Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Research Article

# IoT intrusion detection system based on machine learning and deep learning

[1]Karrar Majid Jasim
Informatic Institute for Postgraduate Studies
University of Information Technology and Communications
Baghdad, Iraq
karrarmajidjasim@gmail.com
Ms202310757@iips.edu.iq

[2]A.P.Dr. Joolan Rokan Nayef
Informatic Institute for Postgraduate Studies
University of Information Technology and Communications
Baghdad, Iraq
newjolan@gmail.com
dr.jolan_alkhazraji@iips.edu.iq

ARTICLE INFO
Article History
Received: 09/02/2025
Accepted: 02/04/2025
Published: 05/06/2025


## ABSTRACT

The proliferation of Internet of Things IoT devices has amplified cybersecurity challenges, necessitating robust Intrusion Detection Systems IDS to safeguard against threats such as botnets and Distributed Denial-of-Service DDoS attacks. This paper evaluates the performance of Machine Learning ML and Deep Learning DL models on two benchmark datasets, BoT-IoT and CIC-IDS2017, to develop efficient IDS. Among ML models, XGBoost demonstrated the best performance, achieving 99.99% accuracy on BoT-IoT and 99.91% on CIC-IDS2017 with superior computational efficiency. For DL, Convolutional Neural Networks CNNs achieved 99.99% accuracy on BoT-IoT and 99.61% on CIC-IDS2017 with preprocessing, highlighting the critical role of data preparation. These findings underline the effectiveness of advanced ML/DL models and preprocessing techniques in enhancing IoT security, providing a pathway for real-time, scalable intrusion detection in IoT environments.

***Keywords:*** *IoT Security; Intrusion Detection; Machine Learning; Deep Learning.*

## 1. INTRODUCTION

Information systems require robust protection from unauthorized access, misuse, and errors a concept central to cybersecurity, which safeguards data, services, and infrastructure [1]. Among the many innovations shaping our world, the IoT stands out as a transformative network of interconnected devices. By enabling smart environments like cities and healthcare, IoT improves connectivity, but also introduces significant vulnerabilities due to limited device resources and inadequate standardization [2]. The rapid expansion of IoT has heightened cybersecurity concerns, drawing increased attention from academia and industry to detect and prevent potential attacks [3].

A prominent threat within this landscape is botnets—networks of compromised devices controlled by attackers to execute harmful activities such as DDoS attacks, spam distribution, and data theft. These botnets, managed through command-and-control servers, disrupt online services and infrastructure at an alarming scale [4]. To counteract such threats, Intrusion Detection Systems IDS are deployed to monitor for security breaches. IDS employ various methods, such as signature-based detection for known threats or specification-based approaches to analyze traffic behavior, though improving accuracy and reducing false alarms remain ongoing challenges [5].

Advancements in technology have further empowered defense mechanisms. Machine learning, through data-driven models, detects anomalies by identifying unusual patterns indicative of intrusions. Meanwhile, deep learning, a subset of machine learning, enhances detection accuracy with techniques like CNNs and autoencoders, automating analyses and uncovering IoT software vulnerabilities. Together, these innovations are pivotal in addressing the growing landscape of cyber threats [5].

## 2. EVALUATION METRICS

Performance for machine learning models KNN, SVM, RF, XGBoost and deep learning model CNN for intrusion pattern classification in the BoT-IoT and CIC-IDS2017 datasets is measured in terms of the following metrics

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

(accuracy, recall, F1-score and precision). The accuracy of classification models in intrusion detection systems is evaluated on the basis of four key metrics: Accuracy, Precision, Recall, and F1-score. Accuracy provides an overall idea of how many instances are correctly classified. However, in imbalanced datasets like BoT-IoT and CIC-IDS2017, accuracy can be misleading. Precision is crucial in minimizing false alarms by measuring the ratio of true attacks to all the predicted attacks. Recall determines the accuracy with which the model detects actual attacks, and it is of essential importance in security-related applications in which undetected intrusions have disastrous repercussions. F1-score, the harmonic mean between precision and recall, gives an overall measure where both false negatives and false positives need to be managed. Combined, these present a better quantification of the performance of a model in realistic applications in the field of cybersecurity. The equations defining these metrics are given by [14]:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (3)$$

$$\text{F1score} = 2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}\right) \quad (4)$$

Here, TN, TP, FN, and FP denote true negatives, true positives, false negatives, and, false positives respectively. These metrics will provide an overall review of the model's performance with respect to intrusion pattern detection.

