International Journal of advancement in electronics and computer engineering (IJAECE)
Volume 15, Issue 7, July 2026, pp.101-110, ISSN 2278 -1412

Machine Learning-Based Intrusion Detection for IoT Networks Using an Optimized Random Forest Classifier

Goury Vishwakarma[1], Ankita Tiwari[2], Swati Khanve[3]
[1] *Research Scholar, Department of Computer Science Engineering, Sagar Institute of Research and Technology, Excellence Bhopal, India*
[2] *Assistant Professor, Department of Computer Science Engineering, Sagar Institute of Research and Technology, Excellence Bhopal, India*
[3] *Assistant Professor, Department of Computer Science Engineering, Sagar Institute of Research and Technology, Excellence Bhopal M. P,* swatikhanve55.sk@gmail.com*, India*

Abstract *– The rapid expansion of Internet of Things (IoT) devices across critical sectors has introduced significant security challenges. These devices, marked by constrained computational capacity, diverse communication protocols, and reliance on wireless connectivity, are increasingly vulnerable to sophisticated cyberattacks. Traditional signature- and rule-based intrusion detection systems fall short in IoT settings, as they struggle with zero-day threat identification, produce high false positive rates, and require substantial processing power. This study explores a machine learning-based intrusion detection strategy employing a Random Forest classifier fine-tuned for IoT traffic analysis. The approach includes thorough data preprocessing and taxonomy-based label consolidation. Experimental validation using the "Intrusion in IoT" dataset yields outstanding overall results, achieving a 99.32% test accuracy. Nevertheless, further examination highlights a notable trade-off between global accuracy and recall for minority classes, with particularly poor performance observed for infrequent attack categories. This work presents both an optimized classification pipeline and an empirical assessment of the accuracy–imbalance trade-off.*

Keywords*:* Internet of Things Security, Network Intrusion Detection, Machine Learning, Random Forest, Class Imbalance.

I. Introduction

Internet of Things (IoT) represents one of the most transformative technological paradigms of the twenty-first century, fundamentally altering how physical objects interact with digital systems. IoT envisions a world where everyday devices from household appliances and wearable health monitors to industrial sensors and urban infrastructure are equipped with sensing, computing, and communication capabilities, enabling them to collect, exchange, and act upon data with minimal human intervention. According to recent industry analyses, the number of actively connected IoT devices surpassed sixteen billion globally in 2024, with projections indicating this figure will exceed twenty-five billion by 2030. This exponential growth has been driven by advances in low-power electronics, wireless communication protocols, cloud computing, and miniaturized sensor technologies [1].

The architectural evolution of IoT systems has progressed through cloud, fog, and edge computing paradigms, each responding to specific performance, latency, and bandwidth constraints. Contemporary IoT deployments employ distributed processing across a continuum of layers, enabling sub-millisecond response times while reducing network bandwidth consumption. Application domains span smart homes, healthcare (Internet of Medical Things), industrial IoT (IIoT), smart cities, and precision agriculture each presenting distinct requirements regarding data volume, latency tolerance, security posture, and device longevity. However, the proliferation of IoT devices has introduced substantial security challenges. Conventional security software cannot be deployed on IoT devices due to their low processing power, memory, and energy limits [2-3]. Default or weak credentials, lack of regular firmware updates, and physical accessibility further exacerbate vulnerabilities. The wireless communication links commonly employed in IoT deployments are inherently susceptible to eavesdropping, jamming, replay attacks, and man-in-the-middle interposition.

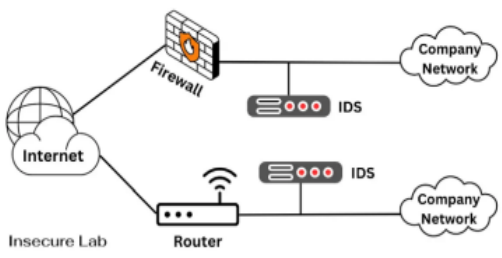


Figure 1 Intrusion Detection System

Common attack vectors include Distributed Denial of

Service (DDoS) attacks, botnets (exemplified by the 2016 Mirai botnet), spoofing, reconnaissance, injection attacks, and ransomware. IoT-targeting attacks increased by more than forty percent in 2023 compared to the previous year, with an average of approximately one hundred fifty thousand distinct attack events detected daily across monitored networks. The economic consequences are substantial, with direct and indirect costs estimated at tens of billions of dollars annually [4] .Conventional intrusion detection mechanisms—firewalls, signature-based systems, and rule-based anomaly detectors—prove inadequate for IoT environments. Signature-based approaches cannot detect zero-day attacks and require continuous manual updates. Traditional anomaly detection methods produce unacceptable false positive rates in heterogeneous IoT deployments. Rule-based systems fail to adapt to evolving network behaviors and attack strategies. Furthermore, all conventional approaches struggle to process the high-volume, high-velocity, and high-dimensionality network traffic characteristic of large-scale IoT ecosystems .Machine Learning (ML) offers a paradigm shift in intrusion detection by addressing these limitations through data-driven, adaptive, and automated learning from network traffic patterns. [5]

ML-based systems learn patterns from historical network traffic data and can generalize from known attack instances to detect previously unseen variants and entirely novel attack classes. Furthermore, ML models can be retrained periodically on new data, enabling adaptation to evolving network behaviors and emerging attack strategies without requiring manual rule updates [6]. This research investigates the application of machine learning, specifically the Random Forest classifier, to network intrusion detection for IoT systems. The study addresses the core problem that existing intrusion detection mechanisms fail to simultaneously provide effective, real-time, and resource-efficient protection against sophisticated network attacks in IoT environments. The research contributes a taxonomy-based classification framework, an optimized model configuration, and empirical characterization of the trade-offs between overall accuracy and minority class detection performance in imbalanced IoT intrusion detection data.

