

IoT Network Security based on Intrusion Detection System using Stacked Ensemble

HASSAN KHAZANE¹, MOHAMMED RIDOUANI¹, FATIMA SALAHDINE²,
NAIMA KAABOUCH³

¹RITM Laboratory, CED Engineering Sciences, ENSEM
Hassan 2 University
Casablanca
MOROCCO

²Dept. of Electrical and Computer Engineering
University of North Carolina
Charlotte, North Carolina
USA

³Artificial Intelligence Research Center
University of North Dakota
Grand Forks, North Dakota
USA

Abstract: - The rapid evolution of IoT networks has led to an increasing number of devices connecting to the internet, exposing them to various cyber threats. Detecting intrusions in IoT environments is essential but challenging. Network Intrusion Detection Systems are vital in analyzing network traffic to differentiate normal and malicious activities without compromising security. However, the abundance of benign traffic complicates accurate detection. To overcome this challenge, we propose an Ensemble-based Network Intrusion Detection Systems framework, where five Machine Learning classifiers are combined through a Stacking approach and with nature-inspired feature selection techniques to enhance the detection effectiveness. The performance of the proposed model was compared to four base models - Random Forest, Extra Trees, AdaBoost, and Gradient Boosting - in terms of several metrics. The experimental results on the CIIoT2023 dataset reveal that the proposed stacking model consistently outperforms the base classifiers across all evaluation metrics.

Key-Words: - Stacked Ensemble, Intrusion Detection, Internet of Things, Cybersecurity, Machine Learning, Deep Learning.

Received: July 21, 2024. Revised: April 9, 2025. Accepted: May 11, 2025. Published: June 25, 2025.

1 Introduction

Heterogeneous IoT networks pose serious difficulty in learning and characterizing complex traffic patterns due to the existence of different protocols and connected systems, [1]. The more considerable network traffic has many benign flows and a small number of malicious flows, making the process valuable in making more sophisticated attack detection problematic. Single machine learning (ML) classifiers often do not have sufficient strength for effective detection and reliable classification of intrusions in such environments, [2]. In response, scientists have utilized ensemble classifiers [3], which combine several separate models to improve the accuracy and detection

performance. Ensemble learning benefits from the weaknesses of the individual classifiers, especially when there is little training data, and stacking ensemble methods have been found helpful in enhancing Network Intrusion Detection Systems (NIDS).

In 1992, Wolpert proposed Stacked Generalization, [4]. It has served as a fundamental basis for many modern solutions. With the growing interest in integrating ML and deep DL methods, various studies presented NIDS, significantly enhancing the detection performance based on stacking ensemble methods. For instance, authors in [5] developed a stacking model combining three base models, Linear Regression (LR), Random

Forest (RF), and K-Nearest Neighbors (KNN), and a Support Vector Machine (SVM) as the meta-model with an entropy-based feature selection strategy. The performance on UNSW-NB15 and UGR'16 datasets demonstrated 94% accuracy and a 5.2% False Alarm Rate (FAR), although the model has not been evaluated in real-world IoT environments. Likewise, the authors in [6] proposed B-Stacking, a method that combines boosting and stacking with KNN, RF, and XGBoost, which is used as base models to remove the bias and variance and improve level-1 model data quality. Another work [7], used a stacking ensemble with RNN to identify whether an instance belongs to an anomaly in the BoT-IoT dataset with satisfactory results for binary and multi-class (5 and 11 classes) properties. The authors in [8] developed an IoT NIDS that reduces false positives. The system records 85.81% for binary classification and 83.83% for multi-class classification, using the NSL-KDD dataset and KNN decision trees through Bagging and Boosting ensemble methods. In [9], the study has proposed a NIDS with base learners as XGBoost, KNN, Gaussian Naive Bayes, and RF as meta-classifiers, achieving a detection rate of 99.99% on the DS2OS dataset. In [10], the authors proposed a stacking ensemble framework called DIS-IoT with LSTM, CNN, DNN, and MLP models and achieved high accuracy on different open-source datasets: 99.6% (ToN_IoT), 98.7% (CICIDS2017), and 99.7% (SWaT).

