World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

Machine learning-based intrusion detection for resource constrained networks

Mayowa Samuel, Alade [1, 2, *], Samuel Olujimi, Adejumo [1], Godspower Ifeanyi, Akawuku [3], Olatunde Ayodeji Akano [4], Aminu A. Olanrewaju [4] and Godwin O, Osakwe [5]

[1] *Department of Cybersecurity, Faculty of Physical Sciences, Nnamdi Azikiwe University, Awka, Nigeria.*
[2] *Department of Computer Science, Faculty of Physical Sciences, Nnamdi Azikiwe University, Awka, Nigeria.*
[3] *Department of Software Engineering, Faculty of Physical Sciences, Nnamdi Azikiwe University, Awka, Nigeria.*
[4] *Department of Computer Sciences, Faculty of Engineering and Technology, Abiola Ajimobi Technical University, Ibadan, Nigeria.*
[5] *Department of Computer Science, Faculty of Computing, Southern Delta University, Ozoro, Delta, Nigeria.*

World Journal of Advanced Engineering Technology and Sciences, 2026, 18(03), 124-139

Publication history: Received on 15 January 2026; revised on 01 March 2026; accepted on 02 March 2026

Article DOI: https://doi.org/10.30574/wjaets.2026.18.3.0129

Abstract

The rapid growth of interconnected devices in small office and home networks has introduced heightened cybersecurity risks, yet traditional Intrusion Detection Systems (IDS) often demand extensive computational resources, making them unsuitable for deployment in resource-constrained environments. This study presents the design, implementation, and evaluation of a lightweight machine learning-based IDS optimized for small networks with limited processing power and memory. The research employed the CICIDS2017 dataset as the primary benchmark, subjecting it to comprehensive preprocessing, including data cleaning, normalization, encoding, feature scaling, and dimensionality reduction through Principal Component Analysis (PCA). Multiple classical Machine Learning algorithms, including Decision Tree, Random Forest (pruned), Naïve Bayes, K-Nearest Neighbors, and Ridge Classifier, were implemented and comparatively evaluated. Performance metrics such as accuracy, precision, recall, F1-score, CPU utilization, memory usage, and latency were used for assessment. Results indicated that the Random Forest achieved the best balance between accuracy and efficiency with low false positive rates, and minimal computational requirements suitable for lightweight environments. The Random Forest was integrated into a Flask-based RESTful API and a Streamlit dashboard. By bridging machine learning techniques with practical deployment frameworks, it contributes a resource-efficient, scalable, and user-friendly security solution tailored to small enterprises and personal network environments.

Keywords: Intrusion Detection System (Ids); Lightweight Machine Learning; Small Networks; Dashboard Cybersecurity; Ridge Classifier; Resource Constrained Network

1. Introduction

The continuous growth of internet-connected devices in homes, offices, personal laptops, and internet of Things (IoT) systems has exposed many network environments to a wider range of cyber threats. Attackers exploit the limited defense capabilities of these resourced-constrained devices for cyber-attacks such as Distributed Denial of Service (DDoS), brute force, and port scanning, which can easily disrupt operations. Intrusion Detection Systems (IDS) remain one of the most important tools for safeguarding such networks, but most existing IDS solutions are designed for enterprise-grade infrastructures. These systems typically demand significant processing power and memory resources, making them unsuitable for constrained environments where efficiency and lightweight performance are critical (Sommer and Paxson, 2010; Ring et al., 2019). Recent studies have advanced the use of machine learning (ML) and deep learning techniques for intrusion detection. Researchers have demonstrated that classification-based algorithms can significantly improve the detection of both known and emerging threats while reducing the limitations of manual rule-

* Corresponding author: Alade, Samuel Mayowa

based systems. Benchmark datasets such as CICIDS2017 and UNSW-NB15 are widely used to train and validate these models, and many proposed frameworks have achieved detection accuracies above 95% (Shone et al., 2018; Ferrag et al., 2020). These contributions underline the value of ML in IDS research, showing its potential to strengthen cybersecurity defenses. However, most existing IDS research focuses heavily on accuracy while neglecting computational efficiency, leaving a gap in real-world applicability. Many of the proposed models rely on high-dimensional features or resource-intensive algorithms, which are impractical for small networks or IoT devices with limited CPU and memory capacity. This often results in high false positive rates, increased detection delays, and poor adaptability outside laboratory conditions. Only a few studies have considered the trade-off between detection accuracy and resource consumption, meaning that lightweight IDS solutions for constrained environments remain underexplored (Khan et al., 2021).

This study is aimed at designing and implementing a lightweight, machine learning-based intrusion detection model integrated into a dashboard, capable of accurately detecting and classifying malicious network activities in low-resource environments. It proposes a lightweight machine learning-based IDS that is specifically optimized for resource-limited networks. It incorporates effective data preprocessing, normalization, and dimensionality reduction, followed by the evaluation of efficient algorithms such as Decision Tree, Naïve Bayes, Ridge Classifier, and K-Nearest Neighbors. The most suitable model was then integrated into a lightweight Application Programming Interface (API) and dashboard for real-time deployment. The system was tested using DDoS scenarios and other intrusion attempts to confirm its performance. Through this approach, the study contributes to both academic research and practical implementation by providing an IDS that is accurate, computationally efficient, and deployable in small-scale network environments.

2. Review of Related Literature

Intrusion Detection Systems (IDS) are key mechanisms designed to detect unauthorized access and malicious activities in a network. However, traditional IDS are resource-intensive and may not adapt well to the constraints of lightweight environments. Shanono et al. (2025) discussed how lightweight devices like smartphones and IoT gadgets, despite their widespread adoption, pose significant challenges for IDS implementation due to their low computational power and storage limitations. Similarly, Zhao et al. (2023) emphasized that small networks, often lacking dedicated IT support, demand IDS solutions that can function efficiently without overwhelming the network or device resources.

There exist several types of Intrusion Detection Systems Namely Network-Based Intrusion Detection System (NIDS), Host-Based Intrusion Detection System (HIDS) and Hybrid Intrusion Detection System (HIDS). The NIDS monitors network traffic at multiple points to detect anomalies, making it effective in large-scale deployments (Ahmed et al., 2022). Host-based Intrusion focuses on monitoring specific devices by analyzing log files, process activities, and system changes to identify suspicious behaviors (Chandola et al., 2023), while Hybrid IDS combines NIDS and HIDS to enhance security effectiveness, providing broader coverage and reducing false positives (Shone et al., 2021). In the same vein, there are different techniques employed in detecting intrusion in a system which include Signature-Based Detection and Anomaly based detection. Signature-based method compares network traffic against a database of known attack signatures (Fenanir et al., 2021). This method is highly accurate for detecting known threats but ineffective against zero-day attacks. The Anomaly-Based Detection employs machine learning techniques to detect deviations from normal behavior (Alwaisi et al., 2024). This approach enhances adaptability but may generate false positives. The hybrid Detection method combines the features of signature-based and anomaly-based techniques to provide a balance between accuracy and adaptability (Farooq and Khan, 2024).