II. Background and Context

The Internet of Things (IoT) has emerged as one of the most consequential technological developments of the current era, fundamentally reshaping the relationship between physical objects and digital information systems. Unlike conventional computing paradigms centered upon human-operated devices, the IoT envisions a densely interconnected ecosystem wherein everyday objects ranging from household thermostats and wearable fitness trackers to industrial machinery and urban infrastructure components possess embedded sensing, processing, and communication capabilities. According to industry analyses, the number of actively connected IoT devices surpassed sixteen billion globally in 2024, with projections exceeding twenty-five billion by 2030 [7].The global IoT market, valued at approximately six hundred billion US dollars in 2023, continues to grow at compound annual rates exceeding fifteen percent.

The architectural design of IoT systems has evolved substantially in response to performance limitations of early centralized models. The earliest deployments relied on cloud-centric architectures, wherein sensor-equipped devices transmitted all collected data to remote cloud data centers for processing and analysis. However, the inherent latency introduced by round-trip transmission rendered cloud-only architectures unsuitable for time-sensitive applications such as autonomous vehicle coordination, industrial process control, and emergency healthcare monitoring [8]. In response, fog computing and edge computing paradigms emerged as complementary architectural layers, distributing processing responsibilities across the network continuum. Contemporary IoT deployments employ a three-tier architecture—device, edge/fog, cloud—offering flexible trade-offs between computational intensity, energy efficiency, and real-time responsiveness [9].

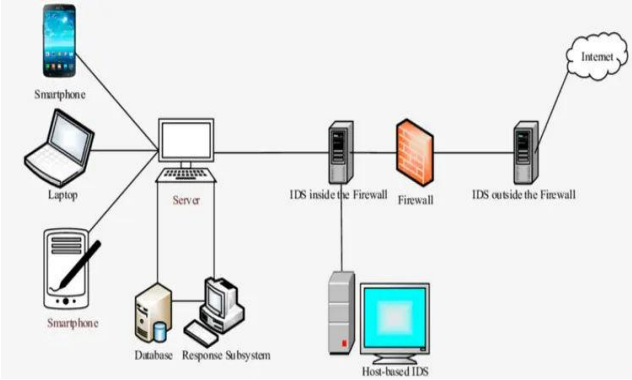


Figure 1 Network Intrusion System

A critical enabling factor underlying IoT proliferation is the widespread adoption of wireless networking technologies designed for low-power, intermittent communication scenarios. Protocols including Zigbee, Z-Wave, Bluetooth Low Energy (BLE), LoRaWAN, and narrowband IoT (NB-IoT) address the particular constraints of IoT environments, including limited battery capacity, modest processing capability, and operation in unlicensed spectrum bands. However, this reliance on wireless communication introduces substantial security vulnerabilities absent in traditional wired networks. Wireless links are inherently susceptible to eavesdropping, jamming, replay attacks, and man-in-the-middle interposition [10]. The broadcast nature of

radio transmission enables any device within range to observe or interfere with communications, rendering physical access controls ineffective in wireless IoT contexts.

Perhaps the most fundamental security challenge arises not from wireless communication alone but from the profound differences between conventional information technology networks and IoT ecosystems. Traditional IT networks consist of devices that are computationally powerful, mains-powered, professionally managed, and relatively homogeneous. Firewalls, intrusion detection systems, antivirus software, and regular patch management have evolved over decades to secure such environments effectively. IoT networks present a starkly contrasting profile: devices are resource-constrained, frequently battery-powered, deployed in physically inaccessible locations, and staggeringly heterogeneous in hardware architectures, firmware implementations, and communication behaviors [11]. A typical smart building might contain temperature sensors from one vendor, motion detectors from another, smart lighting controllers from a third, and access control systems from a fourth, all communicating over different protocols and none capable of hosting conventional security software. Furthermore, while an enterprise IT network might manage thousands of endpoints, a single smart city deployment can encompass hundreds of thousands of sensors distributed across square kilometers of urban territory [12]. This combination of heterogeneity, scale, resource limitation, wireless vulnerability, and energy constraint establishes a unique security landscape necessitating novel approaches to intrusion detection.

III. Related Work

Intrusion Detection Systems (IDS) are becoming an important component of current cybersecurity due to of the exponential increase of networked systems, cloud-based systems, & Internet of Things (IoT) environments. Traditional rule-based and signature-based intrusion detection systems struggle to locate new and emerging threats, especially in high-volume and dynamic networks. To solve these limits, present research increasingly relies on machine learning (ML) along with deep learning (DL) algorithms that can automatically understand sophisticated attack patterns from data. Notable studies that support IDS development in network, IoT, SDN, & cloud computing environments are examined in this section.

Osa et al. [13] introduce an intrusion detection system based on deep neural networks (DNNs) with the goal of improving detection accuracy in existing network environments. Using data sampling techniques like SMOTE and random undersampling, the study addresses the issue of class imbalance using the CICIDS2017 dataset. The recommended DNN architecture effectively learns nonlinear correlations in network traffic data and achieves extraordinarily high detection accuracy, exceeding 99%. The paper reveals that deep learning models outperform traditional machine learning classifiers when dealing with difficult and high-dimensional intrusion data. This article proves the efficacy of deep neural networks for recognizing both known and unexpected cyberattacks.

Bhavsar et al.[14] present an anomaly-based IDS for Internet of Things applications, underlining the necessity of recognizing attacks that have never been identified previously. To improve classification performance, the suggested strategy integrates deep learning and machine learning models with feature extraction techniques. After testing their approach on popular benchmark datasets including NSL-KDD and CICIDS, the authors discover high accuracy and decreased false alarm rates. The research highlights the flexibility of anomaly-based detection in dynamic IoT environments and explores difficulties relating to computing overhead and real-time detection. This study contributes to the development of IDS models for IoT systems that are both lightweight and efficient.