## 3. RELATED WORKS

The rapid growth of IoT devices has amplified cybersecurity challenges, particularly with botnet-driven DDoS attacks. Intrusion Detection Systems IDS, supported by advancements in Machine Learning ML and Deep Learning DL, are essential for mitigating these threats. Recent studies offer various approaches, methodologies, and datasets for addressing IoT botnets.

**[1]** Develops a hybrid IDS combining ML and DL, using the CICIDS2017 dataset and ensemble learning methods (e.g., Random Forest and CNN) to improve detection rates, introducing a feature reduction technique to enhance computational efficiency.

**[2]** Explores unsupervised methods like autoencoders and clustering on the BoT-IoT dataset for anomaly detection, addressing class imbalance with data augmentation.

**[3]** Proposes lightweight ML models (e.g., KNN, Decision Trees) for resource-constrained IoT devices, leveraging preprocessing techniques like Z-score normalization on CICIDS2017.

**[4]** Evaluates DL models CNN, and MLP for DDoS detection, emphasizing feature engineering on datasets like BoT-IoT and NSL-KDD.

**[5]** Introduces RDTIDS, a hybrid IDS combining decision trees and rule-based classifiers, achieving high detection rates on CICIDS2017 and BoT-IoT datasets.

**[6]** Presents CorrAUC, a feature selection algorithm that improves ML model performance on BoT-IoT by reducing computational complexity.

**[7]** Evaluates ML models on BoT-IoT, emphasizing its realistic attack scenarios and promoting it as a benchmark dataset for IDS research.

**[8]** Focuses on vehicular IoT networks, achieving over 99% accuracy with ensemble methods like Extreme Gradient Boosting on datasets like CICIDS2017 and CIC-DDoS-2019.

**[9]** Highlights the effectiveness of DL models CNN, and LSTM in detecting botnets and handling high-dimensional IoT traffic using CICIDS2017.

**[10]** Introduces BotDetectorFW, emphasizing feature selection to enhance IDS performance with minimal computational overhead on CICIDS2017.

**[11]** Presents MidSiot, a collaborative IDS framework using edge and cloud computing, achieving 99.68% accuracy on IoTID20, BoT-IoT, and CICIDS2017 datasets.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

These studies collectively emphasize the critical role of innovative IDS techniques, efficient feature selection, and advanced ML/DL methods in addressing IoT botnet threats.

## 4. DATASETS DESCRIPTION

Two different datasets were used in this paper these datasets are described in the following sections.

### 4.1 Bot-IoT

This research utilized the Bot-IoT dataset, developed by UNSW Canberra, to evaluate proposed models and schemes. The dataset combines simulated and genuine IoT attack traffic across six attack types using five IoT devices, recorded through the lightweight MQTT protocol. It contains 72 million records of legitimate and attack traffic, including DoS, DDoS, OS, data exfiltration, and service scans. For computational efficiency, a 5% subset of the dataset, divided into four CSV files with 3,668,045 attack records and 477 normal records, was used. The dataset's CSV format includes 46 features per row, making it versatile for various IoT systems as shown in Table 1.

TABLE I. Attack Types in Bot-IoT Dataset [9].

| *Category* | *Attack Type* | *Flow Count* | *Training* | *Test* |
|---|---|---|---|---|
| ***BENIGN*** | BENIGN | 9,543 | 7,634 | 1,909 |
| ***Information gathering*** | Service scanning | 1,463,364 | 117,069 | 29,267 |
| ***Information gathering*** | OS Fingerprinting | 358,275 | 28,662 | 7,166 |
| ***DDoS attack*** | DDoS TCP | 19,547,603 | 1,563,808 | 390,952 |
| ***DDoS attack*** | DDoS UDP | 18,965,106 | 1,517,208 | 379,302 |
| ***DDoS attack*** | DDoS HTTP | 19,771 | 1,582 | 395 |
| ***DoS attack*** | DoS TCP | 12,315,997 | 985,280 | 246,320 |
| ***DoS attack*** | DoS UDP | 20,659,491 | 1,652,759 | 413,190 |
| ***DoS attack*** | DoS HTTP | 29,706 | 2,376 | 594 |
| ***Information theft*** | Keylogging | 1,469 | 1,175 | 294 |
| ***Information theft*** | Data theft | 118 | 94 | 24 |
| ***Total*** | / | 73,370,443 | 5,877,647 | 1,469,413 |

### 4.2 CIC-IDS-2017

The dataset used, which the current paper focuses on, includes CICIDS2017 with benign and attack traffic. It contains analysis results of network traffic by CICFlowMeter and labelled flow based on timestamp, IP with port for source, IP with Port for destination, protocols, and attack. The capturing started from Monday, July_3_2017 to Friday July_7_2017 for five days in total. The attack types in this dataset are Heartbleed, DoS, infiltration, brute force SSH, brute force FTP, DDoS, web attack, and Botnet as seen in Table 2. Other datasets of IDS have separated the training from the testing dataset, but CICIDS2017 have collected all records of each attack into one CSV file. This dataset has 80 network flow features [4].