In IDS, predictive analytics enables systems to analyze network traffic patterns and predict potential intrusions before they can compromise the system. Various machine learning algorithms, such as Decision Trees, K-Nearest Neighbors (KNN), and Naive Bayes, have been widely employed to facilitate predictive intrusion detection (Kumar andGarg, 2018). These models learn from labeled datasets, allowing them to identify malicious behaviors based on past patterns. For small networks, predictive analytics is invaluable because it supports real-time detection while ensuring computational efficiency, which is essential, given their limited resources.

Several researchers have made significant strides in integrating machine learning techniques into lightweight Intrusion Detection Systems (IDS) with diverse approaches aimed at balancing detection accuracy and computational efficiency. For instance, Pham et al. (2020) proposed a Lightweight Convolutional Neural Network (CNN)-based IDS, where they transformed raw network traffic into image representations to reduce feature space complexity. Their approach achieved 95% detection accuracy while maintaining a reasonable detection time and requiring relatively small training data, making it viable for small-scale networks and encrypted traffic scenarios. Lipperman et al. (2000) also reviewed an IDS that used the host and network-based intrusion detection with an overall performance in identifying previously unseen new, stealthy, and Windows NT attacks.

Alzahrani et al. (2024) addressed the challenge of network vulnerabilities and threats with a view to build a lightweight IDS based on hybrid ML models for IoT systems. To achieve the objective of building energy-aware IDS in the fog layer, with a focus on reducing and mitigating overhead, three (3) datasets namely CICIoT2023, KDD-99 and NSL-KDD were utilized. The authors designed and developed a hybrid deep learning machine learning model that combines convolutional neural networks (CNN) and long short-term memory (LSTM) to build an energy-aware, anomaly-based IDS. The performance of the model developed was tested and results evaluated comparatively using several key metrics such as latency, energy consumption, false alarm rate and detection rate. Findings yielded 92% accuracy and a false alarm rate below 0.38%. Also, the results revealed that the developed system enhanced security without resource utilization.

Aliyu et al. (2021) developed an anomaly-based, human immune, lightweight IDS for the fog layer based on anomalous events. To achieve a low resource overhead, the authors designed the model activities using Unified Modelling Language UML tool, particularly the Activity diagram and simulated the proposed IDPS, which is an immune-based IDPS for Fog Computing networks using OMNET++ and Artificial neural network (ANN) on the KDD 99 and NSL-SL datasets collected in the study on RPi to detect attacks. The model was evaluated using three metrics namely: accuracy, latency and energy -consumption power. The result of the experiment showed the model achieved a low resource overhead by distributing the IDS functions among the fog nodes and the cloud giving accuracy of 98.8%, and a 10% reduction in the energy consumption of the fog node when compared with deploying a neural network on the fog node.

Khater et al. (2019) employed a Multilayer Perceptron (MLP) model with a Raspberry Pi acting as a fog node to develop a lightweight intrusion detection system for a fog computing environment. The ADFA-LD dataset collected and used achieved 94% accuracy, 95% recall, and 92% F1 score, and on the ADFA-WD dataset, outcome showed 74% accuracy, 74% recall, and 74% F1 score, to identify various attack types. This study revealed an average testing time of roughly 750 microseconds, which is appropriate for a lot of Internet of Things applications.

Roy et al. (2022) addressed the challenge of security vulnerabilities encountered in IoT networks. To achieve the objectives of effectively mitigating and detecting cyber-attacks and anomalies in resource-constraint network devices using CICIDS2017 and NSL-KDD datasets, the authors designed a novel B-Stacking lightweight supervised intrusion detection model for IoT networks adopting improved machine learning algorithms pipeline methodology, including processes such as optimization techniques like sampling, data scaling, dimensionality reduction, and multicollinearity removal. The result of the experiment shows that B-Stacking achieved an accuracy, precision, recall, F1-scores of 99.11 ,99.08, 99.11 and 99.08 respectively using CICIDS2017 dataset. In a similar vein, B-Stacking also achieved an accuracy of 98.5 ±0.3 on NSL-KDD dataset. Thus, the B-Stacking machine learning has a high detection rate and a very low false alarm rate as it performs better most of the state-of-the-art techniques.

Samy et al. (2020) investigated the problem of vulnerabilities associated with IoT devices. The authors developed a framework using six distinct DL models, including GRU, LSTM, Bi-LSTM, CNN, CNN-LSTM and DNN, and applied it on five different datasets such as CICDOS 2017, NSL-KDD, etc. Additionally, the model was simulated and evaluated using Kera's and TensorFlow in Python Programming Language to detect several types of cyberattacks. The model achieved an accuracy of 99.97% detection rate, 99.96% detection accuracy in binary classification and 99.65% detection accuracy in multi-class classification. The result showed that the LSTM model performs better than the other DL models in terms of accuracy and detection rate. The experiments demonstrated that the LSTM model outperforms the other DL models in relation to accuracy and detection rate.

Similarly, Zhao et al. (2021) developed an efficient Knowledge Distillation-based lightweight IDS, leveraging separable convolution layers to reduce the computational memory footprint. The model developed by the researchers achieved accuracy rates of 94.30% on the UNSW-NB15 and 91.46% on the KDDCUP-99 datasets respectively while reducing model size and computational costs by 99% compared to traditional deep learning approaches. Another contribution by Özer et al. (2021) focused on feature selection optimization. They demonstrated that selecting optimal feature pairs, rather than using full feature sets, significantly enhances energy efficiency without compromising detection performance. Using the Bot-IoT dataset, they trained lightweight IDS models achieving more than 90% detection accuracy, and addressing energy constraints in IoT-based small networks.

Furthermore, Dinh et al. (2024) proposed a lightweight IDS tailored for Vehicular Ad-Hoc Networks (VANETs) to address the challenge of compromise in the operation of the VANET. In the study, the authors employed a machine learning approach to design the framework for mitigating cyberattacks on the network. The framework developed utilized logistic regression model involving steps such as preprocessing and dimensionality reduction through the application of Principal Component Analysis on the UNR-IDD dataset to minimize operational overhead while

maintaining strong detection rates. The results of this research showed that small, efficient ML models could effectively secure VANETs with minimal resource consumption.

Additionally, Stolz et al. (2024) developed a lightweight IDS deployable on resource-constrained devices to address the security problem associated with IoT network. The researchers developed the solution on Raspberry Pi, combining signature-based and ML-based anomaly detection techniques. The architecture developed comprised of three components namely: Traffic capture, where PyCharm was applied to collect packet data; Threat detection algorithms (Random Forest and K-nearest neighbor); and Alert component. Their evaluation confirmed the system's capability to detect both known and unknown attacks while maintaining minimal CPU and memory usage, crucial for small network environments.

Jauhari and Guiana (2024) determined the computational capability of the IDS for resource constrained IoT devices. The researchers developed a hybrid Convolutional architecture comprising of a lightweight CNN and Bidirectional Long Short-Term memory (Bi-LSTM) algorithms to enhance the performance of the IDS on UNSW-NB15 dataset. The model achieved an accuracy of 97.28% and 96.91% for binary and multiclass classification of attacks respectively.

Lai et al. (2024) introduced a Finite Dirichlet Mixture Model (DMM)-based lightweight IDS, leveraging extended stochastic variational inference to reduce computational load while ensuring robust classification performance. The results showed competitive detection rates with significantly lower training and detection times compared to traditional ML and DL models, validating its feasibility in lightweight environments.