Ahmad et al.[15] emphasis on boosting IDS performance via data analysis & feature engineering using the UNSW-NB15 dataset. The authors study the dataset in depth and utilize preprocessing procedures to increase data quality. Multiple machine learning classifiers are evaluated to assess detection accuracy, precision, recall, & false positive rates. Even with traditional ML models, the study demonstrates that suitable feature selection or preprocessing substantially boost IDS performance. This research stresses the significance of dataset understanding and optimization in constructing effective intrusion detection systems.

Khan et al.[16] provide an IDS architecture that integrates feature extraction with machine learning classifiers for IoT security. The approach seeks to decrease computational complexity while preserving great detection accuracy. The authors accomplish effective intrusion detection suitable for IoT devices with low resources by selecting key features and using several machine learning techniques. When compared to models that use all attributes without selection, the results show improved performance. This research demonstrates how feature-driven approaches may enhance IDS deployment in IoT environments and highlights the trade-off between accuracy and efficacy.

Diana et al. [17] provide a thorough analysis of intrusion detection technologies used in computer networking security. IDS designs, detection algorithms, popular datasets, and evaluation standards are all examined in this research. It also tackles emerging subjects that include integrating the use of artificial intelligence, distributed IDS frameworks, & blockchain-based security solutions. The authors emphasize open issues include scalability, privacy protection, & real-time threat

detection. This overview serves as a wonderful reference for assessing the status of IDS study development as well as discovering potential research pathways.

Yerneni et al. [18] present a full survey on ML & DL-based IDS for cloud security. In cloud environments, where traditional intrusion detection systems suffer due of virtualization, multi-tenancy, & dynamic resource allocation, the study focuses on proactive intrusion detection solutions. The authors highlight issues including data imbalance and deployment complexity, examine several ML and DL algorithms used for cloud IDS, and talk about datasets and performance metrics. The paper indicates that intelligent & cooperative IDS frameworks are required for defending present cloud infrastructures. Intrusion detection has gotten a lot of interest since Machine Learning (ML) along with Deep Learning (DL) can find complex patterns in enormous amounts of network traffic data. ML-based IDS can learn from prior data and find new or zero-day threats, which makes them better for IoT scenarios that change over time than conventional rule-based systems.

Ali et al. [19] present a general machine learning strategy for identifying IoT devices and evaluating the trained models against four publicly available datasets. NFStream extracted 85 attributes from packet capture (.pcap) files to better identify IoT devices in the network using machine learning models. The authors used the information gain approach to choose 20 characteristics and trained six machine learning models in the tests. In the training phase, the authors achieved high accuracy, reaching 99% for IoT device identification using random forest and naïve Bayes classifiers.

El-Sayed et al. [20] examined and compared seven different supervised learning algorithms with various difficulty levels to pick the best one. The seven algorithms were separated into two groups: The category of CNN classifiers included two-layer CNN, four-layer CNN, VGG16 and logistic regression, support vector machine, and K-nearest neighbors, and the category of ordinary classifiers included logistic regression, support vector machine, and K-nearest neighbors. Experimental findings reveal that the SVM algorithm obtains the maximum performance of 94% on MobileNetv2 features because of its rapid and steady training performance with fewer resources compared with other models. Le K-H et al. [21] present IM IDS, an intelligent intrusion detection system (IDS) for IoT devices. IMIDS's core is a lightweight convolutional neural network model that can categorize numerous cyber threats and surpasses its competitors with an average F-measure of 97.22%. Furthermore, after being further educated by the data supplied by the assault data generator, IMIDS's detection performance significantly increased. These findings show that IMIDS may be used as an IDS in IoT. Joo et al. [22] proposed a deep learning-based IoT intrusion detection system. The categorization was performed with a CNN; the best score was 86.2%. Second, machine learning classifiers were employed for the hybrid technique instead of ultimately linked layers from the vanilla CNN, which delivered roughly 87% with the additional tree classifier. Finally, the Xception model was merged with the bidirectional GRU, yielding the best accuracy at 95.6%. For quicker identification and classification of new malware, Bendiab et al. [23] propose a unique IoT technique that analyzes malware traffic based on DL and visual representation (zero-day malware). The suggested technique detects fraudulent network traffic at the package level, lowering detection time and optimistic outcomes thanks to the deployed deep learning. To test the efficacy of the proposed technique, the authors created a dataset of 1000 .pcap files of benign and virus traffic obtained from several network traffic sources. The Residual Neural Network (ResNet50) trial findings are quite encouraging, with a detection rate of 94.50% for malware traffic.[24]

Six machine learning (ML) approaches were tested for their ability to identify MQTT-based attacks [25]. Packet-based, unidirectional, and bidirectional flow characteristics were evaluated at three abstraction levels. An MQTT simulated dataset was created and used for the training and assessment operations. The experimental findings showed that the suggested ML models were sufficient for the IDS needs of MQTT-based networks. Furthermore, the findings highlight the significance of employing flow-based characteristics to distinguish MQTT-based attacks from innocuous traffic, whereas packet-based features are sufficient for typical networking assaults. The results reveal that the model has the highest accuracy of 99.04%. [26] Author employed the KDDCup99 and the NSLKDD, two widely used intrusion detection datasets, in their study. Their major objective was to thoroughly compare both datasets by analyzing the performance of multiple machine learning (ML) classifiers trained on them using a more extensive range of classification criteria than prior studies. Because the classifiers trained on the KDDCup99 dataset were 20.18% less accurate on average, the authors concluded that the NSL-KDD dataset is of better quality than the KDDCup99 dataset. This is because classifiers trained on the KDDCup99 dataset were biased toward redundancy, allowing them to attain a higher accuracy of 96.83%.