This work introduces an NIDS designed to effectively and accurately detect various categories of DoS and DDoS attacks in IoT networks. We implemented nature-inspired feature selection methods to reduce dimensionality and remove redundancy by selecting the relevant features of the CICIoT2023 [11] Dataset. The framework is based on the stacking ensemble approach, where four base learners are integrated: Random Forest (RF), Extra Trees (ET), AdaBoost (Ada), and Gradient Boosting (GB). Logistic Regression (LR) is used as a meta-learner to combine their outputs effectively. This advanced architecture processes the input data and delivers a robust final classification.

The structure of this paper is as follows: Section 2 elucidates the methodology employed in developing our NIDS proposal. Section 3 presents a detailed analysis of the experimental results. Section 4 concludes this research endeavor.

2 Methodology

This section outlines a systematic methodology for building and evaluating a stacking model for

intrusion detection in an IoT environment Fig. 1. First, the preprocessing steps are conducted to optimize the dimensionality of the CICIoT2023 [10] Dataset. We refine the dataset by focusing solely on DDoS attacks. Then, we select and identify features by calculating their importance and correlation from the given dataset to make the dimensionality optimum and relevant. Splitting the data into training, validation, and test sets is the final step of the preprocessing stage.

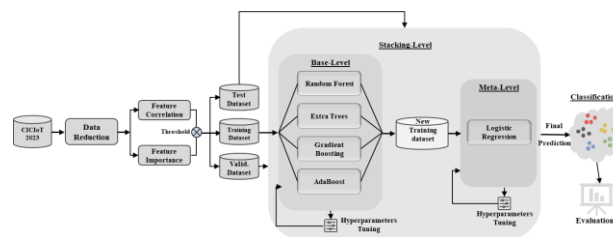


Fig. 1: Methodology Flow Chart

For each base classifier, the hyperparameter tuning is performed with GridSearch. To measure the performance of different models, we propose a new measure called the Tuning Score (TS), which, in our case, is equal to the ratio of accuracy to the time to complete the tuning operation. The concept of accuracy was introduced to achieve an outstanding balance between prediction exactitude and computation complexity, and it is essential in actual NIDS, where false alarm elimination and fast response are valuable.

The individual training outputs of these base learners are then combined to form a new dataset, which serves as the input for the meta-classifier. The meta-classifier is trained to adjust for the errors of the base classifiers, leveraging their outputs to optimize predictive accuracy. We also employ 5-fold cross-validation for the meta-classifier training phase to further enhance generalizability. This iterative process helped to minimize errors and improve the stacking model's overall performance.

Finally, the trained base and meta-learners are stored, and the stacking model is constructed and prepared for testing. We use the test dataset to assess the stacking model through multiple evaluation metrics, which will be explained in the following section. The evaluation approach teaches us about using thorough data preprocessing, model stacking, hyperparameter tuning, and cross-validation to develop an intrusion detection system with high-performance levels to manage complex real-world network security complexities.

2.1 Data Processing

The IDS requirements determine three sequential processes for preparing experimental data. The initial data processing stage transforms and reduces the information to meet modeling standards and enables efficient computation. The second preprocessing stage reveals the linkages between the system inputs and output targets. The final step uses feature importance methods to recognize essential data patterns and unimportant features that should be omitted. Fig. 2 summarizes the data processing flow.

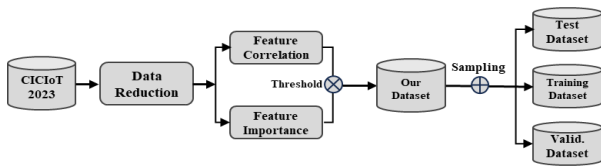


Fig. 2: Data Processing Flowchart

Data Reduction: To develop a balanced and representative dataset for our study, 169 CSV files from the CICIoT2023 dataset were merged into a single resource, focusing on DDoS attacks and benign traffic due to computational constraints. A stratified random sampling of 5,000 samples from 11 classes ensured balanced data. The dataset was shuffled to reduce biases and saved as a new CSV file for model development. Fig. 3 illustrates the number of rows corresponding to the 11 different classes.