TABLE II. Attack Types in CICIDS2017 Dataset [9].

| *Category* | *Total* | *Total (-Rows with Lack Info)* | *Training* | *Test* |
|---|---|---|---|---|
| ***BENIGN*** | 2,273,097 | 2,271,320 | 20,000 | 20,000 |
| ***DDoS*** | 128,027 | 128,025 | 2,700 | 3,300 |
| ***DoS slowloris*** | 5,796 | 5,796 | 1,350 | 1,650 |
| ***DoS Slowhttptest*** | 5,499 | 5,499 | 2,171 | 1,169 |
| ***DoS Hulk*** | 231,073 | 230,124 | 4,500 | 5,500 |
| ***DoS GoldenEye*** | 10,293 | 10,293 | 1,300 | 700 |
| ***Heartbleed*** | 11 | 11 | 5 | 5 |
| ***PortScan*** | 158,930 | 158,804 | 3,808 | 4,192 |
| ***Bot*** | 1,966 | 1,956 | 936 | 624 |
| ***FTP-Patator*** | 7,938 | 7,935 | 900 | 1,100 |
| ***SSH-Patator*** | 5,897 | 5,897 | 900 | 1,100 |
| ***Web Attack-Brute Force*** | 1,507 | 1,507 | 910 | 490 |
| ***Web Attack-XSS*** | 652 | 652 | 480 | 160 |
| ***Web Attack-SQL Injection*** | 21 | 21 | 16 | 4 |
| ***Infiltration*** | 36 | 36 | 24 | 6 |
| ***Total Attack*** | 471,454 | 470,365 | 20,000 | 20,000 |
| ***Total*** | 2,830,743 | 2,827,876 | 40,000 | 40,000 |

## 5. METHODOLOGY

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

The proposed system is developed using both ML and DL techniques to enhance the detection of cyber threats, especially in IoT environments. The methodology entails four main stages: data aggregation, pre-processing, model training, and testing. Two publicly available datasets, Bot-IoT and CICIDS2017, have been used for evaluating the system. These datasets include a wide variety of network traffic scenarios, including both normal and malicious activities. A stratified approach to pre-processing ensures the quality of input data for training reliable models.

### 5.1 Data aggregation

The BoT-IoT dataset was collected at the AACSR Cyber Range Lab in Australia, simulating an IoT environment with real devices (e.g., sensors and smart home devices) under normal and attack scenarios. It includes labeled normal and malicious traffic, featuring attacks like DDoS, data exfiltration, and scanning, executed using tools like Metasploit and LOIC. This dataset is a valuable resource for IoT security research and Intrusion Detection System IDS development.

CIC-IDS-2017, developed by the Canadian Institute for Cybersecurity, is a complete, real-world network with normal and malicious traffic generated by human participants. It includes such simulated attacks as DDoS, brute force, infiltration, and web attacks. Captured with tools like Wireshark, it provides over 80 extracted features, including packet details, flow statistics, and timestamps. The labeled dataset would be helpful in support of the training and evaluations necessary in IDSs.

### 5.2 Data preprocessing

Data preprocessing is a crucial step in building reliable machine learning models, especially for intrusion detection on real-world datasets like BoT-IoT and CIC-IDS2017. The original data may contain noise, missing values, and inconsistencies that would affect the performance of a model in a negative way. Data preprocessing involves cleaning the data, handling missing values, feature normalization, and encoding categorical variables to prepare the dataset for learning. These methods help to improve model accuracy, reduce bias, and enhance generalizability, ultimately leading to more robust and accurate predictions.

The Bot-IoT dataset was reduced to 5% of its original size. Irrelevant columns like pkSeqID, stime, and daddr were removed, and categorical features such as proto and state were label-encoded. Numerical features were standardized, and highly correlated attributes were dropped to minimize redundancy. Missing values were imputed with column means, and memory usage was optimized by downcasting data types. The final dataset included 39 key features, with both normal and attack traffic labeled, and was split into training and testing sets. This preprocessing reduced the dataset's storage size from 985.21 MB to 549.28 MB, making it efficient for IoT intrusion detection tasks.

