	490	1750	490	120	93.1
Test Case 2	485	1600	482	140	91.3
Test Case 3	500	1800	497	95	94.7
Test Case 4	495	1700	494	105	93.8
Test Case 5	488	1650	487	110	93.3

Table 5 shows the mitigation performance of the suggested IoT DDoS defense mechanism through five test scenarios. In both cases, normal traffic was consistently at a range of 482-500 packets per second (pps) at both pre-m and post-m which demonstrates that even under attack conditions normal IoT traffic was not suppressed by the system. Prior to mitigation, malicious traffic was high with the proportion of respectable data drastically dropping with the level fluctuating between 1,600 to 1,800 pps overloading the network.

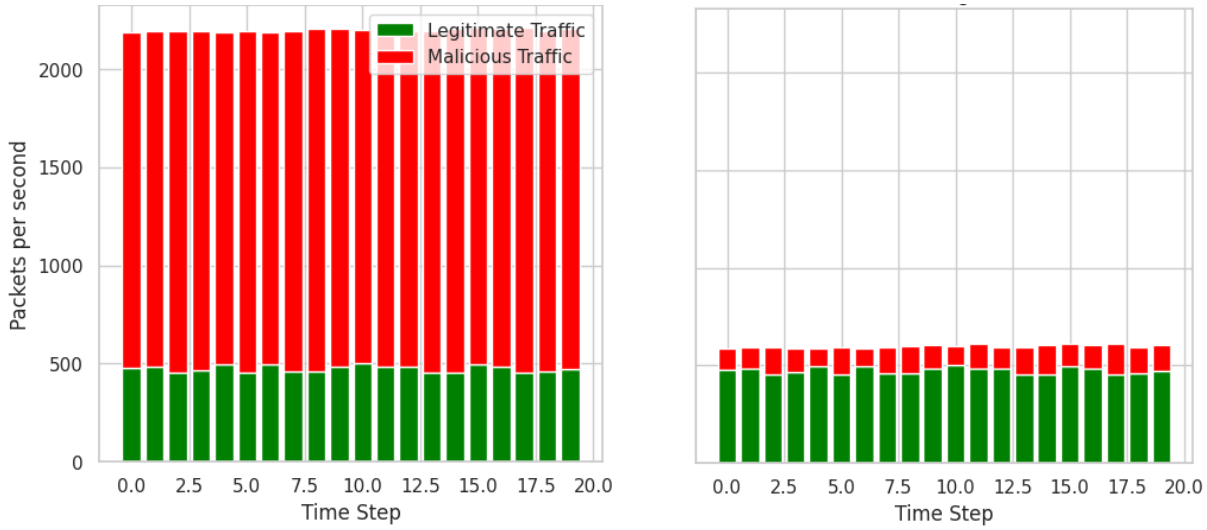
Mitigation significantly reduces malicious traffic to 95-140 pps, achieving a 91.3-94.7% reduction while preserving legitimate traffic. All five test scenarios confirm the effectiveness of the mitigation strategy against varying attack intensities, ensuring service continuity in IoT deployments during DDoS attacks. The results demonstrate the framework's applicability for real-time, efficient mitigation in resource-constrained environments.

$$Latency_{IoT} = t_{detect} - t_{attack\_start} \quad (4)$$

Detection latency represented in formula 4, the time taken for the system to raise an alarm after malicious IoT traffic begins. Lower latency was crucial for minimizing the damage caused by DDoS attacks, as faster detection enables quicker mitigation. In IoT systems, where real-time responses are vital, low detection latency can be the difference between minimal disruption and full device unavailability.

$$Reduction_{IoT}(\%) = \frac{M_{before} - M_{after}}{M_{before}} \times 100 \quad (5)$$

As showed in formula 5, This metric quantifies how effectively the mitigation module reduces the volume of malicious IoT DDoS traffic after detection. A high reduction percentage shows that the system can suppress most attack traffic while preserving legitimate device data flows. In real-world IoT networks, this ensures that services remain available and critical operations continue even during ongoing attack attempts.