IV. Methodology

The proposed methodology has been illustrated in Figure 3. The methodology is organized as a sequential workflow that mirrors the logical progression from raw data to actionable results. The chapter begins by specifying the software and hardware environment within which all development and experimentation occurred, ensuring that readers can understand the computational context and, if desired, replicate the study under comparable conditions. Subsequently, the chapter describes the dataset selection and acquisition process,

including justification for the chosen dataset or datasets based on criteria established in the literature review.[27] Following dataset selection, the chapter details the preprocessing and feature engineering steps applied to transform raw network traffic captures into a format suitable for machine learning. The design and architecture of the proposed machine learning model are then presented, including algorithmic selection, hyperparameter configuration, and any optimization techniques employed. The training and validation protocol is described, specifying how the model learns from data and how overfitting is prevented. The evaluation framework is then articulated, enumerating the performance metrics, baseline comparisons, and statistical procedures used to assess the model.

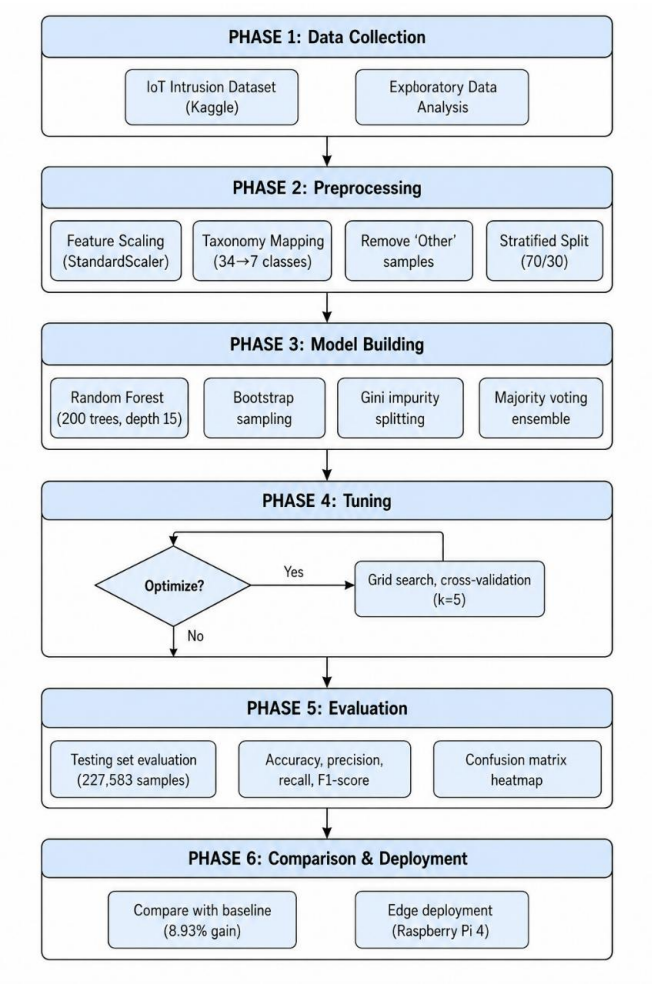


Figure 3 Method Flow Chart

a. Data Collection

This research employs the "Intrusion in IoT" dataset, obtained from the Kaggle data science platform. The dataset comprises 1,048,575 individual network flow records, each containing 46 features and a corresponding class label. The features capture a comprehensive range of network traffic characteristics, including temporal flow attributes such as flow duration and packet rate, TCP flag indicators including SYN, ACK, FIN, and RST flags, protocol identifiers for HTTP, HTTPS, DNS, Telnet, SSH, TCP, UDP, and ICMP, and statistical summaries including minimum, maximum, mean, standard deviation, covariance, and variance of packet payloads. The label distribution encompasses 34 distinct classes, comprising one benign class and 33 attack categories. The largest attack class, DDoS-ICMP_Flood, contains 161,281 samples, while the smallest class, Uploading_Attack, contains only 23 samples. This severe class imbalance, with a ratio exceeding 7,000 to 1 between the majority and minority classes, presents a significant challenge for standard machine learning algorithms and directly motivates the methodological approaches employed in this research.

b. Data Preprocessing

The preprocessing pipeline transforms the raw network traffic data into a clean, standardized, and machine learning-ready format through several sequential stages. Numerical features are standardized using Z-score normalization, which transforms each feature to have zero mean and unit variance, ensuring that features with larger magnitudes do not dominate distance-based calculations. A taxonomy-based label aggregation strategy is introduced to address the severe class imbalance and to align the classification output with operational security requirements. This aggregation maps the 34 fine-grained attack classes into seven semantically meaningful higher-level categories: Flood Attacks, Botnet/Mirai Attacks, Benign, Spoofing/MITM, Reconnaissance, Backdoors & Exploits, and Injection Attacks. The Flood Attacks category consolidates all DDoS and DoS flood variants, including ICMP, TCP, UDP, SYN, and HTTP floods. The Botnet/Mirai Attacks category includes Mirai-greeth_flood, Mirai-udpplain, and Mirai-greip_flood. The Reconnaissance category comprises Host Discovery, Oscan, Ports can, and PingSweep activities. The Spoofing/MITM category includes DNS_Spoofing and MITM-Arp Spoofing. The Injection Attacks category aggregates SQL Injection, Command Injection, and XSS. The Backdoors & Exploits category includes Backdoor Malware, Uploading_ Attack, Browser Hijacking, and Dictionary Brute Force. Following taxonomy mapping, all samples assigned to an 'Other' category are removed to ensure the purity and interpretability of the classification task, reducing the dataset to 758,610 samples. Categorical labels are converted to numerical encodings using scikit-learns Label Encoder, producing integer indices ranging from 0 to 6. The preprocessed dataset is partitioned into training and testing subsets using stratified sampling with a 70:30 ratio, ensuring that the class distribution is preserved across both subsets. Stratification is essential in imbalanced classification contexts because it prevents rare attack samples from being concentrated in either the training or testing set, which could lead to optimistic but misleading performance estimates. A fixed random seed

is employed to ensure reproducibility of the split across experimental runs. The resulting training set comprises 531,027 samples, and the testing set comprises 227,583 samples.