The CIC-IDS-2017 dataset preprocessing streamlined the data for intrusion detection tasks. Starting at 843.88 MB, it was reduced to 745.09 MB by removing irrelevant columns like `Destination Port` and `Flow Duration`. Missing values were handled through imputation, and the `Label` column was encoded into numeric form. Numerical features were standardized for consistency, and highly correlated features were removed to enhance efficiency. After memory optimization, 63 key features remained, including traffic metrics and packet statistics. The dataset was split into training and testing sets, providing a clean, efficient foundation for machine learning models.

A data pre-processing step inclusion allows the training procedure to be more reliable and the model to be more precise [7].

### 5.3 Machine Learning Models

The steps involved in the machine learning process are clearly outlined and illustrated in Figure 1.

- **K-Nearest Neighbors KNN:**

The KNN algorithm was tested with $N \in \{3,5,7\}$ to explore the impact of the neighbor count on performance. For each value of N, the performance metrics computed are Precision, recall, accuracy, and F1-score.

Algorithm 1: KNN Classification.

**Input**: CSV files containing the dataset, Hyperparameter (N: 3, 5, 7).

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

**Output**: Classification metrics (Accuracy, Precision, Recall, F1 Score), Training and Prediction times.

**Steps:**
**Step 1**: Begin.
**Step 2**: Load multiple CSV files and merge them into a single dataset.
**Step 3**: Read data.
**Step 4**: Drop irrelevant/object-type columns, encode categorical data, normalize numerical columns, and impute missing values. (feature reduction)
**Step 5**: Split data into features and target, perform stratified train-test split (10 tests, 10 validations, 80 trainings).
**Step 6**: Initialize the KNN model, train the model on the training set, predict on the test set and record prediction time,
**Step 7**: Calculate metrics (Accuracy using equation1, Precision, Recall, F1 Score).
**Step 8**: End.

- **Support Vector Machines SVM:**

Various kernel functions (linear, RBF, polynomial, and sigmoid) were tested to identify optimal decision boundaries. The experiments assessed the performance of each kernel on both datasets, considering accuracy, precision, recall, and F1-scores.

Algorithm 2: SVM Classification.

**Input:** CSV files containing the dataset, Kernels (N: Linear, RBF, Poly, Sigmoid).
**Output:** Classification metrics (Accuracy, Precision, Recall, F1 Score), Training and Prediction times.

**Steps:**
**Step 1**: Begin.
**Step 2**: Load multiple CSV files and merge them into a single dataset.
**Step 3**: Read data.
**Step 4**: Drop irrelevant/object-type columns, encode categorical data, normalize numerical columns, and impute missing values.
**Step 5**: Split data into features and target, perform stratified train-test split (10 test, 10 validation, 80 training).
**Step 6**: Initialize the SVM model with the kernel, train the model on the training set, predict the test set and record prediction time.
**Step 7**: Calculate metrics (Accuracy using equation1, Precision, Recall, F1 Score).
**Step 8**: End.

- **Random Forest RF:**

The RF algorithm was evaluated with N-estimators $\in\{70,100,130\}$ to analyze its performance scalability. Metrics such as accuracy and F1-scores were computed, alongside an analysis of training and prediction times to gauge computational efficiency.

Algorithm 3: Random Forest Classification.

**Input:** CSV files containing the dataset, Hyperparameter: (N: 70, 100, 130).
**Output:** Classification metrics (Accuracy, Precision, Recall, F1 Score), Training and Prediction times.

**Steps:**
**Step 1**: Begin.
**Step 2**: Load multiple CSV files and merge them into a single dataset.
**Step 3**: Read data.
**Step 4**: Drop irrelevant/object-type columns, encode categorical data, normalize numerical columns, and impute missing values.
**Step 5**: Split data into features and target, perform stratified train-test split (10 test, 10 validation, 80 training).
**Step 6**: Initialize the Random Forest model, train the model on the training set, predict on the test set and record prediction time.
**Step 7**: Calculate metrics (Accuracy using equation1, Precision, Recall, F1 Score).
**Step 8**: End.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

- **XGBoost:**

Gradient boosting with XGBoost was tested with N-estimators ∈{70,100,130}, Metrics such as precision, accuracy, recall, and F1-score were calculated for each configuration, and training/prediction times were analyzed to assess its suitability for real-time intrusion detection.

Algorithm 4: XGBoost Classification.