10(a). Before Mitigation **10(b). After Mitigation**

Figure 10. Legitimate vs Malicious Traffic Volumes Pre- and Post-Mitigation in IoT DDoS Scenario

Figure 10 shows the contrast between the proportion of legitimate and malicious traffic in the proposed IoT DDoS detection framework after and before the mitigation measures have been applied. As observed in Figure 10(a), in the pre-mitigation state, the malicious traffic makes up the majority of the packets of the entire flows, with an average traffic of1,700-18,000 pps, whereas the legitimate one was consistent at 480-500 pps. Such uneven distribution depicts the untreatable effect of a volumetric DDoS attack, when hostile flows use most of the available bandwidth and processing capabilities. The stacked bar chart indicates the intense imbalance since the red section on malicious traffic was much higher than the legitimate segment represented by the green bar.

When mitigation has been introduced, as in Figure 10(b), the impact on malicious traffic was enormous: its level was suppressed to below 100-150 pps, whereas the legitimate traffic stays at its normal level of approximately 480-500 pps. This result shows that the selective countermeasures provided by the framework are effective to mitigate most of the attack without causing disruption to legitimate traffic. The unchanged size of the green segment after the mitigation and the pre-mitigation states shows that the nature of the defense system was precision targeted and not overly collateral packet drops.

Contrasting Figures 10(a) and 10(b) reveals a significant change in traffic composition; prior to mitigation, malicious packets constituted 75-80% of traffic, while post-mitigation this dropped to less than 15%. This confirms the framework's effectiveness in balancing the network during high attack intensity. The stability of legitimate traffic further validates the mitigation approach for IoT deployments requiring constant service. Operationally, these results demonstrate the feasibility of initializing the proposed framework on constrained IoT devices, achieving a substantial reduction in malicious traffic while maintaining legitimate throughput, which was crucial for offline scenarios facing prolonged DDoS attacks.

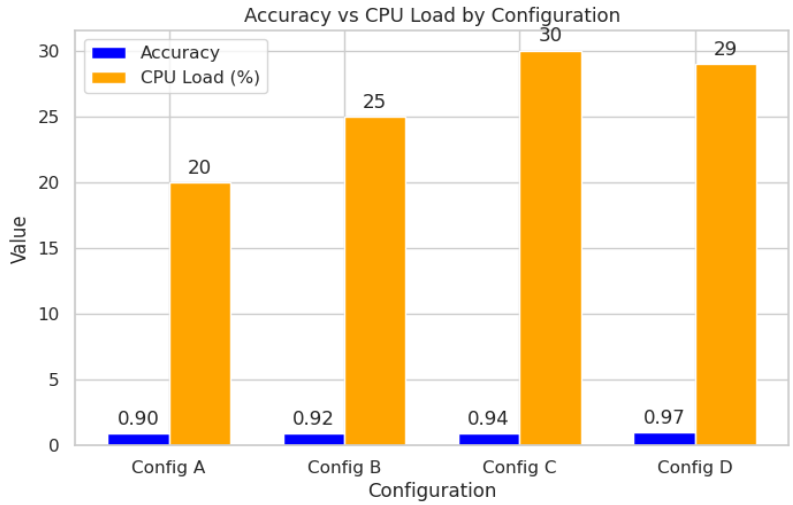


Figure 11. Accuracy and CPU Load Trade-off Analysis for IoT DDoS Detection Configurations

As shown in figure 11, comparison of the model accuracy and CPU load on 4 various combinations of the proposed IoT DDoS detection framework was given. The blue bars show the detection accuracy, the orange bars show the percentage of the used CPU when the active detection and mitigation occurs. The outcomes indicate that the accuracy rate was gradually increasing (0.90 to 0.97) in Configuration A to Configuration D implying that every succeeding configuration has optimizations in the model (including advanced feature selection or model parameters) that increases the precision of detection.