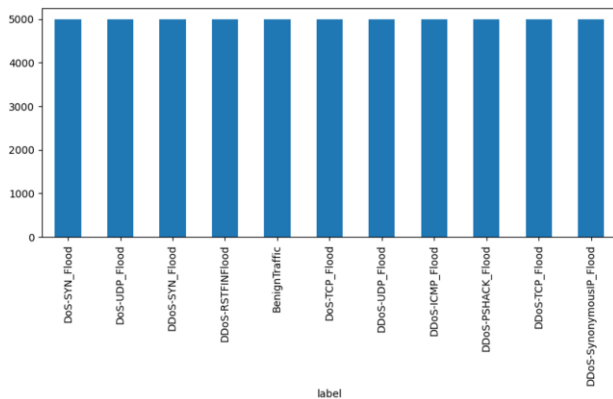


Fig. 3: Count of Samples for Each Class in the Reduced Dataset

Feature Correlation: In predictive modeling, not all variables contribute effectively, and irrelevant or redundant features can reduce classification accuracy, increase computational demands, and complicate model interpretability. Feature correlation is a widely used statistical method for feature selection, analyzing relationships between variables to uncover significant connections. Highly

correlated features, with values close to 1, often indicate redundancy, as they provide overlapping information, while correlations near 0 suggest weak or no relationships. This study visually identifies these relationships by employing a correlation matrix heatmap to remove uninformative features. The heatmap represents the strength and direction of correlations, helping streamline the dataset for efficient and accurate model development. Fig. 4 and

Table 1 detail the detected correlations, emphasizing their role in optimizing feature selection for improved performance.

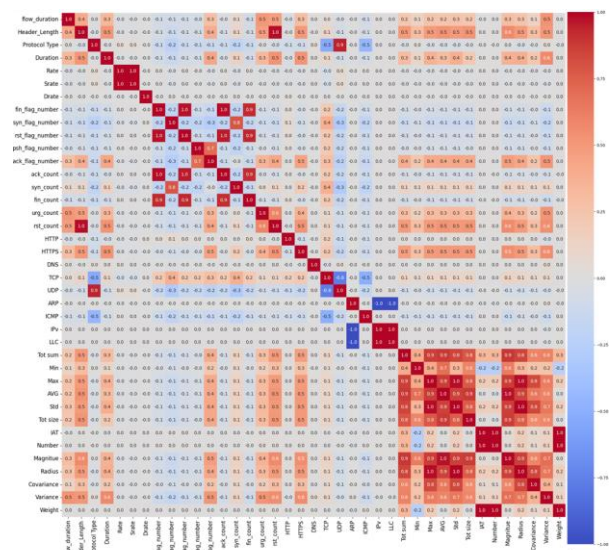


Fig. 4: Correlation Heatmap of Feature Pairwise Correlation Matrix

Table 1. Correlated Features

Column	Correlated Row
Srate	Rate
rst flag number	fin flag number
ack count	fin flag number
ack count	rst flag number
fin count	fin flag number
fin count	rst flag number
fin count	ack count
rst count	Header Length
UDP	Protocol Type
LLC	IPv
Std	Max
Number	IAT
Magnitude	AVG
Radius	Max
Radius	Std
Weight	IAT
Weight	Number

Feature Importance: It is essential for identifying input features that significantly impact

the prediction of the target variable. It assigns a score to each feature based on its contribution to the model's performance, with higher scores indicating a more significant influence.