**Input:** CSV files containing the dataset, Hyperparameter (N: 70, 100, 130)

**Output:** Classification metrics (Accuracy, Precision, Recall, F1 Score), Training and Prediction times.

**Steps:**
**Step 1**: Begin.
**Step 2**: Load multiple CSV files and merge them into a single dataset.
**Step 3**: Read data.
**Step 4**: Drop irrelevant/object-type columns, encode categorical data, normalize numerical columns, and impute missing values.
**Step 5**: Split data into features and target, perform stratified train-test split (10 test, 10 validation, 80 training).
**Step 6**:Initialize the XGBoost model, train the model on the training set, predict on the test set and record prediction time.
**Step 7**: Calculate metrics (Accuracy using equation1, Precision, Recall, F1 Score).
**Step 8**: End.

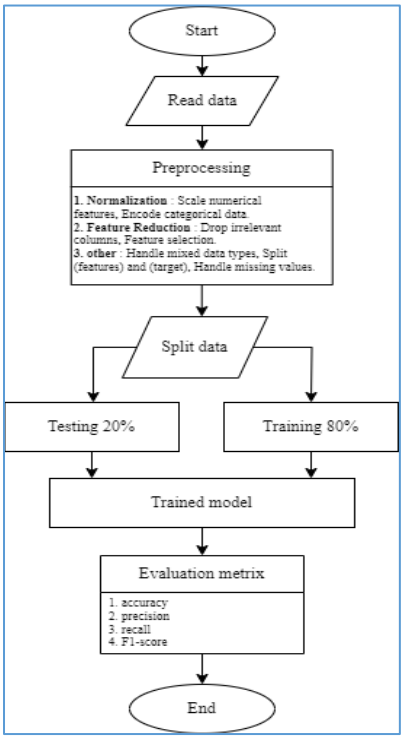


*Fig. 1. Machine learning steps.*

### 5.3 Deep Learning Models

**Convolutional Neural Networks CNNs:**

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

CNNs were implemented to classify the Bot-IoT dataset effectively, designed to process 1D time-series and tabular data. The architectures consisted of layers such as Conv1D, MaxPooling1D, Flatten, Dropout, and Dense. The models were compiled using the Adam optimizer with a learning rate of 0.01 and trained with early stopping to prevent overfitting. Two configurations were evaluated:

**With Preprocessing P:**

Preprocessing steps included MinMax scaling for feature normalization, dropping object-type and high-imbalance columns, and reshaping data for compatibility with CNN input.

Training data was augmented with noise to enhance model robustness. Features were reduced to accelerate convergence and reduce overfitting.

**Without Preprocessing NP:**

The model was trained directly on the raw dataset without any feature scaling or dimensionality reduction.

The steps involved in the deep learning process are clearly outlined and illustrated in Figure 2.

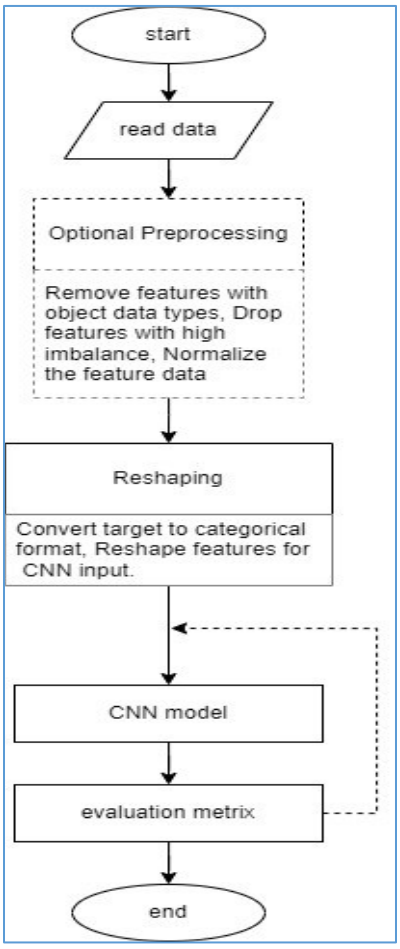


*Fig. 2. Deep learning steps.*

Algorithm 5: CNN Classification with preprocessing.

**Input:** CSV files containing the dataset, Hyperparameters: epochs = 50, learning rate = 0.01.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

**Output:** Classification metrics (Precision, Accuracy, F1 Score, Recall), Training and Prediction times.

**Steps:**
**Step 1**: Begin.
**Step 2**: Load multiple CSV files and merge them into a single dataset.
**Step 3**: Read data.
**Step 4**: Drop irrelevant/object-type columns, encode categorical data, normalize numerical columns, and impute missing values.
**Step 5**: Split data into features and target, perform stratified train-test split. (10 test, 10 validation, 80 training)
**Step 6**: Drop high-imbalance features and normalize remaining features using MinMaxScaler.
**Step 7**: Reshape the feature data for CNN input and convert the target variable into categorical format.
**Step 8**: Use the categorical crossentropy loss function for multi-class classification (objective function).
**Step 9**: Initialize the CNN model with Conv1D, MaxPooling1D, Dropout, and Dense layers.
**Step 10**: Compile the model using the Adam optimizer (learning rate = 0.01) and categorical crossentropy loss.
**Step 11**: Train the model for 50 epochs with a batch size of 64.
**Step 12**: Evaluate the model on the test set and record prediction time.
**Step 13**: Calculate metrics (Precision, Accuracy, F1 Score, Recall).
**Step 14**: End.