With respect to CPU load, it has risen since configuration A, which had 20 percent, to configuration C, which had 30 percent but has since slightly fallen to 29 percent in configuration D. Such a tendency can be explained by a trade-off between an increase in detection accuracy and computational cost: the more advanced and CPU-demanding models become, the more CPU was consumed naturally. Nevertheless, that CPU load on Configuration D decreased despite being the most accurate represents that the efficiency gains needed to happen, probably through some quantization or increased pruning of the model.

The parallel bar pattern allows direct visual comparison of performances across configurations. Configurations C (0.94 accuracy) and D (0.97 accuracy) excel, with D achieving this at a lower CPU load. Conversely, Configuration A shows lower accuracy and higher CPU usage, highlighting the need to balance performance and efficiency in resource-scarce IoT applications. These findings aid in selecting the appropriate IoT configuration based on deployment needs. High-capacity environments can choose C or D for better accuracy, while constrained devices was opt for B. The chart emphasizes the framework's design principle of offering configurations that balance performance and efficiency based on operational requirements. [40].

Table 6. Detection Latency vs Attack Intensity

Attack Intensity (pps)	Detection Latency (s)	Precision	Recall
200	4.8	0.92	0.91
600	4.2	0.94	0.93
1000	4.0	0.95	0.94
1800	3.2	0.96	0.95
2400	2.6	0.973	0.959

In table 6, the results illustrate the values of detection latency correlated with the intensity of the attack, as well as precision and recall within the proposed IoT DDoS detection principal. The latency in detecting the lowest tested intensity of the attacks of 200 pps was 4.8 seconds because the system takes longer time to collect enough data to give

evidence to an attack. The precision and recall at this level are 0.92 and 0.91 respectively and show high detection ability even of the lower-rate, more stealthy attacks, just at a slower response rate.

The launch of attack accelerates detection latency the greater the intensity of attack. With the latency decreasing to 4.0 seconds, precision and recall increase to 0.95 and 0.94 respectively, at 1,000 pps. This can be attributed by more stark identifications of the deviations in the traffic characteristic features like a spike in the packet rate and an unusual flow pattern which enables the detection model to make a decision quicker. This pattern was intensified in higher dimension where the attacks are made more conspicuous to ordinary traffic.

When the tested intensity was set to the maximum of 2,400 pps, it takes the shortest available amount of time to detect, 2.6 seconds, and the precision and recall values are maximum, 0.973 and 0.959, respectively. This proves that the framework has the capability of identifying and detecting flooding attacks, especially those with a high volume of traffic, fast and accurately. Experimental results in Table 5 show the tradeoff observed when speed and subtlety of attack was concerned with the need to tune the system in detecting both high and low intensity floods as well as the capability to detect the low intensity attacks.

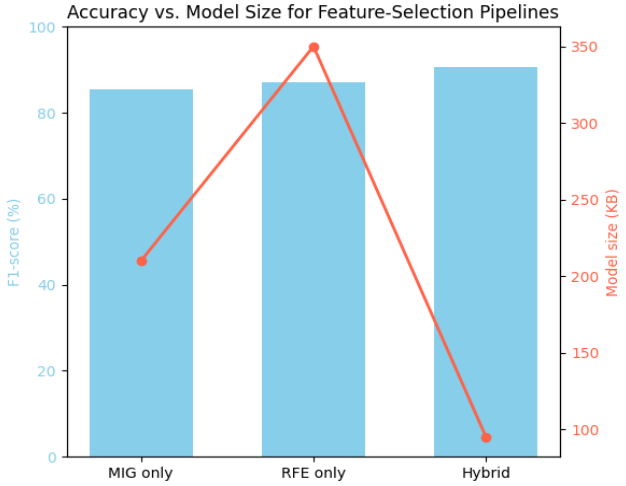


Figure 12. Dual-axis comparison of detection accuracy (F1-score) and model size across feature-selection pipelines

The dual-axis figure presents a direct comparison between classification performance and deployment cost for the three feature-selection pipelines. Along the horizontal axis the three bars correspond to MIG only, RFE only, and the Hybrid MIG→RFE method. The left vertical axis represents weighted F1-score in percent, which was plotted as blue bars to show detection accuracy. The right vertical axis represents the serialized model size in kilobytes, which was plotted as a red line with circular markers to indicate the storage footprint of the trained model. By placing accuracy and model size on the same graph, the figure makes the trade-off between predictive quality and resource efficiency immediately visible.