Sahu et al. (2024) investigated the problems of distributed environments with resource limitations using Automatic Termination-based Whale Optimization with ELM to detect the various intrusions in fog computing. The methodology applied includes data preprocessing involving data normalization of dataset columns as well as optimization using automatic termination-based whale optimization algorithm (ATWOA) for selecting the features. Subsequently, the model was designed using the hybrid extreme learning machine (HELM), which classifies the various instruction types from ATWOA optimal features. The designed model was simulated using OIDS-FCE thus generating 98.52% accuracy, 96.38% precision, 95.50% of recall, and 95.90% of F1-score using UNSW-NB dataset, which outperformed other artificial intelligence IDS models that has been developed.

Pandey et al. (2025) designed a lightweight based model in a bid to minimize the computational and memory complexity of an IoT based system. The authors developed and evaluated the machine learning (Decision Tree) framework on the NL-KDD and Bot-IoT dataset using performance metrics such as accuracy, memory and energy efficiency as well as inference time. The result from the experiment achieved an accuracy of 98.2% and 97.9% respectively, and 0.45W-0.48W energy efficient, 12.5 to 13.1MB memory efficient and 0.8ms to 0.9ms inference time. Similarly, out of the machine learning model (Random Forest, Support Vector machine, K-nearest neighbor and Logistic regression) utilized, decision tree performs better than the others. It classifies attacks using Random Forest and addresses data imbalances using deep learning.

The CICIDS2017 dataset has been used to test a variety of machine learning-based IDS algorithms (Bookham and Gavito, 2018). An IDS based on deep neural networks (DNNs) is suggested to identify and categorize cyberattacks (Vinaykumar et al., 2019). Researchers have used a variety of datasets, such as KDDCUP 99, NSL-KDD, UNSW-NB15, WSN-DS, and CICIDS2017 (Boukhamla and Gavito, 2018; Li et al 2014; Moustafa and Slay, 2015; Almo Maini et al., 2016), to conduct a thorough evaluation of DNNs and other classical machine learning classifiers. They verified that DNNs outperform the conventional machine learning classifiers in terms of performance. Sethi et al. (2021) presented a reinforcement learning-based intrusion detection system (IDS) that classifies network attacks using attention mechanisms and Deep Q-Network logic in multiple distributed agents.

Despite these notable advancements, most studies focus heavily on IoT, VANET, or cloud-fog-based systems, leaving small traditional networks such as home offices or SMEs underrepresented. Furthermore, many experiments remain confined to controlled datasets without thorough real-world deployment validation. Several researches have also been conducted to enhance intrusion detection using machine learning techniques in different environments including IoT and Software Defense Network (SDN). However, there are still some limitations in these studies among which are reliance on simulated environments, lack of real-world deployment validation, least regards for resource-constrained systems and little consideration for development of effective lightweight solutions, neglecting general small networks like home offices and SMEs (Egwu, et al. 2025).

Most researchers also utilize large labelled datasets, which may not reflect real-time network conditions in small-scale environments. Therefore, this study was conducted to bridge these gaps by designing an optimized and lightweight IDS

tailored for general small networks. Real-time traffic testing and performance evaluation metrics such as accuracy, precision, recall, F1 score, mean absolute error (MAE) were used to determine the best machine learning techniques.

3. Methodology

To achieve the goal of designing an efficient, lightweight ML classifier model, and subsequently a system capable of accurately detecting and classifying malicious network activities in low-resource environments, the methodology adopted is structured to directly address the four research objectives. It involved a sequential process beginning with data preprocessing, feature engineering, model development and comparative evaluation, integration into an application programming interface (API) and dashboard framework, and finally, deployment and testing with simulation of real-world small network environment features. The theoretical framework guiding this research draws upon principles of supervised machine learning, specifically classification-based intrusion detection, and the resource-optimization paradigm for constrained computing environments. This framework supports the hypothesis that by combining efficient feature selection, normalization, and pruned algorithms, it is possible to achieve real-time intrusion detection without significant degradation of performance in resource-limited hardware such as office routers, personal laptops, and low-power edge devices.

3.1. Data Preprocessing for intrusion Detection Datasets

This study employed publicly available datasets that are widely recognized in intrusion detection research, CICIDS2017. These datasets were selected due to their comprehensive representation of modern cyber-attack scenarios, including types such as DoS, Port Scan, DDoS, Brute Force, alongside application-layer threats, as well as realistic benign traffic patterns. The raw datasets were obtained in CSV format and subsequently loaded into Jupyter Notebook under the Anaconda environment. Figure 1 shows the workflow architecture of the system.

The preprocessing phase commenced with an extensive data cleansing process to address issues of missing values, outliers, and inconsistent entries. Missing values were either imputed using statistical measures such as mean or median for numerical attributes or replaced with mode for categorical attributes. Records with irreparable inconsistencies were discarded to maintain data integrity. Outliers, particularly those resulting from anomalies in traffic capture tools, were identified and treated using interquartile range (IQR) methods. Infinite values, often arising from division-by-zero errors in flow-based metrics like Flow Bytes, were replaced with finite approximations to ensure computational stability.

3.2. Feature Engineering

Feature engineering is an essential step aimed at improving the performance or sensitivity of the given dataset. In fact, this step was undertaken to improve the discriminatory power of the model. In the process, redundant and highly correlated features were eliminated through correlation-based feature selection (CFS), which helps reduce dimensionality and prevents model overfitting. Additionally, categorical attributes such as protocol type, service, and flag were transformed into numerical representations via label encoding to ensure compatibility with machine learning algorithms. Attack class labels were mapped to binary or multi-class numerical values depending on the classification scenario being tested.

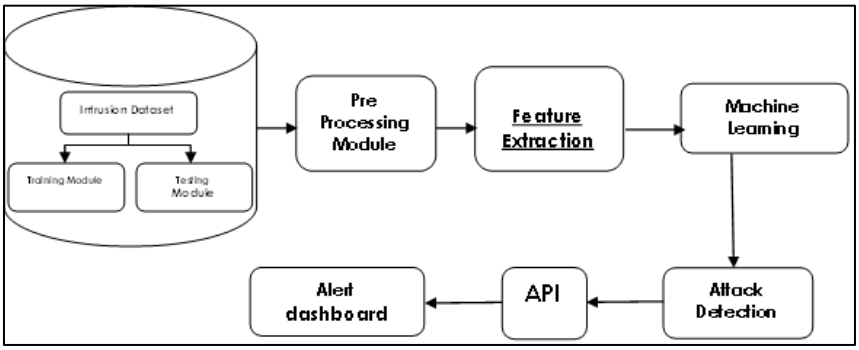

Figure 1 Intrusion Detection Workflow

3.3. Normalizing and Encoding Dataset Features

Normalization involves adjusting the data distribution to ensure uniformity and consistency across features. Raw network traffic data often contains inconsistencies, missing values, redundant information, and outliers, all of which can distort model training. To address these issues, the dataset was subjected to a multi-stage preprocessing pipeline. Missing values were treated using statistical imputation (mean and median for continuous variables, mode for categorical variables), while records with irrecoverable errors were dropped. Outliers caused by traffic capture anomalies were filtered using the interquartile range (IQR) technique, and infinite values in flow features such as "Flow Bytes" were replaced with finite estimates to ensure computational stability. Min-Max scaling was chosen to map all feature values into the range [0, 1]. This method was selected over Z-score standardization due to its computational efficiency and predictability in resource-constrained deployments.