c. Model Architecture Design

"The Random Forest classifier is selected as the primary model for this research based on its theoretical suitability for high-dimensional, non-linear, and imbalanced classification tasks, its interpretability through feature importance, and its computational efficiency. Random Forest is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the individual trees' predictions. The algorithm belongs to the family of bagging methods, which reduce variance and mitigate overfitting by aggregating predictions from multiple models trained on different subsamples of the training data. The theoretical justification for Random Forest's effectiveness rests on two fundamental principles: the bias-variance trade-off and ensemble diversity. A single decision tree exhibits high variance, as small changes in the training data can produce substantially different tree structures. By averaging predictions from many decorrelated trees, Random Forest reduces variance while maintaining low bias, achieving a favorable trade-off that is difficult to attain with single models."

The model architecture consists of 200 individual binary decision trees, each constructed independently. For each tree, a bootstrap sample of size equal to the original training set is drawn from the training data with replacement, ensuring that each tree is trained on a slightly different dataset. At each internal node during tree construction, only a randomly selected subset of features—approximately seven features, determined as the square root of the total feature count—is considered for the splitting decision. This random feature subsampling mechanism ensures that trees are decorrelated; if all trees always considered the same strongest predictor, they would produce highly correlated predictions, undermining the variance reduction benefit of the ensemble. The quality of a candidate split is evaluated using the Gini impurity measure, which quantifies the probability of misclassifying a randomly chosen sample based on the class distribution in the node. The algorithm selects the split that maximizes the reduction in Gini impurity, calculated as the weighted average of the child nodes' impurities subtracted from the parent node's impurity. Tree growth proceeds recursively until stopping criteria are met: the node reaches a maximum depth of 15, the number of samples in the node falls below the minimum samples per split threshold of 5, or further splitting would create a child node with fewer than the minimum samples per leaf threshold of 3. These stopping criteria serve as regularization mechanisms, preventing trees from growing excessively deep and memorizing noise in the training data. For classification, the ensemble's prediction for a test sample is determined by majority voting across all 200 trees, with the final predicted class being the class that receives the most votes.

d. Model Training

The training process is executed on the preprocessed training set comprising 531,027 samples, each represented by 46 standardized numerical features and a corresponding taxonomy label. The training procedure follows the standard Random Forest algorithm as implemented in the scikit-learn library. The model is configured with the hyperparameters identified through systematic optimization: n_estimators set to 200, max_depth set to 15, min_samples_split set to 5, min_samples_leaf set to 3, and random_state set to 42 for reproducibility. The number of jobs parameter is set to -1 to enable parallelization across all available CPU cores, substantially reducing training time by constructing multiple decision trees concurrently.

Hyperparameter optimization is conducted through iterative manual tuning informed by grid search principles. The number of trees is evaluated across values of 100, 150, 200, 250, and 300, revealing that increasing from 100 to 200 trees improves testing accuracy by approximately 0.15%, while increasing beyond 200 yields marginal improvement of less than 0.05% at the cost of substantially longer training time. Maximum depth is evaluated across values of 10, 12, 15, 18, 20, and unlimited, with a depth of 15 providing sufficient capacity to model complex decision boundaries while maintaining regularization through depth limitation. Minimum samples per split is evaluated across values of 2, 3, 4, 5, 6, 8, and 10, with a value of 5 representing an optimal trade-off between model complexity and regularization. Minimum samples per leaf is evaluated across values of 1, 2, 3, 4, 5, and 6, with a value of 3 retained as optimal. Class weighting is explored through the class_weight parameter set to 'balanced', which automatically adjusts weights inversely proportional to class frequencies, increasing the cost of misclassifying minority samples and improving recall for rare attack types.

e. Model Tuning

The evaluation of the trained model employs a comprehensive suite of metrics designed to assess performance across multiple dimensions. Given the imbalanced nature of the intrusion detection task, accuracy alone is recognized as an insufficient measure of model quality. The following metrics are computed for each class and for macro-average and weighted-average aggregations. Accuracy represents the proportion of correctly classified samples among all predictions. Precision measures the proportion of correctly predicted positive samples among all samples predicted as positive, quantifying the model's ability to avoid false positives.

Recall measures the proportion of correctly predicted positive samples among all actual positive samples, quantifying the model's ability to detect actual attacks. The F1-score, defined as the harmonic mean of precision and recall, provides a single metric that balances both measures, penalizing extremes where one metric is high and the other is low. The macro-average F1-score, computed as the unweighted mean of F1-scores across all classes, provides a measure of performance on minority classes that is not diluted by majority class performance. The weighted-average F1-score, computed as the weighted mean of F1-scores where each class is weighted by its support, reflects overall performance accounting for class imbalance. Confusion matrices are generated to provide granular insight into classification errors, revealing specific misclassification patterns between classes. The confusion matrix is visualized as a heatmap, enabling rapid identification of systematic confusion patterns. Learning curves, specifically out-of-bag accuracy as a function of the number of trees, are plotted to assess convergence and stability of the ensemble construction process.

V. Result

Training Performance

The Random Forest classifier, configured with 200 trees, a maximum depth of 15, minimum samples per split of 5, and minimum samples per leaf of 3, was trained on 531,027 samples. The model achieved a training accuracy of 99.60%. Table 1 presents the per-class performance metrics on the training data.