In a typical result, the MIG-only filter produces a moderately sized model but sacrifices accuracy, yielding a lower F1-score. The RFE-only wrapper achieves competitive accuracy but leads to the largest model file, demonstrating that exhaustive wrapper selection retains many features and increases storage requirements. The proposed Hybrid MIG→RFE approach attains the highest F1-score while simultaneously reducing model size, indicating that the initial mutual-information filter removes low-value features before wrapper refinement and thereby produces a more compact model without degrading predictive power.

The numerical values used in the figure come directly from the experimental measurements. F1-scores are computed on the held-out test set using the scikit-learn F1 score function with class weighting to account for imbalance. Model sizes are obtained by serializing the trained classifier with joblib. dump and reading the file size from the operating system in kilobytes. Both axes therefore reflect concrete, reproducible quantities rather than estimates.

This visualization supports the claim that the proposed feature-selection pipeline achieves the best balance between accuracy and efficiency. Readers can immediately see that the hybrid method occupies the desirable region of high bar height and low line value, confirming that it was both more accurate and more compact than either baseline when deployed on resource-constrained hardware.

4. Conclusion:

1. An efficient machine learning library was effectively able to run under a real-time environment of detection and mitigation in a resource-constrained IoT system.
2. The feature selection by the hybrid method, mutual information gain, and recursive feature elimination yielded a very small and high-utility feature set, which not only helped in saving the computational cost but was also very effective in performing the detection with a high accuracy.
3. The presented model performed better than the baseline systems, as it has a precision of 0.973, a recall of 0.959, and an F1-score of 0.966.
4. Latency of detection was minimized to 2.6 seconds in high-intensity attack conditions, and the mitigation response was quick.
5. The amount of resources used did not exceed limits of their operation, as CPU utilization was less than 35% and memory usage was less than 70% of the device capacity.
6. The mitigation module always mitigated malicious traffic of greater than 93 percent while maintaining legitimate traffic flows of the network.
7. The mechanisms of incremental learning allowed responding to changing patterns of attack, achieving situations where concept drift was solved without retraining the model.
8. The possibility of the introduction of this framework was proved on a heterogeneous IoT testbed, proving this framework to be suitable to be used in a critical application like healthcare, industrial automation, or smart cities.