Machine Learning ML and Deep Learning DL models vary in their strengths regarding intrusion detection. ML models such as K-Nearest Neighbors KNN, Support Vector Machine SVM, Random Forest RF, and XGBoost are usually quicker to train, are less computationally intensive, and are suitable for working with structured small datasets. They usually take advantage of manual feature selection, providing additional control and interpretability, and this is beneficial in applications where explain ability is important.

On the other hand, DL models, such as Convolutional Neural Networks CNNs, are better at handling large and high-dimensional data. They can automatically learn complicated patterns and hierarchical features without any human intervention, which improves performance in identifying subtle or non-linear intrusion patterns. DL models require larger datasets, more computational resources, and longer training time, though.

While ML models are suitable for faster deployment and simpler scenarios, DL models are more effective in detecting more complex and dynamic attack patterns. Both approaches are contrasted in this research to highlight their strengths and determine the optimal solution for intrusion detection in the BoT-IoT and CIC-IDS2017 datasets.

## 6. RESULTS

### 6.1 Machine Learning Results

Summary of the results is shown in Table 3.

*TABLE III. Machine Learning Models Results.*

| *Dataset* | *Model* | *Parameter(s)* | *Accuracy (%)* | *Precision (%)* | *Recall (%)* | *F1-Score (%)* | *Training Time (sec)* | *Testing Time (sec)* |
|---|---|---|---|---|---|---|---|---|
| ***BoT-IoT*** | KNN | k ∈ {3, 5, 7} | 99.99 | 99.99 | 99.99 | 99.99 | 0.1580–0.1627 | 1886.1914–1924.9005 |
| ***BoT-IoT*** | SVM | Linear, RBF, Poly, Sigmoid | 99.98–99.99 | 99.99 | 99.99 | 99.99 | 110.5493–3517.3344 | 1.8854–78.3139 |
| ***BoT-IoT*** | RF | n_estimators ∈ {70, 100, 130} | 99.99 | 99.99 | 99.99 | 99.99 | 620.5632–1170.4096 | 1.3637–2.5167 |
| ***BoT-IoT*** | XGBoost | n_estimators ∈ {40, 70, 100} | 99.99 | 99.99 | 99.99 | 99.99 | 8.4849–12.9930 | 0.1076–0.1414 |
| ***CICIDS2017*** | KNN | k ∈ {3, 5, 7} | 99.77–99.82 | 99.77–99.82 | 99.77–99.82 | 99.77–99.82 | 0.5862–0.5983 | 1488.7760–1522.4705 |
| ***CICIDS2017*** | SVM | Linear, RBF, Poly, Sigmoid | 10.22–61.50 | 35.96–79.29 | 36.22–66.80 | 35.96–66.80 | 728.5806–1345.3776 | 92.4794–571.4643 |
| ***CICIDS2017*** | RF | n_estimators ∈ {70, 100, 130} | 99.89 | 99.89 | 99.89 | 99.89 | 797.1805–1461.9145 | 4.1193–7.5860 |
| ***CICIDS2017*** | XGBoost | n_estimators ∈ {40, 70, 100} | 99.91 | 99.91 | 99.91 | 99.91 | 41.1007–98.0428 | 0.6004–1.3275 |

Computational performance of machine learning algorithms was evaluated based on training and testing time on BoT-IoT and CIC-IDS2017. Although all algorithms recorded high accuracy (≥99.99%) on BoT-IoT, XGBoost recorded the best in terms of computational performance with the shortest training time (8.48–12.99 sec) and testing time (0.10–

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

0.14 sec). However, KNN, although accurate, recorded extremely long testing times (up to ~1925 sec), which made it less suitable for real-time detection.

SVM showed tremendous variation in both datasets depending on the kernel used. While it performed well in BoT-IoT, it performed poorly with low accuracy and long training/testing time (up to ~571 sec) in CIC-IDS2017, especially with the Sigmoid kernel. Random Forest showed high accuracy in both datasets but longer training time (up to ~1461 sec), though its testing time was moderate.