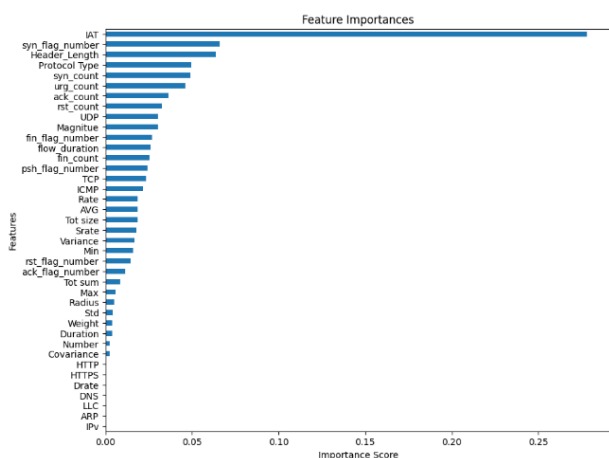


Fig. 5: Feature Importance with Random Forest Classifier (MDI)

Table 2. Correlated Features with Their Associated Feature Importance Scores

Feature	Correlated Feature	Features Importance	Correlated Feature Importance
Srate	Rate	0.018404	0.019127
rst_flag_number	fin_flag_number	0.016861	0.041224
ack_count	fin_flag_number	0.020802	0.041224
ack_count	rst_flag_number	0.020802	0.016861
fin_count	fin_flag_number	0.024687	0.041224
fin_count	rst_flag_number	0.024687	0.016861
fin_count	ack_count	0.024687	0.020802
rst_count	Header_Length	0.030999	0.055493
UDP	Protocol_Type	0.034883	0.045574
LLC	IPv	0.000003	0
Std	Max	0.006301	0.004401
Number	IAT	0.003304	0.271541
Magnitude	AVG	0.024915	0.026067
Radius	Max	0.006431	0.004401
Radius	Std	0.006431	0.006301
Weight	IAT	0.003822	0.271541
Weight	Number	0.003822	0.003304

Various methods exist to measure feature importance, including model coefficient-based, decision tree-based, and permutation-based

approaches. In this study, we utilized a decision tree-based method using Gini importance (Mean Decrease Impurity), which evaluates a feature's significance by calculating the reduction in node impurities across all trees in the ensemble. This method handles both categorical and numerical data and accounts for feature interactions. Then, we set a feature importance threshold of 0.01 to refine the dataset, retaining only the most impactful features. Importance scores were then analyzed alongside feature correlation results in

Table 1. The less significant feature was removed for highly correlated feature pairs to reduce redundancy and multicollinearity, Table 2. Non-correlated features below the threshold were also eliminated. This systematic process reduced the feature set from 46 to 17, resulting in a streamlined and efficient dataset for analysis. Fig. 5 illustrates the feature importance rankings.

2.2 Stacking Model

Stacking, also recognized as Stacked Generalization, is one of the modern ensemble learning solutions in ML. The main objective of this technique is to combine different models that handle the same problem through their individual strengths. The main idea of stacking depends on a fundamental understanding that various models examine distinct problem details without attempting to handle the complete complexity. Several independent models called base learners implement base-level training for the stacking process. The base learners generate intermediate outcomes that serve as input for a higher-level model termed the meta-model. The meta-model receives all individual predictions from base learners before it learns how to assemble them in an ideal way that produces a more precise and robust prediction. The combination of deep learning with ML components has become common for IoT network intrusion detection solutions during the past few years. The proposed solution develops an ensemble framework that specifically targets multi-class classification. The framework uses Random Forest, Extra Trees, AdaBoost, and Gradient Boosting as its four base learners. Separate training processes enable base learners to produce differing intermediate predictions. A Logistic Regression model works as the meta-learner to effectively unite the base learner outputs while generating the final classification results. The Fig. 6 shows the architectural flow of this framework.

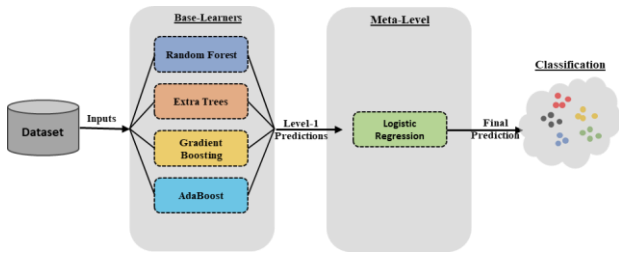


Fig. 6: Our Stacking Model Architecture

2.3 Performance Metrics

The model's performance evaluation employed the standard metrics from intrusion detection systems which incorporated Accuracy, Precision, Recall, F1-score, False Positive Rate (FPR), and False Negative Rate (FNR). These metrics comprehensively measure the model's ability to classify normal and attack instances effectively.