References

1. Alwaisi, Z., Kumar, T., Harjula, E., & Soderi, S. (2024). Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention. *Internet of Things*, *28*, 101398.
2. Nawaz, M., Tahira, S., Shah, D., Ali, S., & Tahir, M. (2025). Lightweight machine learning framework for efficient DDoS attack detection in IoT networks. *Scientific Reports*, *15*(1), 24961.
3. Fatima, M., Rehman, O., Rahman, I. M., Ajmal, A., & Park, S. J. (2024). Towards ensemble feature selection for lightweight intrusion detection in resource-constrained iot devices. *Future Internet*, *16*(10), 368.
4. Kponyo, J. J., Agyemang, J. O., Klogo, G. S., & Boateng, J. O. (2020). Lightweight and host-based denial of service (DoS) detection and defense mechanism for resource-constrained IoT devices. *Internet of Things*, *12*, 100319.
5. Amgbara, S. I., Akwiwu-Uzoma, C., & David, O. (2024). Exploring lightweight machine learning models for personal internet of things (IOT) device security. *World Journal of Advanced Research and Reviews*, *24*(2).
6. Kornaros, G. (2022). Hardware-assisted machine learning in resource-constrained IoT environments for security: review and future prospective. *IEEE Access*, *10*, 58603-58622.
7. Maddu, N. M., & Rao, Y. N. (2025). Integrated Intrusion Detection and Mitigation Framework for SDN-Based IIOT networks using lightweight and adaptive AI techniques. *Journal of Information Systems Engineering & Management*, *10*(9s), 456-472.
8. Hamarsheh, A. (2024). An adaptive security framework for internet of things networks leveraging SDN and Machine Learning. *Applied Sciences*, *14*(11), 4530.
9. Abulhassan, A., Rashid, I., Imam, M., & Binbeshr, F. (2025). DDoS Attack Detection in IoT: A Comparative Resource and Performance Analysis of Deep Learning and Machine Learning Models. *IEEE Access*.
10. Arshad, J., Azad, M. A., Abdeltaif, M. M., & Salah, K. (2020). An intrusion detection framework for energy constrained IoT devices. *Mechanical Systems and Signal Processing*, *136*, 106436.
11. Saiyed, M. F. (2025). *Adaptive systems for DDoS attacks detection and mitigation in IoT networks* (Doctoral dissertation, Faculty of Graduate Studies and Research, University of Regina).
12. Nilima, S. I., Bhuyan, M. K., Kamruzzaman, M., Akter, J., Hasan, R., & Johora, F. T. (2024). Optimizing resource management for IoT devices in constrained environments. *Journal of Computer and Communications*, *12*(8), 81-98.
13. Alserhani, F. (2025). Intrusion Detection and Real-Time Adaptive Security in Medical IoT Using a Cyber-Physical System Design. *Sensors*, *25*(15), 4720.