On the CIC-IDS2017 dataset, XGBoost again took the lead with the best accuracy (99.91%) with relatively low training time (41–98 sec) and very low testing time (0.6–1.3 sec). KNN, although accurate, again had very high testing time (up to ~1522 sec), which may limit its application in time-sensitive scenarios.

In summary, XGBoost always provides the best trade-off between classification performance and computational efficiency and is therefore the best model to apply to practical intrusion detection systems.

### 6.2 Deep Learning Results

Summary of the results is shown in Table 4.

TABLE IV. Deep Learning Models Results.

| *Dataset* | *Preprocessing* | *Model* | *Accuracy (%)* | *Precision (%)* | *Recall (%)* | *F1-Score (%)* | *Training Time (sec)* | *Testing Time (sec)* | *Epochs* |
|---|---|---|---|---|---|---|---|---|---|
| ***BoT-IoT*** | Yes | CNN | 99.99 | 99.99 | 99.99 | 99.99 | 15078.3296 | 59.2579 | 50 |
| ***BoT-IoT*** | No | CNN | 99.99 | 99.99 | 99.99 | 99.99 | 15038.9921 | 75.6157 | 50 |
| ***CIC-IDS2017*** | Yes | CNN | 99.61 | 99.60 | 99.61 | 99.59 | 12602.1800 | 25.6680 | 50 |
| ***CIC-IDS2017*** | No | CNN | 66.04 | 43.61 | 66.04 | 52.53 | 9822.1842 | 23.3477 | 50 |

The CNN model's performance and computational cost were compared both on BoT-IoT and CIC-IDS2017 datasets with and without preprocessing. On the BoT-IoT dataset, CNN recorded 99.99% accuracy regardless of whether preprocessing was applied, demonstrating its ability to learn effectively from cleaner, better-organized data. While preprocessing did reduce testing time considerably (59.26 sec vs. 75.62 sec), it only slightly improved training consistency, both training times fluctuating around 15,000 seconds for 50 epochs.

Conversely, for more difficult and variable CIC-IDS2017 dataset, preprocessing had an effect. With preprocessing, CNN gained 99.61% accuracy and 99.59% F1-score, while without preprocessing, accuracy decreased to 66.04% and F1-score reduced to 52.53%. Even though the training time increased with preprocessing (12,602 sec vs 9,822 sec), it worths additional expenses.

Overall, CNNs are computationally intensive, especially during training, but can yield better outcomes when effectively paired with effective preprocessing. While preprocessing does add to the computation load, it is required when dealing with complex datasets like CIC-IDS2017 to produce high accuracy, generalization, and correct intrusion detection.

### 6.3 Trade-Offs between Accuracy and Computational Efficiency in Real-Time IDS

Generally, there is a trade-off between high accuracy and computational expense in real-time IDS deployment. While high accuracy is critical to identify intrusions and minimize false positives/negatives, more accurate models—especially deep models like CNNs—will generally need longer training and testing time, more memory, and greater processing power. For instance, in this work, CNNs were 99.99% accurate on the BoT-IoT dataset but had a large training time (~15,000 sec) and relatively moderate test time. While suitable for offline analysis, such models may not perform well in time-critical environments with resource scarcity.

Conversely, XGBoost produced a nearly identical accuracy (99.99%) with significantly lower training (8–13 sec) and testing times (0.1–0.14 sec), and thus is more appropriate for real-time deployment. However, as dataset complexity increases (e.g., CIC-IDS2017), this trade-off becomes increasingly challenging.

Thus, selecting an IDS model for real-time use must be carefully balanced in terms of: Detection performance (not to sacrifice security), Resource availability (CPU/GPU, memory), and Latency constraints (to respond to threats in a timely fashion).

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

In short, high precision is required to carry out reliable intrusion detection, but computationally intensive models like XGBoost provide a reasonable compromise, particularly where resource-limited IoT settings require real-time response.

### 6.4 Analysis of False Positive and False Negative Rates

To quantify the rate of reliability in IDS, it is important to look at the False Positive Rate FPR and False Negative Rate FNR. FPR is an innocent traffic incorrectly said to be malicious, causing false alarms. FNR is a genuine attack that go undetected, which is very dangerous.

In this study, models like XGBoost and CNN with preprocessing achieved high precision and recall (≥99.60%) and consequently low FPR and FNR. For example, CNN with preprocessing based on the CIC-IDS2017 dataset recorded FPR and FNR below 1%. Yet, CNN without preprocessing achieved low precision (43.61%) and recall (66.04%), with a consequence of much higher FPR (~56%) and FNR (~34%). This reinforces the need for robust models as well as decent preprocessing to lower errors and produce sound intrusion detection.