Label encoding was also carried out as part of normalization. The attack class column, originally categorical, was transformed into binary form assigning a label of 1 for all attack instances and 0 for normal traffic. Normalization ensured that all entries in the dataset were clean, logically consistent, and uniformly formatted, thereby reducing the risk of bias, skewed training, and unnecessary memory usage during model learning and inference.

3.4. Feature Scaling and Dimensionality Reduction

Following the normalization stage, the datasets were subjected to feature scaling to ensure equal contribution of each feature to the model's learning process. Furthermore, Principal Component Analysis (PCA) was applied as a dimensionality reduction technique to condense the feature space into a smaller set of uncorrelated principal components while retaining most of the variance in the data. This step directly contributed to the lightweight nature of the IDS by lowering computational costs during both training and inference. In this study, Min-Max Scaling was used. It transformed all features into a fixed range between 0 and 1 represented mathematically as

$$X\text{scaled} = a + \frac{((x - \min(x))}{(\max(x) - (\min(x))} \quad (1)$$

Where

- $Xscaled$ is the scaled value of the feature.
- X is the original value of the feature.
- $Xmin$ is the minimum value of the feature in the dataset.
- $Xmax$ is the maximum value of the feature in the dataset.
- The scaled values are mapped within a fixed range, typically [-1, 1].

This technique preserved the relationships among values in each feature column while ensuring computational simplicity. Unlike Z-score standardization, Min-Max scaling consumes fewer resources and is more predictable in terms of memory and runtime qualities that make it particularly suitable for resource-limited deployment contexts. The scaling was applied only to the input features and not to the label column. The output of this process was a uniform and efficiently scaled dataset that could be seamlessly fed into the selected machine learning models without introducing feature dominance, high variance, or training bias.

3.5. Development and Comparative Evaluation of Machine Learning Models for Intrusion Detection

This section outlines the process of developing and evaluating a machine learning-based intrusion detection model, specifically optimized for environments with constrained computational resources.

3.5.1. Developing the ML-Based IDS Model

Different machine learning techniques have been applied by different researchers to categorize network intrusions, as discussed in Related work. To determine the most effective model, multiple classic machine learning algorithms were implemented, including Decision Trees, Random Forest (with pruning techniques), Naive Bayes, and k-Nearest Neighbors. Decision Trees provide fast classification and interpretability with minimal computational demand, Naïve Bayes offers simplicity and high-speed probabilistic classification, Ridge Classifier provides linear modeling with strong generalization ability under low memory requirements, while K-Nearest Neighbors, although computationally intensive at inference, was optimized with dimensionality reduction techniques to make it viable for smaller datasets. Random Forest was pruned to reduce model size and overhead while maintaining accuracy. Model training was conducted in Python using Scikit-learn, with the preprocessed dataset using 70:30 splitting ratio into training and testing sets.

Hyperparameter tuning was done using grid search within resource-efficient bounds. The algorithm classifier with optimal performance was used for developing the IDS. Python's Scikit-learn library, along with other essential packages such as Pandas and NumPy, facilitate model development.

3.5.2. Comparative Analysis on Machine Learning Algorithm

For comparative analysis, supervised machine learning techniques such as Decision Trees, Random Forest (with pruning techniques), Naive Bayes, k-Nearest Neighbors and Ridge Classifier were used with Accuracy, Precision, Recall, F1-Score, MAE, MSE and other statistical evaluation metrics as key performance indicators to determine the best and the most suitable algorithm for the lightweight intrusion detection. Additionally, latency measurements will assess real-time detection feasibility.

3.5.3. Integration of the Model into an API and Dashboard

The model developed was integrated the model into a deployable framework that allows real-time interaction, analysis, and visualization of intrusion detection results. The theoretical basis for this stage lies in service-oriented architecture (SOA) principles, where computational models are exposed via well-defined interfaces such as REST APIs, enabling modularity, scalability, and interoperability with various client applications. The integration was achieved in two primary stages: development of a RESTful API using Flask and creation of a Streamlit-based dashboard for end-user interaction.

3.5.4. Development of the Application Programming Interface

The Flask framework was chosen due to its lightweight nature, ease of integration with Python-based machine learning models, and minimal overhead requirements making it well-suited for small network environments with limited processing resources. The Random Forest model, previously trained and serialized using Joblib, was loaded into the API for inference operations. Endpoints like the predict for the submission of script with feature vectors for classification. Uploaded script files were parsed and validated to ensure that all features required by the model were present. Preprocessing steps identical to those applied during model training (such as scaling and encoding) were executed on the incoming data to maintain prediction consistency.

The API returned structured JavaScript Object Notation (JSON) response containing predicted labels, classification probabilities, and summarized detection statistics. In cases of malicious traffic, additional metadata, such as detected attack type (e.g., DoS, PortScan, Brute Force), was included in the response. This ensured compatibility with external network monitoring tools or SIEM systems, allowing the Application Programming Interface (API) to serve as both a standalone detection service and a component in broader security architecture.

3.5.5. Development of Dashboard

To provide an intuitive user interface for small network administrators, a Streamlit-based dashboard was developed. The dashboard served as a frontend client to the Flask API, offering a simple drag-and-drop mechanism, uploads and the execution of Python-based attack scripts for real-time simulation. Upon file upload, the dashboard invoked the API, retrieved classification results, and presented them in a visually interpretable format. The dashboard's features included Summary Metrics Display, Visualization, Detailed Results Table and Downloadable Report. This integration not only enhanced usability but also demonstrated the feasibility of deploying an ML-based IDS forming the bridge between the research's theoretical contributions and its practical deployment potential for simple useability.

3.6. Implementation and Testing of the Intrusion Detection Model

The implementation phase began with packaging the trained Ridge Classifier model, StandardScaler, and selected features into serialized files using Joblib. These files were loaded into the Flask API, which served as the model inference engine. The Streamlit dashboard was deployed locally, connecting to the Flask API through HTTP requests for prediction and visualization.

To simulate resource constraints, the API was tested under conditions of limited CPU and memory availability. Here, the testing of the IDS was conducted in two stages: offline testing using benchmark datasets and live testing using simulated network traffic. For offline testing, unseen CSV datasets from CICIDS2017 were fed into the dashboard, and the results were compared against ground-truth labels. For live testing, malicious traffic such as DoS and PortScan was generated using Python-based Scapy scripts, while benign traffic was generated through regular web browsing and file transfer. This was achieved by running the Flask server alongside other background processes and monitoring system

performance using built-in tools such as Task Manager and Python's psutil library. The goal was to assess the model's responsiveness and stability when subjected to realistic multitasking scenarios.