Accuracy measures the ratio of correctly classified samples to the total samples, given by:

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

Precision evaluates the proportion of true positive samples among all predicted positive samples and is computed as:

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

Recall, also known as the detection rate, reflects the model's capacity to identify attack samples accurately and is calculated using:

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

F1-score, the harmonic mean of Precision and Recall, balances their contributions and is expressed as:

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

False Positive Rate (FPR) quantifies the proportion of normal samples misclassified as attacks, calculated as:

$$FPR = \frac{FP}{FP + TN} \quad (5)$$

False Negative Rate (FNR), also called the missed alarm rate, measures the proportion of attack samples misclassified as normal instances:

$$FNR = \frac{FN}{FN + TP} \quad (6)$$

In these formulas, TP (True Positive) refers to correctly identified attack samples, TN (True Negative) to correctly identified normal samples, FP (False Positive) to normal samples misclassified as attacks, and FN (False Negative) to attack samples misclassified as normal.

A confusion matrix provides a detailed view of classification performance, showing how instances are distributed across four categories: TP, TN, FP, and FN. The confusion matrix contains 11 rows and 11 columns, matching the 11 dataset classes that include normal traffic and different DDoS attacks. The matrix identifies incorrect classifications through false alarms (FP) and false negative detections (FN), providing knowledge about model performance and recommending improvement in attack detection.

3 Results and Discussion

This research proposed an IoT NIDS framework using stacking which detects normal IoT activity and DDoS attack indicators within multi-class analytical environments. The main goal of this research was to determine how effective stacking proved to be at improving the network intrusion detection systems' performance. The proposed framework was implemented using the Scikit-learn library in Python, ensuring reproducibility and cutting-edge machine learning tools. Comprehensive performance evaluation was conducted using key metrics, including accuracy, precision, recall, F1-score, FPR, and FNR. The detailed results are presented in the subsequent tables and figures.

Table 3 outlines a benchmark of performance metrics for both the base and stacking models, evaluated on the test dataset and through cross-validation. The stacking model demonstrated superior performance on the test dataset, achieving the highest accuracy, precision, recall, and F1-Score scores, all at 99.891%. In the cross-validation phase, the stacking model continued to outperform the individual models, with accuracy, precision, recall, and F1-Score averaging around 99.846%. The minor decrease of 0.045% is negligible and reflects the robustness and generalization capabilities of the stacking model.

Table 3. Performance Benchmark Between Stacking Model and Base Models

Model	Test				Cross-validation			
	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
Random Forest	99.800	99.801	99.800	99.800	99.733	99.740	99.736	99.737
Extra Trees	84.091	88.161	84.091	83.690	87.878	91.035	87.818	87.598
Gradient Boosting	99.873	99.873	99.873	99.873	99.872	99.784	99.782	99.782
AdaBoost	85.255	83.115	82.067	82.067	89.673	90.239	89.673	863907
Stacking Model	99.891	99.891	99.891	99.891	99.846	99.847	99.846	99.846

The stacking model demonstrates high effectiveness in distinguishing benign traffic from various DDoS and DoS attack types, achieving consistently high precision, recall, and F1-score, as illustrated in Table 4. Notably, it excels in identifying DDoS attacks, with many categories attaining perfect precision and recall, thereby minimizing false positives. The model's classification of benign traffic also shows impressive precision (99.51%) and ideal recall (100%), which helps reduce alert fatigue. However, some DoS attack categories show slightly lower recall than DDoS types, indicating areas for potential improvement.

