

Optimized AI-Based Intrusion Detection System for Accurate DDoS Attack Detection

Rodrigue SAOUNGOUNI SOURPELE

rsaoungoumi@univ-garoua.cm

Université de Garoua

Patalet LAYIBE

University of Ngaoundéré

Franklin TCHAKOUNTE

University of Ngaoundéré

Blaise Omer YENKE

University of Ngaoundéré

Ado Adamou ABBA ARI

University of Maroua

Research Article

Keywords: Artificial Intelligence, Detection, Threats, Computer Networks, CICDDoS2019

Posted Date: September 2nd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10637753/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Optimized AI-Based Intrusion Detection System for Accurate DDoS Attack Detection

Rodrigue Saoungoumi Sourpele^{1,2} Layibe Patalet¹ Franklin Tchakounte¹
Blaise Omer Yenke¹ Ado Adamou Abba Ari^{3,4,5}

¹ LIN Lab, The University of Ngaoundere, PO BOX 455 Ngaoundere, Cameroon

² LIN Lab, University of Garoua, PO BOX 346 Garoua, Cameroon

³ LaRI Lab, University of Maroua, PO Box 814 Maroua, Cameroon

⁴ CREATIVE, Institute of Fine Arts and Innovation, University of Garoua, PO Box 346 Garoua, Cameroon

⁵ DAVID Lab, Université de Paris-Saclay, University of Versailles Saint-Quentin-en-Yvelines,
45 Avenue des Etats-Unis, 78000 Versailles, France

rsaoungoumi@univ-garoua.cm

Abstract

Modern network environments require a high level of security to ensure safe and reliable communication of information between different organizations. In these environments, cyberattacks are becoming increasingly sophisticated. Therefore, crucial that protection technologies adapt alongside threats. One of the most frequent and difficult attacks to detect with traditional detection systems is the Distributed Denial of Service (DDoS) attack, a variant of the DoS attack. While some work has been conducted to propose solutions based on the use of artificial intelligence algorithms to counter these attacks, it does not offer effective solutions for detecting modern DDoS threat types. It is with this in mind that we propose an AI-based intrusion detection system in our study. Five machine learning algorithms and one deep learning algorithm were tested on the CICDDoS2019 dataset. This dataset, consisting of two sets (training and test datasets), underwent preprocessing steps. To achieve good performance from our models, we applied the GridSearchCV optimization algorithm to determine the optimal hyperparameters. The test data was not balanced, but despite the unbalanced classes, we found that our models performed best on real-world data. Performance was evaluated using F1 scores, false positive and false negative rates, and AUC curves for each model on each type of attack. The Random Forest and XGBoost models performed excellently, with the best metrics and reasonable prediction times for UDP, MSSQL, LDAP, and NetBIOS attacks. Random Forest achieved F1 scores of 0.896, 0.944, 0.902, and 0.798, respectively. Its AUC (macro) was 0.9528, its FPR was 0.05, and its FNR was 0.2. XGBoost achieved F1 scores of 0.896, 0.934, 0.862, and 0.793, respectively, along with an AUC(macro) of 0.9445, an FPR of 0.04, and an FNR of 0.2. These results indicate that detection based on these models is more accurate and effective at identifying types of distributed denial-of-service (DDoS) attacks. It should be noted that these algorithms were compared with several other algorithms before being validated.

Keywords: *Artificial Intelligence, Detection, Threats, Computer Networks, CICDDoS2019*

1 Introduction

The use of technology has become truly indispensable in today’s fast-paced, interconnected world. From the moment we wake up to the time we go to sleep, digital tools and platforms influence nearly every aspect of our daily routines, reshaping how we work, communicate, and entertain ourselves. The internet, which forms the very cornerstone of this technological age, is now crucial for managing a wide array of functions that society depends upon. These include operating critical infrastructure such as power grids and transportation systems, handling complex financial operations like banking and stock trading, maintaining long-distance connections with loved ones through messaging and video calls, shopping online for goods and services, and accessing an almost limitless wealth of information and entertainment, from news articles to streaming movies.

However, the excessive and widespread use of the Internet has brought with it a host of serious security challenges. In essence, every object or person connected to the Internet—whether a smartphone, a home security camera, or a corporate server—becomes a potential target for cybercriminals operating across the globe. These malicious actors are constantly refining their methods. The techniques used by cybercriminals adapt as technology evolves, becoming more sophisticated and harder to counter, and they are always carefully designed to exploit specific vulnerabilities in software, hardware, or human behavior [12]. These cyberattacks can take various forms, ranging from phishing emails that trick users into revealing passwords to ransomware that locks entire computer systems. One of the most common types of cyberattack, and one that is particularly difficult for traditional detection systems to identify, is the distributed denial-of-service (DDoS) attack [5, 17]. In a DDoS attack, cybercriminals use vast networks of compromised devices, known as botnets, to flood targeted servers with malicious requests until those servers become overwhelmed and unavailable to legitimate users [22, 8, 9].