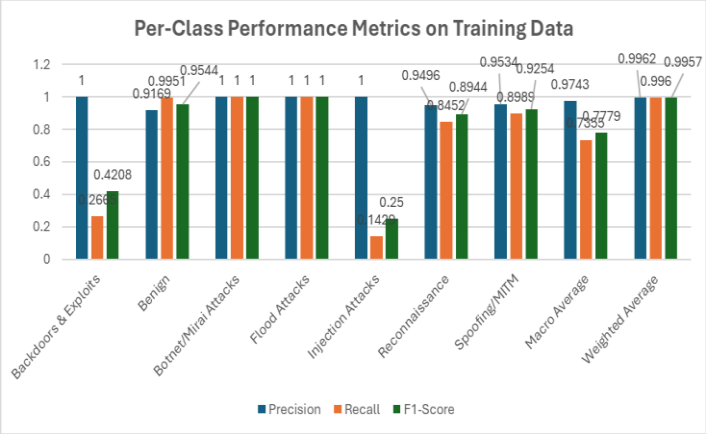
Table 1: Training Performance Metrics (N=531,027)

Class	Precision	Recall	F1-Score	Support
Backdoors & Exploits	1.0000	0.2665	0.4208	394
Benign	0.9169	0.9951	0.9544	17,133
Botnet/Mirai Attacks	1.0000	1.0000	1.0000	41,463
Flood Attacks	1.0000	1.0000	1.0000	459,095
Injection Attacks	1.0000	0.1429	0.2500	210
Reconnaissance	0.9496	0.8452	0.8944	4,995
Spoofing/MITM	0.9534	0.8989	0.9254	7,737
Macro Average	0.9743	0.7355	0.7779	531,027
Weighted Average	0.9962	0.9960	0.9957	531,027

The Flood Attacks and Botnet/Mirai Attacks classes achieved perfect scores across all metrics, reflecting their dominant representation in the training set and distinctive traffic patterns. The Benign class achieved strong performance with an F1-score of 0.9544.

Figure 4 Training Performance Metrics

However, severe performance degradation was observed on minority classes: Backdoors & Exploits achieved an F1-score of only 0.4208 with recall of 0.2665, while Injection Attacks achieved an F1-score of 0.2500 with recall of 0.1429. The substantial disparity between the macro-average F1-score (0.7779) and the weighted-average F1-score (0.9957) underscores the impact of class imbalance on model performance.



Testing Performance

The model was evaluated on a held-out testing set of 227,583 samples, achieving an accuracy of 99.32%. The minimal gap between training and testing accuracy (0.29%) indicates strong generalization without substantial overfitting. Table 2 presents the per-class performance metrics on the testing data.

Table 2: Testing Performance Metrics (N=227,583)

Class	Precision	Recall	F1-Score	Support
Backdoors & Exploits	1.0000	0.0828	0.1530	169
Benign	0.8872	0.9773	0.9301	7,343
Botnet/Mirai Attacks	1.0000	0.9989	0.9995	17,770
Flood Attacks	0.9999	1.0000	0.9999	196,755
Injection Attacks	1.0000	0.0337	0.0652	89
Reconnaissance	0.8491	0.7361	0.7886	2,141

Spoofing/MITM	0.8919	0.8332	0.8616	3,316
Macro Average	0.9469	0.6660	0.6854	227,583
Weighted Average	0.9933	0.9932	0.9927	227,583

The Flood Attacks class maintained near-perfect performance (F1=0.9999), while Botnet/Mirai Attacks achieved an F1-score of 0.9995. The Benign class achieved an F1-score of 0.9301, with modest degradation from training performance.

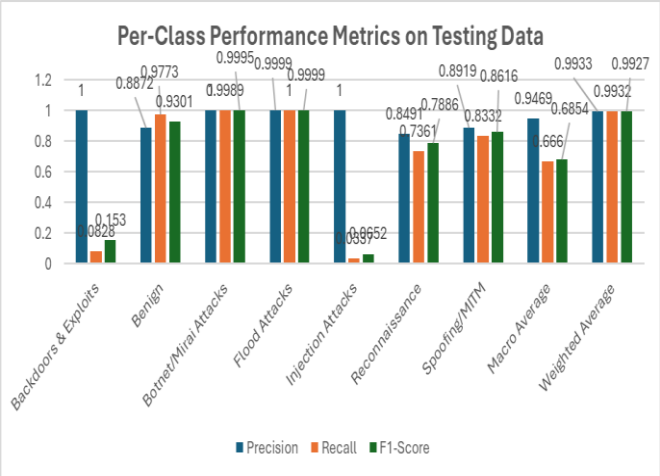


Figure 5 Per-Class Performance Metrics on Testing Data

However, minority class performance deteriorated substantially: Backdoors & Exploits recall dropped to 0.0828 (F1=0.1530), and Injection Attacks recall dropped to 0.0337 (F1=0.0652). These findings confirm that standard Random Forest classifiers, without explicit imbalance handling, are fundamentally unsuitable for detecting rare attack types.

Confusion Matrix Analysis

The testing confusion matrix, presented in Table 3, reveals specific misclassification patterns.