4. Evaluations

4.1. Evaluation of dataset and Preprocessing

In the experiment, the methodology adopted was evaluated using a well-known and openly accessible dataset, CICIDS2017 (https://www.unb.ca/cic/datasets/ids-2017.html). CICIDS2017 dataset includes benign and popular attack network flows that satisfy real-world standards and are accessible to the general public. It is extensively utilized for real-world intrusion detection in the most recent cybersecurity research. Additionally, network traffic analysis results from CICFlowMeter are included in this dataset. It has 79 features for every record, 78 of which relate to network traffic and the remaining one determines whether the traffic is typical or a specific type of intrusion. The intrusion detection system has 14 distinct attack classes and one normal traffic class.

In comparison to other classes of the dataset, five classes namely: Infiltration, Web Attack Brute Force, Web Attack XSS, Web Attack SQL Injection, and Heartbleed have nearly no records, as shown in Figure 2. To represent these five classes as a single class, the authors combined them. Additionally, when compared to other classes, it was discovered that DDoS, PortScan, and DoS Hulk have significantly more records. Therefore, to produce a balanced dataset, the authors employed the various sampling techniques as specified in Section 3.1. Eleven classes result from combining the five classes in CICIDS2017; normal traffic is represented by 0 and each number between 1 and 10 represents an anomalous class, while some records that had missing values have been deleted.

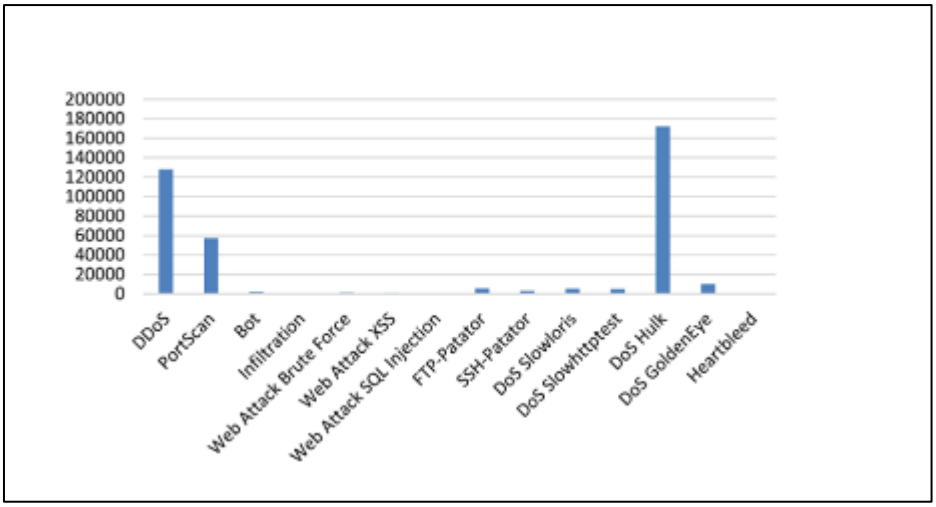

Figure 2 Graph Record Count of Each Class in CICIDS2017

4.2. Performance Metrics

Based on the preprocessing operation carried out on the dataset, 70% training dataset was used in building the machine learning algorithms model by fitting dataset in the model, and 30% of the testing dataset was used for prediction. To determine the most suitable algorithm for lightweight and accurate intrusion detection in constrained environments, five machine learning algorithms were implemented and evaluated. These include Decision Tree Classifier, Ridge Classifier, Random Forest, KNN and Naïve Bayes.

Each model was assessed based on key performance indicators or metrics: Accuracy, Precision, Recall, F1-score, MAE, MSE, and R2 Score. These metrics are defined based on Confusion matrix, which provides insights into the ability of each classifier to correctly detect both normal and attack instances. The confusion matrix is a N * N matrix designed to presents a graphical depiction of the classification outcomes, showing the counts of true positives, true negatives, false positives, and false negatives.

- **True Positives (TP):** An outcome where both the original data points and the predicted data points are true. Number of correctly classified intrusions.

- **True Negatives (TN):** An outcome where both the original data points and the predicted data points are false. Number of correctly classified legitimate (ham) messages.
- **False Positives (FP):** An outcome where the original data points are false but the predicted data points are true.
- **False Negatives (FN):** Number of outcome where original data points are true but the predicted data points are false.

4.3. Accuracy (ACC)

Accuracy is a performance metric that is capable of measuring the percentage of predictions that are classified correctly for the total test data instance and it is expressed as

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \quad (2)$$

4.3.1. Precision

Precision is a measure of True positive. The mathematical representation is as follows:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \quad (3)$$

4.3.2. Recall

Recall is a metric that is used to determine the proportion of properly predicted positive outcomes made out of all the outcomes in the given class and can be defined as follows

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \quad (4)$$

4.3.3. F1 Score

This is the precision and recall weighted average that gives a number between 0 and 1. This metric is seen as the best performance metric than accuracy and its mathematical representation is as follows:

$$F1\ Score = 2 * \frac{(\mathrm{recall} \times \mathrm{precision})}{\mathrm{recall} + \mathrm{precision}} \quad (5)$$

4.4. Mean Absolute Error (MAE) and Resource Metrics

MAE, CPU usage, memory usage, and latency were used to determine the model's efficiency and suitability for deployment in small networks.

5. Results and Discussion of Findings

The performance of the proposed intrusion detection mechanism, the comparison of the model performance was carried out using the metrics like Precision, accuracy, recall, F1-score, and MAE on CICIDS2017 dataset.

5.1. Evaluation of dataset

Figure 3 shows the confusion matrix of the proposed model for the CICIDS2017 dataset. The dataset was portioned into ratio 70:30 meaning that 70% was used for training and 30% for testing. The number of cases from each class that our method properly classified in this dataset is represented by the diagonal elements in Figure 3. The combined class (Benign, Bot, Brute Force, Dos, Infiltration, Port Scan, and Web Attack) have a total of 4319, 383, 565, 3855, 6, 1585, and 411 observations in our test data.