Table 4. Evaluation Metrics for Different Types of Attacks

	Accuracy(%)	Precision(%)	Recall(%)	F1-score(%)
BenignTraffic	99.51	100.0	99.75	1007
DDoS-ICMP_Flood	99.71	99.90	99.81	1033
DDoS-PSHACK_Flood	100.0	100.0	100.0	964
DDoS-RSTFINFlood	100.0	99.80	99.90	1014
DDoS-SYN_Flood	100.0	99.80	99.90	997
DDoS-SynonymousIP_Flood	100.0	100.0	100.0	1005
DDoS-TCP_Flood	100.0	99.90	99.95	984
DDoS-UDP_Flood	99.81	99.90	99.86	1045
DoS-SYN_Flood	100.0	99.90	99.95	973
DoS-TCP_Flood	99.81	100.0	99.90	1025
DoS-UDP_Flood	99.89	99.48	99.68	953

This conclusion is supported by the confusion matrix shown in Fig. 7, which displays the distribution of true positives and misclassifications across different classes. It highlights the model's sensitivity and specificity, demonstrating strong performance in identifying DDoS-ICMP Flood and revealing vulnerabilities in detecting specific DoS attacks.

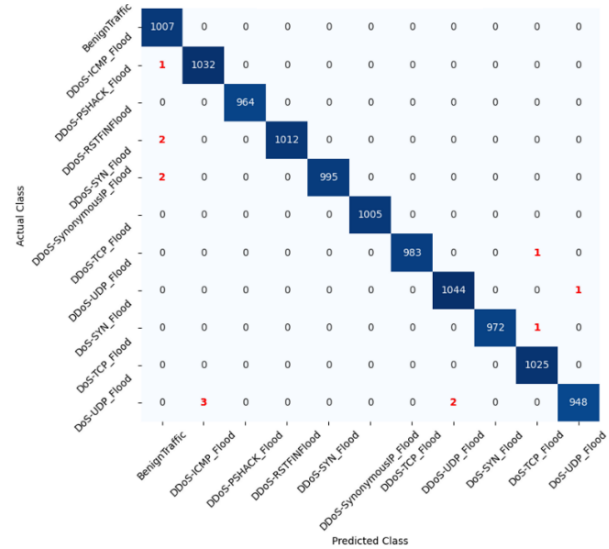


Fig. 7: Confusion Matrix for the Staking Model

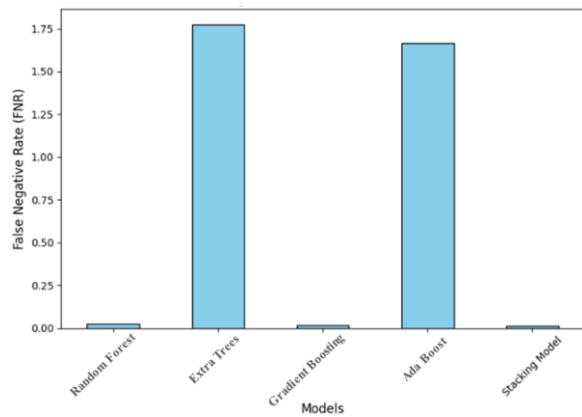


Fig. 8: False Negative Rate

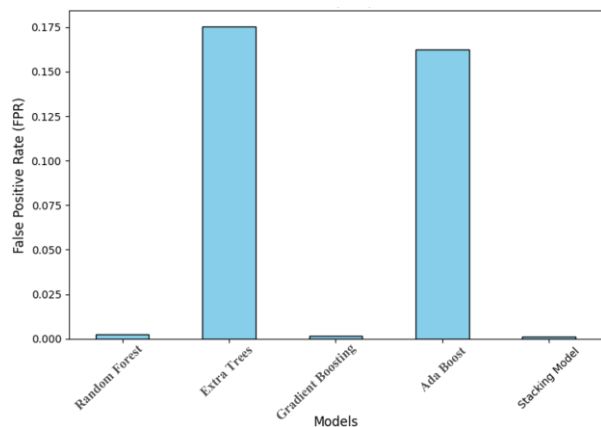


Fig. 9: False Positive Rate

The stacking model maintains an even performance level for FPR and FNR compared to the individual base models, including RF, GB, and ADA. The system achieves minimal error rates in attack detection shown in Fig. 8, because of its low FNR performance. Accurate threat detection and mitigation must be established due to its critical importance for process efficiency. The FPR

measurement for this model remains low according to Fig. 9 and shows excellent ability to prevent incorrect positive results. High FPR levels create operational inefficiency since they generate too many needless security alerts for security teams to handle.