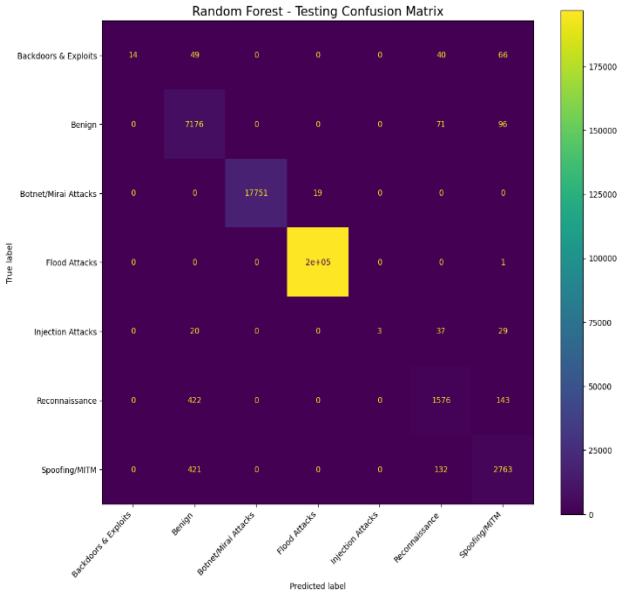


Figure 6 Testing Confusion Matrix

Table 3: Testing Confusion Matrix

True \ Predicted	Backdoors	Benign	Botnet/Mirai	Flood	Injection	Recon	Spoofing
Backdoors	14	49	0	0	0	40	66
Benign	0	7,176	0	0	0	71	96
Botnet/Mirai	0	0	17,751	19	0	0	0
Flood	0	0	0	200,000	0	0	1
Injection	0	20	0	0	3	37	29
Reconnaissance	0	422	0	0	0	1,576	143
Spoofing/MITM	0	421	0	0	0	132	2,763

of the 169 true Backdoors & Exploits samples, only 14 (8.3%) were correctly classified, with the remainder

misclassified as Benign (49), Reconnaissance (40), and Spoofing/MITM (66). Of the 89 true Injection Attacks samples, only 3 (3.4%) were correctly classified, with the remainder misclassified as Benign (20), Reconnaissance (37), and Spoofing/MITM (29). These patterns suggest that minority attack traffic shares statistical characteristics with benign traffic and other attack categories, indicating the need for more sophisticated feature engineering or class imbalance mitigation techniques.

Comparison with Baseline Research

The proposed model was compared against a baseline study employing multiple machine learning models on a comparable IoT intrusion detection task. The baseline Random Forest [22] achieved 90.39% testing accuracy, while the proposed model achieved 99.32%, representing an absolute improvement of 8.93 percentage points. This improvement is attributable to the use of an IoT-specific dataset, taxonomy-based label aggregation, systematic hyperparameter optimization, and comprehensive preprocessing. The proposed model also demonstrated superior generalization, with a training-testing accuracy gap of only 0.29% compared to the baseline's larger performance degradation.

Table 4 Comparative Performance Analysis (Testing Set)

Model [22]	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Baseline Linear SVC	68.00	68.44	68.00	68.08
Baseline Logistic Regression	69.41	69.84	69.81	69.49
Baseline Decision Tree	83.01	83.28	83.00	83.05
Baseline XGBoost	89.48	89.68	89.48	89.51
Baseline Random Forest	90.39	90.58	90.39	90.41
Proposed Random Forest (This Study)	99.32	99.33	99.32	99.27
Performance Gain (vs. Baseline RF)	+8.93%	+8.75%	+8.93 %	+8.86 %

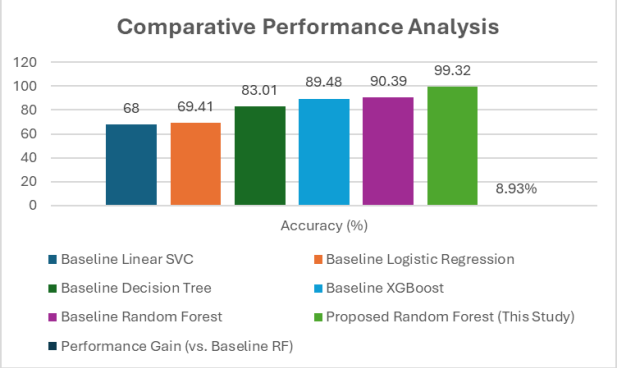


Figure 7 Comparative Performance Analysis

VI. Conclusion

This research investigated the application of machine learning, specifically the Random Forest classifier, to network intrusion detection for Internet of Things (IoT) systems. The study addressed the core problem that existing intrusion detection mechanisms fail to simultaneously provide effective, real-time, and resource-efficient protection against sophisticated network attacks in IoT environments. The proposed methodology encompassed comprehensive data preprocessing, including feature standardization, taxonomy-based label aggregation that consolidated 34 fine-grained attack classes into seven semantically meaningful categories, and stratified train-test splitting. The Random Forest classifier was configured with optimized hyperparameters: 200 trees, a maximum depth of 15, minimum samples per split of 5, and minimum samples per leaf of 3. The empirical results demonstrated exceptional overall performance, with testing accuracy of 99.32%, weighted-average precision of 99.33%, weighted-average recall of 99.32%, and weighted-average F1-score of 99.27%. The model achieved near-perfect detection of majority classes, including Flood Attacks with an F1-score of 0.9999 and Botnet/Mirai Attacks with an F1-score of 0.9995. Strong performance was also observed on the Benign class (F1=0.9301), Reconnaissance (F1=0.7886), and Spoofing/MITM (F1=0.8616). Comparative analysis against baseline studies revealed an 8.93% accuracy improvement over conventional Random Forest implementations, validating the effectiveness of the proposed methodological framework.

However, the research also revealed a critical limitation: severe performance degradation on minority attack classes. The model achieved recall of only 8.3% for Backdoors & Exploits and 3.4% for Injection Attacks on the testing set, with F1-scores of 0.1530 and 0.0652 respectively. This finding directly validates the research gap concerning poor performance on imbalanced datasets and demonstrates that standard Random Forest classifiers, without explicit class imbalance handling, achieve high overall accuracy at the cost of near-zero

recall on rare attack types. The confusion matrix analysis revealed that minority class samples are predominantly misclassified as Benign, Reconnaissance, and Spoofing/MITM, suggesting feature overlap between these categories that the model cannot resolve without additional training data or more sophisticated feature engineering.