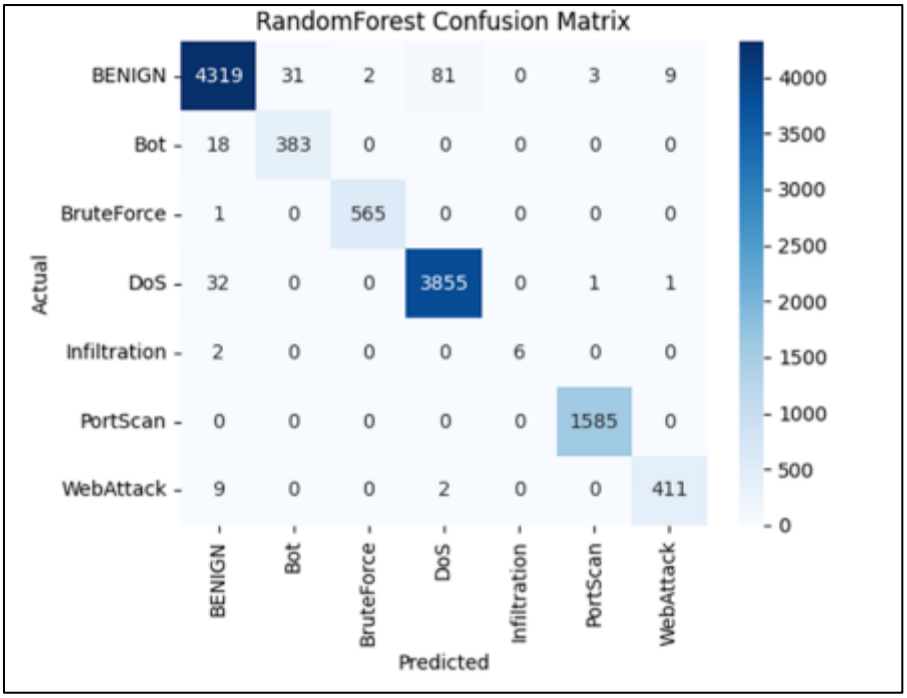

Figure 3 Confusion Matrix of Detection Model on CICIDS2017 Database

5.1.1. Decision Tree Performance Evaluation Metrics Result

The Decision Tree (DT) Classifier demonstrated fast training time and interpretability, making it useful for initial experiments. However, its performance, while decent, suffered from overfitting in some cases due to its complexity. Its accuracy was approximately 98.2%, with a precision of 98.2%, recall of 98.2%, and F1-score of 98.2%. Although it showed a relatively balanced performance, the size of the model and resource consumption made it less favorable for deployment in constrained networks. This metrics result is shown in Table 1.

Table 1 Performance Metrics Result Using Decision Tree

S/N	METRICS	RESULT
1.	Accuracy	0.9818
2.	Precision	0.9819
3.	Recall	0.9818
4.	F1 Score	0.9818
5.	MAE	0.0502
6.	MSE	0.173
7.	R2 score	0.9545

5.1.2. Ridge Classifier Performance Evaluation Metrics Result

The Ridge Classifier, known for its lightweight and regularized nature, had the modest accuracy of 84.95%, precision of 73.9%, recall of 59.48%, and F1-score of 58.79% as shown in Table 2. Its linearity makes it efficient in constrained environments despite a lower detection rate. Furthermore, its balanced performance across all metrics indicates strong generalization, making it a favorable choice for lightweight intrusion detection systems.

Table 2 Performance Metrics Result Using Ridge Classifier

S/N	METRICS	RESULT
1.	Accuracy	0.8495
2.	Precision	0.7390
3.	Recall	0.5950
4.	F1 Score	0.5879
5.	MAE	0.973
6.	MSE	3.687
7.	R2 score	0.0302

5.1.3. Random Forest Performance Evaluation Metrics Result

The ensemble learning model (e.g., Random Forest (RF)) that utilizes several decision trees achieved the highest classification performance among all the evaluated models. By combining the predictions of numerous decision trees, the Random Forest effectively handled both linear and non-linear patterns within the data. It achieved an accuracy of 99.4%, a precision of 99.4%, recall of 98.1%, and an F1-score of 98.6% respectively as presented in Table 3.

Table 3 Metrics Result Using Random Forest

S/N	METRICS	RESULT
1.	Accuracy	0.9944
2.	Precision	0.9938
3.	Recall	0.9811
4.	F1 Score	0.9864
5.	MAE	0.0182
6.	MSE	0.0803
7.	R2 score	0.9792

5.1.4. KNN Performance Evaluation Metrics Result

K-Nearest Neighbors (KNN) is a distance-based classifier that classifies new instances based on the majority label of the closest training samples. The algorithm showed moderate performance, yielding an accuracy of 96.6%, precision of 88.3%, recall of 88.3%, and an F1-score of 83. 2%. While KNN is simple to understand and implement, its main drawback is computational cost during prediction, as it requires scanning the entire dataset to classify a new instance. This reduces its real-world application for intrusion detection in constrained environments unless the dataset size is strictly controlled or approximated with indexing structures. This metrics result is shown in Table 4.

Table 4 Metrics Result Using KNN

S/N	METRICS	RESULT
1.	Accuracy	0.9664
2.	Precision	0.8829
3.	Recall	0.8275
4.	F1 Score	0.8323
5.	MAE	0.0955
6.	MSE	0.3413
7.	R2 score	0.9117

5.1.5. Naive Bayes Performance Evaluation Metrics Result

Naïve Bayes (NB) is a probabilistic classifier with the assumption of feature independence. Among all the models, it was the least computationally expensive to train and execute. The accuracy, precision, recall and F1-score achieved by the NB model were 35.65%, 36.69%, 58.71%, and 33.86% respectively. Despite its simplicity and speed, the performance of NB was limited by its strong assumptions and inability to capture feature interactions effectively. This restricts its effectiveness in scenarios where more complex relationships exist among the input variables, such as in network intrusion patterns. This metrics result is shown in Table 5.

Table 5 Metrics Result Using Naïve Bayes

S/N	METRICS	RESULT
1.	Accuracy	0.3565
2.	Precision	0.3669
3.	Recall	0.5871
4.	F1 Score	0.3386
5.	MAE	1.5364
6.	MSE	5.0934
7.	R2 score	-0.3179

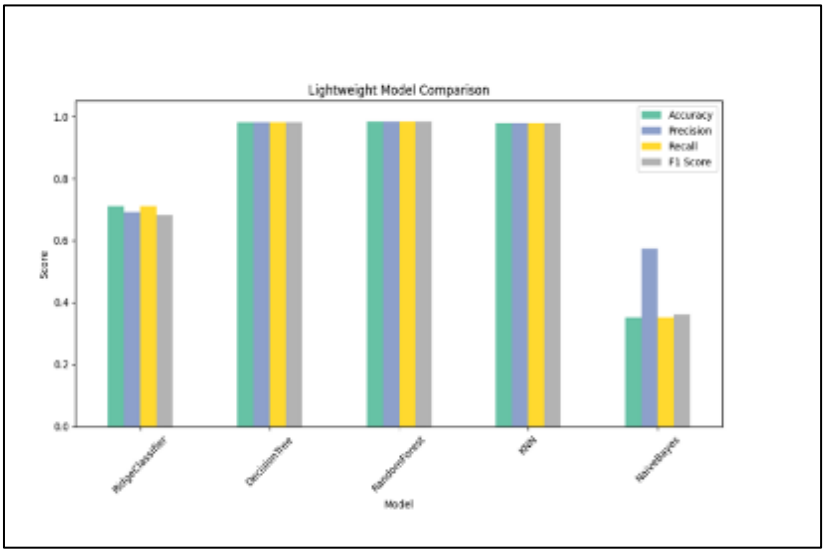

Figure 4 Comparison Bar of all Algorithms

Table 6 Comparative Analysis Table of the Models

Model	Accuracy	Precision	Recall	F1_SCORE	MAE	MSE	R2 Score
Decision Tree	0.9818	0.9819	0.9818	0.9818	0.0502	0.173	0.9545
Ridge	0.8495	0.7389	0.5948	0.5879	0.4861	1.8469	0.5221
Random Forest	0.9944	0.9938	0.98	0.9867	0.0182	0.0803	0.9792
KNN	0.9664	0.8829	0.8275	0.832	0.0955	0.3413	0.9117
Naive Bayes	0.3565	0.3669	0.5871	0.3386	1.5364	5.0934	0.3179

6. Discussion

The performance of the five machine learning algorithms applied were assessed and compared using metrics such as accuracy, precision, recall, and F1 score. Figure 4 gives the Bar chart comparing model performance, and Table 6 shows the comparison values across models. Experimental findings revealed and validated the hypothesis that classical algorithms such as Ridge Classifier and pruned Random Forest can provide strong detection performance while consuming fewer resources thus making them appropriate for operation in constrained environment. Unlike traditional IDS, which favours accuracy at the cost of scalability. In the experiment set up, it was discovered that out of the five machine learning models employed in detecting intrusion, the Random Forest (RF) model performed best followed by KNN, DT, Ridge Random Forest (RRF), and NB.