## 7. CONCLUSION

The results highlight the strong performance of both ML and DL models in IoT intrusion detection. Machine learning models like KNN, SVM, Random Forest, and XGBoost achieved near-perfect accuracy and efficiency, especially on the BoT-IoT dataset. Deep learning models, particularly CNNs, demonstrated adaptability to complex datasets like CIC-IDS2017, achieving 99.61% accuracy with preprocessing. Preprocessing significantly enhanced model performance, reducing overfitting and improving convergence. These findings confirm the importance of advanced models and effective preprocessing in addressing IoT security challenges.

## References

[1] Rashid, A., Chivers, H., Danezis, G., Lupu, E., & Martin, A. (Eds.). (2019). *The Cyber Security Body of Knowledge* (Version 1.0). National Cyber Security Centre. Retrieved from https://www.cybok.org/

[2] Kerrakchou, I., Abou El Hassan, A., Chadli, S., Emharraf, M., & Saber, M. (2023). Selection of efficient machine learning algorithm on Bot-IoT dataset for intrusion detection in internet of things networks. *Indonesian Journal of Electrical Engineering and Computer Science*, *31*(3), 1784–1793.

[3] Alghazzawi, D., Wazzan, M., Algazzawi, D., Bamasaq, O., Albeshri, A., Cheng, L., & Bamasag, O. (2021). Internet of Things Botnet Detection Approaches: Analysis and Recommendations for Future Research: Analysis and recommendations for future research. *Applied Sciences*, *11*(12), 5713.

[4] Jabbar, A. F., & Mohammed, I. J. (2021). BotDetectorFW: An optimized botnet detection framework based on five features-distance measures supported by comparisons of four machine learning classifiers using CICIDS2017 dataset. *Indonesian Journal of Electrical Engineering and Computer Science*, *21*(1), 377–390.

[5] Alosaimi, S., & Almutairi, S. M. (2023). An intrusion detection system using BoT-IoT. *Applied Sciences*, *13*(9), 5427.

[6] Alsoufi, M. A., Siraj, M. M., Ghaleb, F. A., Al-Razgan, M., Al-Asaly, M. S., Alfakih, T., & Saeed, F. (2024). Anomaly-Based Intrusion Detection Model Using Deep Learning for IoT Networks. *Computer Modeling in Engineering & Sciences*, *141*(1), 823–845.

[7] Adekunle, T. S., Alabi, O. O., Lawrence, M. O., Adeleke, T. A., Afolabi, O. S., Ebong, G. N., Egbedokun, G. O., & Bamisaye, T. A. (2024). An intrusion system for internet of things security breaches using machine learning techniques. *Artificial Intelligence and Applications*, *2*(3), 165–171.

[8] Susilo, B., & Sari, R. F. (2020). Intrusion detection in IoT networks using deep learning algorithm. *Information*, *11*(5), 279.

[9] Ferrag, M. A., Maglaras, L., Ahmim, A., Derdour, M., & Janicke, H. (2020). Rdtids: Rules and decision tree-based intrusion detection system for internet-of-things networks. *Future Internet*, *12*(3), 44.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 51, No. 1, 2025, pp. 83-93
DOI: https://doi.org/10.25195/ijci.v51i1.552
Print ISSN: 2313-190X, Online ISSN: 2520-4912

[10] Shafiq, M., Tian, Z., Bashir, A. K., Du, X., & Guizani, M. (2020). CorrAUC: A malicious bot-IoT traffic detection method in IoT network using machine-learning techniques. *IEEE Internet of Things Journal*, *8*(5), 3242–3254.

[11] Korium, M. S., Saber, M., Beattie, A., Narayanan, A., Sahoo, S., & Nardelli, P. H. J. (2024). Intrusion detection system for cyberattacks in the Internet of Vehicles environment. *Ad Hoc Networks*, *153*, 103330.

[12] Jose, J., & Jose, D. V. (2023). Deep learning algorithms for intrusion detection systems in internet of things using CIC-IDS 2017 dataset. *International Journal of Electrical and Computer Engineering (IJECE)*, *13*(1), 1134–1141.

[13] Dat-Thinh, N., Xuan-Ninh, H., & Kim-Hung, L. (2022). MidSiot: A multistage intrusion detection system for internet of things. *Wireless Communications and Mobile Computing*, *2022*(1), 9173291.

[14] Foody, G. M. (2023). Challenges in the real world use of classification accuracy metrics: From recall and precision to the Matthews correlation coefficient. *Plos one*, *18*(10), e0291908.