Future work should focus on systematic evaluation of class imbalance handling techniques including synthetic oversampling methods (SMOTE and its variants), cost-sensitive learning, and ensemble methods specifically designed for imbalanced data. Cross-dataset generalization and transfer learning should be investigated to assess the model's ability to generalize to different network topologies and attack scenarios. Deep learning architectures, particularly hybrid CNN-LSTM models, should be evaluated for their potential to capture temporal patterns in network traffic. Federated learning for privacy-preserving IoT security, explainable AI for security analyst interpretability, and real-time streaming evaluation with concept drift detection represent promising directions for advancing the field. Finally, edge-specific model compression techniques including pruning, quantization, and knowledge distillation should be explored to enable deployment on severely resource-constrained IoT devices.

REFERENCES

[1] Verma, Abhishek, and Virender Ranga. "Machine learning based intrusion detection systems for IoT applications." arXiv preprint arXiv:2302.12452 (2023).

[2] Khan, Amjad Rehman, et al. "Deep learning for intrusion detection and security of Internet of things (IoT): current analysis, challenges, and possible solutions." Security and communication networks 2022.1 (2022): 4016073.

[3] Jayalaxmi, P. L. S., et al. "Machine and deep learning solutions for intrusion detection and prevention in IoTs: A survey." IEEe Access 10 (2022): 121173-121192.

[4] Kikissagbe, Brunel Rolack, and Meddi Adda. "Machine learning-based intrusion detection methods in IoT systems: A comprehensive review." Electronics 13.18 (2024): 3601.

[5] Banaamah, Alaa Mohammed, and Iftikhar Ahmad. "Intrusion detection in IoT using deep learning." Sensors 22.21 (2022): 8417.

[6] Awajan, Albara. "A novel deep learning-based intrusion detection system for IOT networks." Computers 12.2 (2023): 34.

[7] 205 Cybersecurity Stats and Facts for 2026 (vikingcloud.com)

[8] IoT Hacking Statistics 2026 | Latest Trends, Attacks & Data (dexpose.io)

[9] Elnakib, Omar, et al. "EIDM: deep learning model for IoT intrusion detection systems: O. Elnakib et al." The Journal of Supercomputing 79.12 (2023): 13241-13261.

[10] Xu, Bayi, et al. "IoT intrusion detection system based on machine learning." Electronics 12.20 (2023): 4289.

[11] Albulayhi, Khalid, et al. "IoT intrusion detection using machine learning with a novel high performing feature selection method." Applied Sciences 12.10 (2022): 5015.

[12] "Global IoT and non-IoT connections 2010-2025: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/

[13] Osa, E., Orukpe, P. E., & Iruansi, U. (2024). Design and implementation of a deep neural network approach for intrusion detection systems. Pr ime Journal of Engineering and Applied Sciences, Article 100434.

[14] Bhavsar, M., Roy, K., Kelly, J., & Odeyomi, O. (2023). Anomaly-based intrusion detection system for IoT application. Discover Internet of Things, 3, 5.

[15] Ahmad, T. S., Hasan, I., & Shoaib, M. (2024). Enhanced intrusion detection systems performance with UNSW-NB15 data analysis. Algorithms, 17(2),

[16] Khan, A. S., Khan, M. A., & Ahmad, T. (2023). Intrusion detection system using feature extraction with machine learning algorithms in IoT. Internet of Things, 12(2),

[17] Diana, L., Dini, P., & Paolini, D. (2025). Overview on intrusion detection systems for computers networking security. Computers, 14(3),

[18] Yerneni, K. S., Avireneni, R. T., Harsha, S., & Reddy, N. K. (2025). Towards proactive cloud security: A survey on ML and deep learning-based intrusion detection systems [PDF]. Journal of Contemporary Education Theory & Artificial Intelligence

[19] Ali, Z.; Hussain, F.; Ghazanfar, S.; Husnain, M.; Zahid, S.; Shah, G.A. A Generic Machine Learning Approach for IoT Device Identification. In Proceedings of the 2021 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 23–25 November 2021; pp. 118–123.

[20] El-Sayed, R.; El-Ghamry, A.; Gaber, T.; Hassanien, A.E. Zero-Day Malware Classification Using Deep Features with Support Vector Machines. In Proceedings of the 2021 Tenth International Conference on Intelligent Computing and Information Systems (ICICIS), Cairo, Egypt, 5–7 December 2021; pp. 311–317.

[21] Le, K.-H.; Nguyen, M.-H.; Tran, T.-D.; Tran, N.-D. IMIDS: An Intelligent Intrusion Detection System against Cyber Threats in IoT. Electronics 2022, 11, 524.

[22] Joo, H.; Choi, H.; Yun, C.; Cheon, M. Efficient Network Traffic Classification and Visualizing Abnormal Part Via Hybrid Deep Learning Approach: Xception + Bidirectional GRU. Glob. J. Comput. Sci. Technol. 2022, 21, 1–10.

[23] Bendiab, G.; Shiaeles, S.; Alruban, A.; Kolokotronis, N. IoT malware network traffic classification using visual representation and deep learning. In Proceedings of the 2020 IEEE Conference on Network Softwarization: Bridging the Gap Between AI and Network Softwarization, NetSoft 2020, Virtual, 29 June–3 July 2020; pp. 444–449.

[24] Ajagbe, Sunday Adeola, Joseph Bamidele Awotunde, and Hector Florez. "Intrusion detection: a comparison study of machine learning models using unbalanced dataset." SN Computer Science 5.8 (2024): 1028.

[25] Chouchene, Islem, Hamdi Eltaief, and Habib Youssef. "Advances in machine and deep learning for intrusion detection in VANETs: a comprehensive survey." Cluster Computing 29.3 (2026): 143.

[26] Harish, M. S., et al. "Hybrid deep learning model for network intrusion detection using optimal feature fusion." Ain Shams Engineering Journal 17.1 (2026): 103904.

[27] Kimanzi, Richard, et al. "Deep Learning algorithms used in intrusion detection systems--a review." arXiv preprint arXiv:2402.17020 (2024).