The Random Forest model, which had demonstrated superior accuracy during comparative analysis, was selected for deployment. The API successfully processed multiple requests with minimal latency, typically under one second per prediction, even when the system was under moderate load. To simulate continuous monitoring, the API was designed to accept scripts, which were internally converted to data frames before passing it through the model in real-time. This confirmed that the system could support detection and analysis.

Furthermore, the third objective of the study, was to integrate the selected model into an API and Streamlit dashboard to support real-time intrusion detection and visualization. The best model, Random Forest, was serialized and integrated into a Flask-based RESTful API with endpoints for uploading CSV traffic files and real-time Python-script traffic. A Streamlit dashboard was developed to interact with the API, providing users with a graphical interface to upload data, view classifications, and download reports. Testing the API with simulations yielded results within seconds, returning JSON responses with benign and attack classifications. The dashboard successfully displayed summaries in text, tables, and pie charts. Network administrators could visualize attack ratios and download PDF reports of detection outcomes.

Moreso, testing of the IDS was conducted in two stages: offline testing using benchmark datasets and live testing using simulated network traffic. For offline testing, unseen CSV datasets from CICIDS2017 were fed into the dashboard, and the results were compared against ground-truth labels. For live testing, malicious traffic such as DoS and PortScan was generated using Python-based scapy scripts, while benign traffic was generated through regular web browsing and file transfers.

Findings also revealed that while most studies stop at model training and report accuracy (Komisarek et al., 2021), this research extends to practical deployment with an operational API and dashboard. This bridges the research-to-practice gap and emphasizes usability for small businesses and home networks. The integration proved the feasibility of running IDS models in lightweight frameworks. Unlike traditional IDS which demand SIEM platforms, this system provides small networks with an intuitive tool that is both computationally feasible and user-friendly.

The system consistently maintained an accuracy above 95% with low false positive rates. Resource monitoring during testing showed CPU usage below 40% and memory usage under 1.5GB, confirming the system's suitability for lightweight environments. Latency tests indicated that classification for batches of 1,000 traffic records was completed in under two seconds, enabling near real-time detection. The implementation demonstrated that the IDS could run effectively in constrained environments without requiring enterprise-grade hardware, achieving the research aim of delivering a lightweight and practical intrusion detection solution.

As earlier reported, results confirmed that the IDS maintained detection accuracy above 95% while consuming less than 40% CPU and under 1.5GB memory. The deployment and testing phase validated the practicality of the solution, bridging the gap between theoretical model performance and real-world applicability. The system's ability to deliver accurate, timely predictions under resource constraints supports its potential for integration into edge devices, small-scale networks, and other environments where computational efficiency is paramount.

The result obtained in this study is compared to those of the four other sample intrusion detection systems described in related work. Thus, the results obtained presented by the present study is in consonance with the findings of Samy et al (2020) and Roy et al. (2022), who in separate studies investigated the problem of vulnerabilities associated with IoT five different datasets such as CICDOS 2017, NSL-KDD and addressed the challenge of security vulnerabilities, mitigating and detecting cyber-attacks and anomalies in resource-constraint network devices, particularly in IoT networks using CICIDS2017 and NSL-KDD datasets reported an accuracy of 99.97% detection rate, 99.96% detection accuracy in binary classification and 99.65% detection accuracy in multi-class classification, and 99.11% and 98.5 ±0.3. respectively.

Nevertheless, the result obtained and reported in the present study shows a slight contrast, which is better than the result reported in the works of Pandey et al. (2025), Jauhari and Guiana (2024) and Sahu et al (2024) who in their separate experiments using different datasets minimized the computational and memory complexity of an IoT based system, determined the computational capability of the IDS and addressed the unique challenges of distributed environments with resource limitations using Decision Tree, hybrid Convolutional architecture lightweight CNN and Bidirectional Long Short-Term memory (Bi-LSTM) and OIDS-FCE achieved an accuracy of 98.2%, 97.28% and 96.91%, and 98.52% respectively for binary and multiclass classification of attacks respectively in a bid to develop a solution for resource constrained devices. Again, the work of Zhao et al. (2021) reported 95% accuracy with a deep learning-based IDS requiring high computational resources, this study achieved similar accuracy with lightweight classical models optimized for efficiency. This demonstrates that lightweight ML can rival deep learning models when properly tuned. approach in this present work

7. Conclusion

The implementation and evaluation of the lightweight intrusion detection model demonstrated that the Ridge Classifier and Random Forest can deliver effective and real-time predictions while operating within the limitations of resource-constrained devices but Random Forest is most accurate. By integrating Flask and Streamlet, the intrusion system provides a user-accessible, deployable, and adaptable solution that meets the increasing demand for efficient network security tools. So, due to this result after analysis, the golden model was used in building the web application for detecting and mitigating intrusion attacks. The entire system successfully aligns with the original objective of delivering a practical and scalable machine learning-based IDS for environments with limited computational capabilities.

Unsupervised machine learning and deep learning algorithms can be used to improve the effectiveness of this system. Enhancing the model with alerting systems and exploring ensemble learning techniques could as well improve both responsiveness and accuracy. A mobile application can also be built for intrusion detection. Detection and mitigation of intrusion in cloud and embedded systems environment can also be considered in future research. The completed IDS provides a functional solution that meets the need for scalable, low-resource intrusion detection. It offers value to security professionals, academic researchers, and systems administrators by presenting a practical model that can be deployed and extended in various constrained settings. This research lays the groundwork for future innovations in edge-based cybersecurity systems, enabling smarter and more secure networks with reduced computational demands.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58. https://doi.org/10.1145/1541880.1541882

[2] Egwu, C., Akawuku, G., and Adejumo, S. (2025). Comparative Study of Real-Time and Batch Processing Approaches in Machine Learning-Based Fraud Detection for Financial Institutions, 12(2), 60-70. https://doi.org/10.59298/IAAJSR/2025/1226070.00

[3] Fenanir, S., Semchedine, F., and Baadache, A. (2019). A machine learning-based lightweight intrusion detection system for the Internet of Things. Revue d'Intelligence Artificielle, 33(3), 203–211. https://doi.org/10.18280/ria.330306

[4] Ferrag, M. A., Maglaras, L., Moschoyiannis, S., and Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications, 50, 102419. https://doi.org/10.1016/j.jisa.2019.102419

[5] Guezzaz, A., Azrour, M., Benkirane, S., Mohy-Eddine, M., Attou, H., and Douiba, M. (2022). A lightweight hybrid intrusion detection framework using machine learning for edge-based IIoT security. International Arab Journal of Information Technology, 19(5), 822–830. https://doi.org/10.34028/iajit/19/5/14