4 Conclusion

The research team develops and evaluates a stacking model for network intrusion detection systems in IoT environments. Our framework operates for multi-class classification through four base learners implemented by Random Forest, Extra Trees, AdaBoost, and Gradient Boosting. At the same time, a Logistic Regression model acts as the meta-learner. Our results show that the stacking model is an effective NIDS solution. It consistently outperforms the base models with an accuracy of 99.891% on the test dataset and 99.846% on the validation dataset. It demonstrates balanced performance regarding FPR and FNR, indicating reliability in detecting threats. However, some DoS attack categories have slightly lower recall rates, highlighting areas for improvement. Overall, the stacking model is a valuable asset in cybersecurity, and addressing these improvement areas will enhance its threat detection and response capabilities.

References:

- [1] H. Khazane, M. Ridouani, F. Salahdine, and N. Kaabouch, "A Holistic Review of Machine Learning Adversarial Attacks in IoT Networks," *Future Internet*, vol. 16, no. 1, p. 32, Jan. 2024, doi: 10.3390/fi16010032.
- [2] N. R. Al-Milli and Y. A. Al-Khassawneh, "Intrusion Detection System using CNNs and GANs," *WSEAS Transactions on Computer Research*, vol. 12, pp. 281–290, Apr. 2024, <https://doi.org/10.37394/232018.2024.12.27>.
- [3] T. J. Lucas, I. S. de Figueiredo, C. A. C. Tojeiro, A. M. G. de Almeida, R. Scherer, J. R. F. Brega, J. P. Papa, K. . P. da Costa, "A Comprehensive Survey on Ensemble Learning-Based Intrusion Detection Approaches in Computer Networks," *IEEE Access*, vol. 11, pp. 122638–122676, 2023, doi: 10.1109/ACCESS.2023.3328535.
- [4] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, Jan. 1992, doi: 10.1016/S0893-6080(05)80023-1.
- [5] S. Rajagopal, P. P. Kundapur, and K. S. Hareesha, "A Stacking Ensemble for Network Intrusion Detection Using Heterogeneous Datasets," *Security and Communication Networks*, vol. 2020, pp. 1–9, Jan. 2020, doi: 10.1155/2020/4586875.
- [6] S. Roy, J. Li, B.-J. Choi, and Y. Bai, "A lightweight supervised intrusion detection mechanism for IoT networks," *Future Generation Computer Systems*, vol. 127, pp. 276–285, Feb. 2022, doi: 10.1016/j.future.2021.09.027.
- [7] S. I. Popoola, B. Adebisi, M. Hammoudeh, H. Gacanin, and G. Gui, "Stacked recurrent neural network for botnet detection in smart homes," *Computers & Electrical Engineering*, vol. 92, p. 107039, Jun. 2021, doi: 10.1016/j.compeleceng.2021.107039.
- [8] J. Sharma, C. Giri, O.-C. Granmo, and M. Goodwin, "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation," *EURASIP J. on Info. Security*, vol. 2019, no. 1, Dec. 2019, doi: 10.1186/s13635-019-0098-y.
- [9] P. Kumar, G. P. Gupta, and R. Tripathi, "A distributed ensemble design based intrusion detection system using fog computing to protect the internet of things networks," *J Ambient Intell Human Comput*, vol. 12, no. 10, pp. 9555–9572, Oct. 2021, doi: 10.1007/s12652-020-02696-3.
- [10] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowledge-Based Systems*, vol. 279, p. 110941, Nov. 2023, doi: 10.1016/j.knosys.2023.110941.
- [11] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. 23, no. 13, p. 5941, Jun. 2023, doi: 10.3390/s23135941.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed to the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US