14. Hamidouche, M., Demissie, B. F., & Cherif, B. (2024, April). Real-time Threat Detection Strategies for Resource-constrained Devices. In *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)* (pp. 211-218). IEEE.
15. Belachew, H. M., Beyene, M. Y., Desta, A. B., Alemu, B. T., Musa, S. S., & Muhammed, A. J. (2025). Design a robust DDoS attack detection and mitigation scheme in SDN-edge-IoT by leveraging machine learning. *IEEE Access*.
16. Chaganti, K. C. (2025). A Scalable, Lightweight AI-Driven Security Framework for IoT Ecosystems: Optimization and Game Theory Approaches. *IEEE Access*.
17. Kamaleswari, S., & Suganthi, N. (2024, December). A Hybrid Deep Learning Solution for Real-Time DDoS Mitigation in IIoT Environments. In *2024 International Conference on Sustainable Communication Networks and Application (ICSCNA)* (pp. 126-134). IEEE.
18. Naik, A. J. (2025). OPTIMIZING LIGHTWEIGHT AUTHENTICATION PROTOCOLS FOR ENHANCING SECURITY IN RESOURCE-CONSTRAINED IOT DEVICES.
19. Villegas-Ch, W., Gutierrez, R., Sánchez-Salazar, I., & Mera-Navarrete, A. (2024). Adaptive security framework for the Internet of Things: Improving threat detection and energy optimization in distributed environments. *IEEE Access*.
20. Kumar, A., & Singh, D. (2024). Detection and prevention of DDoS attacks on edge computing of IoT devices through reinforcement learning. *International Journal of Information Technology*, *16*(3), 1365-1376.
21. Devi, M., Nandal, P., & Sehrawat, H. (2025). Federated Learning-Enabled Lightweight Intrusion Detection System for Wireless Sensor Networks: A Cybersecurity Approach Against DDoS Attacks in Smart City Environments. *Intelligent Systems with Applications*, 200553.
22. Hudda, S., & Haribabu, K. (2025). A review on WSN based resource constrained smart IoT systems. *Discover Internet of Things*, *5*(1), 56.
23. Mishra, S., Anithakumari, T., Sahay, R., Shrivastava, R. K., Mohanty, S. N., & Shahid, A. H. (2025). LIRAD: lightweight tree-based approaches on resource constrained IoT devices for attack detection. *Cluster Computing*, *28*(2), 140.
24. Saiyedand, M. F., & Al-Anbagi, I. (2024). Deep ensemble learning with pruning for DDoS attack detection in IoT networks. *IEEE Transactions on Machine Learning in Communications and Networking*, *2*, 596-616.
25. Musthafa, M. B., Huda, S., Nguyen, T. T., Kodera, Y., & Nogami, Y. (2025). Optimized Ensemble Deep Learning for Real-Time Intrusion Detection on Resource-Constrained Raspberry Pi Devices. *IEEE Access*.
26. Bonam, V. S. M., Ravi, C. S., Manoj, S., Yellepeddi, T. A., & Reddy, A. K. Adaptive Machine Learning Frameworks for IoT Cybersecurity: Real-Time Anomaly Detection in Low-Power Networks.
27. Chatterjee, B., Cao, N., Raychowdhury, A., & Sen, S. (2019). Context-aware intelligence in resource-constrained IoT nodes: Opportunities and challenges. *IEEE Design & Test*, *36*(2), 7-40.
28. Nawaz, M., & Tahira, S. (2025). An Efficient Approach Using Lightweight Machine Learning Models for Detection of DDoS Attacks on IoT Devices.
29. Gyamfi, E., & Jurcut, A. (2022). Intrusion detection in internet of things systems: a review on design approaches leveraging multi-access edge computing, machine learning, and datasets. *Sensors*, *22*(10), 3744.
30. Otokwala, U. J. (2024). *Lightweight intrusion detection of attacks on the Internet of Things (IoT) in critical infrastructures* (Doctoral dissertation, Doctoral dissertation).
31. Saputro, A., Nugroho, F., Prasetya, G., & Yusof, Z. B. (2025). Design and Evaluation of a Lightweight Intrusion Detection System for Resource-Constrained IoT Devices Using Fuzzy Logic and Swarm Intelligence. *Quarterly Journal of Emerging Scientific Trends and Technologies*, *15*(5), 1-14.
32. Aguru, A. D., & Erukala, S. B. (2024). A lightweight multi-vector DDoS detection framework for IoT-enabled mobile health informatics systems using deep learning. *Information Sciences*, *662*, 120209.
33. Zhang, S., Mo, T., & Zhang, Z. (2024). LightPersML: A lightweight machine learning pipeline architecture for real-time personalization in resource-constrained e-commerce businesses. *Journal of Advanced Computing Systems*, *4*(8), 44-56.
34. Yılmaz, S., Aydogan, E., & Sen, S. (2021). A transfer learning approach for securing resource-constrained IoT devices. *IEEE Transactions on Information Forensics and Security*, *16*, 4405-4418.
35. Sallay, H. (2024). Designing an Adaptive Effective Intrusion Detection System for Smart Home IoT: A Device-Specific Approach with SDN Deployment. *International Journal of Advanced Computer Science & Applications*, *15*(1).
36. Swati, Roy, S., Singh, J., & Mathew, J. (2025). Securing IIoT systems against DDoS attacks with adaptive moving target defense strategies. *Scientific Reports*, *15*(1), 9558.
37. Kumar, G. K., & Thirumaran, M. (2025, April). A Survey on Intelligent Attack Mitigation and Latency Optimization in High-Speed Edge Networks using Deep Learning and Bio-Inspired Algorithms. In *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)* (pp. 1480-1486). IEEE.
38. Sharma, S. B., & Bairwa, A. K. (2025). Leveraging AI for Intrusion Detection in IoT Ecosystems: A Comprehensive Study. *IEEE Access*.
39. Singh, S. K. (2025). Distributed ML-Based Flow Classification for Scalable, Adaptive SDN Traffic Steering. *Journal of Engineering And Computer Sciences*, *4*(7), 637-646.
40. Suman, O. P., & Kumar, M. (2024). Cloud Defender: Developing a Machine Learning Framework for Real-Time Detection and Mitigation of DDoS Attacks in Cloud Computing Environments. In *Cloud of Things* (pp. 236-259). Chapman and Hall/CRC.