[6] Khan, M. A., Junejo, A. K., and Karim, S. (2021). Resource-efficient intrusion detection system for IoT environments. Journal of Network and Computer Applications, 175, 102918. https://doi.org/10.1016/j.jnca.2020.102918

[7] Kim, G., Lee, S., and Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 41(4), 1690–1700. https://doi.org/10.1016/j.eswa.2013.08.066

[8] Kumar, V., and Garg, L. M. (2018). Predictive analytics: A review of trends and techniques. International Journal of Computer Applications, 182(1), 31–37. https://doi.org/10.5120/ijca2018917434

[9] Lippmann, R. P., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000). The 1999 DARPA off-line intrusion detection evaluation. Computer Networks, 34(4), 579–595. https://doi.org/10.1016/S1389-1286(00)00139-0

[10] Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019). A survey of network-based intrusion detection data sets. Computers and Security, 86, 147–167. https://doi.org/10.1016/j.cose.2019.06.005

[11] Sahu, R., and Yadav, D. (2021). Lightweight Intrusion Detection System using Random Forest and XGBoost for IoT environment. Journal of Ambient Intelligence and Humanized Computing. https://doi.org/10.1007/s12652-021-03383-6.

[12] Shanono, N. M., Abu, N. A., and Mohamed, W. (2025). Intrusion detection system in lightweight devices: Systematic review and open research issues. Bulletin of Electrical Engineering and Informatics, 14(1), 800–812. https://doi.org/10.11591/eei.v14i1.4567

[13] Shone, N., Ngoc, T. N., Phai, V. D., and Shi, Q. (2018). A deep learning approach to network intrusion detection. IEEE Transactions on Emerging Topics in Computational Intelligence, 2(1), 41–50. https://doi.org/10.1109/TETCI.2017.2772792

[14] Sommer, R., and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. 2010 IEEE Symposium on Security and Privacy, 305–316. https://doi.org/10.1109/SP.2010.25.

[15] Özer E., Murat İ., and Azimjonov, J. (2021). Toward lightweight intrusion detection systems using the optimal and efficient feature pairs of the Bot-IoT 2018 dataset. https://doi.org/10.1177/15501477211052202

[16] Zhao, G., Wang, Y., and Wang, J. (2023). Lightweight intrusion detection model of the Internet of Things with hybrid cloud-fog computing. Security and Communication Networks, 2023, Article ID 4529757. https://doi.org/10.1155/2023/4529757.

[17] Zhao, R., Chen, Y., Wang, Y., Shi, Y and Xue, Z. (2021). An efficient and light weight Approach for ID based on Knowledge Distillation. IEEE international Conference on Communication. https://doi. 10.1109/ICC42927.2021.9500574/.

[18] Dinh, M., Patel, M., Das, T., and Shukla, R.M. (2024). Small, but Mighty Lightweight machine Learning Intrusion Detection Framework for Vehicle Ad-hoc Network. 2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC) , India. https://doi:10.1109/AIc61668.2024.10795451/

[19] Stolz, C., Li, F., Zhang, J. (2024). Implementing Lightweight Intrusion Detection on resource-constrained Devices. 2024 Conference on Cyber Awareness and Research Symposium (CARS). https://doi.10.1109/CARs61786.2024.10778716.

[20] Pham, V., Seo, E and Chung, T-M. (2020). Lightweight Convolutional Neural Network Based Intrusion Detection System. Journal of Communications, 15(11);808-817.https://doi.10.12720/jcm.15.11.808-817.

[21] Aliyu, F., Sheltami, T., Deriche, M. (2022). Human Immune-Based Intrusion Detection and Prevention System for Fog Computing. Journal of Networking System and Management 30, 11 (2022). https://doi.org/10.1007/s10922-021-09616-6.

[22] Alzahrani, H., Sheltami, T., Barnawi, A., Imam, M., Yaser, A. (2024). A Lightweight Intrusion Detection System Using Convolutional Neural Network and Long Short-Term Memory in Fog Computing. Computers, Materials and Continua, 80(3), 4703–4728. https://doi.org/10.32604/cmc.2024.054203

[23] Samy, A., Yu, H. and Zhang, H. . (2020). Fog-based attack detection framework for internet of things using deep learning, IEEE Access, vol. 8, pp. 74571–74585. doi: 10.1109/ACCESS.2020.2988854.

[24] Khater, B. S., Washab, A.W.A., Idris, M.Y.I., Hussain, M. A., and Ibrahim, A. A. (2019). A lightweight perceptron-based intrusion detection system for fog computing,Applied Sciences ,9(1);178. doi: 10.3390/app9010178.

[25] Roy, S., Li, J., Choi, B., and Bai Y. (2022). A lightweight supervised intrusion detection mechanism for IoT Networks. Future Generation Computer Systems Future Generation Computer Systems 127 (2022) 276–285. https://doi.org/10.1016/j.future.2021.09.027

[26] Yuping Lai, Y., Yu, Y., Guan, W.,Luo, L., Fan, J., Zhou, N.and Pin, Y. (2024). A Lightweight Intrusion Detection System Using a Finite Dirichlet Mixture Model With Extended Stochastic Variational Inference, IEEE Transactions on Network and Service Management PP(99):1-1 DOI:10.1109/TNSM.2024.3391250.

[27] Pandey, V.K., Sahu, D., Prakash, S. et al. A lightweight framework to secure IoT devices with limited resources in cloud environments. Sci Rep 15, 26009 (2025). https://doi.org/10.1038/s41598-025-09885-0.

[28] Boukhamla, A and Gaviro, J.C. (2018). CICIDS2017 dataset: Performance improvements and validation as a robust intrusion detection system testbed, Int. Journal of Information Computer Security.

[29] Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., Al-Nemrat, A.and Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system, IEEE Access (2019) http://dx.doi.org/10.1109/ACCESS.2019. 2895334.

[30] Li, W., Yi, P., Wu, Y., pan, L and Li, J. (2014). A new intrusion detection system based on KNN classification algorithm in wireless sensor network, Journal of Electrical and Computer Engineering. (2014).

[31] http://dx.doi.org/10.1155/2014/240217.

[32] Moustafa, N and Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), 2015, http://dx.doi.org/10.1109/MilCIS.2015.7348942.

[33] Almomani, I. Al-Kasasbeh, B., and Al-Akhras, M. (2016). WSN-DS: A dataset for intrusion detection systems in wireless sensor networks, Journal of Sensors. http://dx.doi.org/10.1155/2016/4731953.

[34] Sethi, K., Madhav, Y.V., Kumar, R and Bera, P. (2021). Attention based multi-agent intrusion detection systems using reinforcement learning, Journal of Information and Security. Application. http://dx.doi.org/10.1016/j.jisa.2021.102923.

[35] Jouhari, M. and Guizani, M. (2024). Lightweight CNN-BiLSTM based Intrusion Detection Systems for Resource-Constrained IoT Devices. 2024 International Wireless Communications and Mobile Computing (IWCMC), Ayia Napa, Cyprus, 2024, pp. 1558-1563, doi: 10.1109/IWCMC61514.2024.10592352.

[36] Lee, J.H. and Park, K.H. (2021). GAN-Based imbalanced data intrusion detection system, Pers. Ubiquitous Computer http://dx.doi.org/10.1007/s00779.