The consequences of such attacks can be severe. System downtime, whether lasting minutes or days, can lead to significant service interruptions, immediate loss of revenue, and long-term damage to an organization’s reputation [2, 16]. Customers who experience unreliable service may take their business elsewhere, and rebuilding trust can take years. According to Jim Holdsworth and Matthew Kosinski [11], the average cost of business disruption caused by a single cyberattack amounts to a staggering \$1.47 million, a figure that includes not only lost sales but also expenses related to incident response, legal fees, and regulatory fines.

To limit these substantial financial and operational losses, organizations have long relied on so-called traditional protection methods. These conventional security measures include intrusion detection and prevention systems (IDS/IPS), which monitor network traffic for suspicious activity and can either alert administrators or actively block threats. Another cornerstone of traditional defense is the firewall, which acts as a barrier between trusted internal networks and untrusted external networks, such as the internet, by filtering incoming and outgoing traffic based on predetermined security rules [3, 4]. Despite their widespread adoption and historical effectiveness against older, less sophisticated threats, it is now clear that cyber attackers have repeatedly breached this defence wall. The fundamental weakness of traditional systems lies in their reliance on signatures (known patterns of previous attacks) and anomalies (deviations from established baselines of normal behavior). Attackers have circumvented these limitations by developing new attack strategies that either mimic legitimate traffic to avoid triggering anomaly alerts or exploit unknown vulnerabilities for which no signature yet exists. Given this evolving threat landscape, developing new cyber defence strategies has therefore become a top priority for all cybersecurity personnel, from front-line analysts to chief information security officers.

Consequently, a new detection approach based on artificial intelligence (AI) has become central to modern cybersecurity frameworks. Unlike static, rule-based traditional methods, AI-powered systems can learn from data, adapt to new information, and identify patterns too subtle or complex for human analysts or signature-based tools to detect. From improving threat

detection by spotting zero-day attacks in real time to automating responses that isolate infected systems within milliseconds, AI is transforming how organizations defend their digital assets. Through its various algorithms—notably machine learning (which enables systems to learn from historical data without being explicitly programmed for every scenario) and deep learning (which uses multi-layered neural networks to model high-level abstractions in data). AI is revolutionising the field of cybersecurity. These advanced techniques are offering new ways to defend against sophisticated attacks that would otherwise overwhelm traditional defences [19].

The main focus of this study is to propose a robust and effective method for detecting distributed denial-of-service (DDoS) attacks based specifically on machine learning and deep learning algorithms. To ensure the validity and comparability of our results, the reference dataset used in this research consists of real-world DDoS attack patterns known as CICDDoS2019, which is sourced from the Canadian Institute for Cybersecurity. This well-regarded public dataset includes a wide variety of modern DDoS attack vectors, such as UDP floods, SYN floods, and HTTP-based attacks, alongside normal traffic profiles. Using this comprehensive dataset enabled us to evaluate the performance of various models under realistic conditions, comparing metrics such as accuracy, precision, recall, and false positive rates across different algorithmic approaches.

The first section of the research presents related work, the second section details the proposed method, section three (3) describes the experimental setup, and section four (4) presents the results and discussion, followed by the conclusion.

2 Related Works

Given the urgency of the subject and the significant losses caused by cyberattacks, improving detection systems was a challenge for many security professionals.

In this section, we cannot claim to have reviewed all the articles using machine learning and deep learning, but we have explored a few studies.

In his study, Zainab Sahib DhahirDhahir [7] proposes the CBLOF (Clustering-based Local Outlier Factor) to detect outliers in traffic and provide a detailed characterisation of abnormal behaviour, combined with the XGBoost (Extreme Gradient Boost) model to classify the detected anomalies as either attacks or normal traffic, and SHAP to facilitate the interpretation of features. The author used the CICIDS2017 dataset in this study. To compare the performance of the hybrid model, they utilised several other algorithms such as Random Forest, SVM, KNN, Naive Bayes and XGBoost. The results show that the CBLOF-plus-XGBoost hybrid model achieved an accuracy of 99.99 per cent, a recall of 100 per cent and a perfect F1 score. Hybrid methods are promising but are rarely combined with anomaly detection techniques such as CBLOF. This approach is computationally intensive and consumes a considerable amount of resources.

Amardeep Singha and *al.* [1] implement a unified auto-encoder combining a multi-scale convolutional neural network and long-term memory (MSCNN-LSTM-AE) for anomaly detection in network traffic. The multiscale convolutional neural network (MSCNN) allows for the extraction of inherited spatial patterns from network traffic. The model’s LSTM-AE (Labeled Sequence Temporal Patterns of Network Traffic) function extracts temporal patterns from the network traffic, in addition to the spatial patterns obtained via MSCNN-AE. This model uses an autoencoder for MSCNN and LSTM training, without labels, i.e., without supervision. This study was conducted on the cybersecurity-referenced datasets CICDDoS2019 and UNSW-NB15, and the best accuracy was 93.76%, with an F1 score of 92.26%.

Amy Thiandoum [25] worked on the XGBoost and CNN-LSTM algorithms for intrusion detection in the IoT over the MQTT (Message Queuing Telemetry Transport) protocol. In his work, he presented the vulnerabilities of this protocol and highlighted the promising potential of AI, specifically Machine Learning, for proactive security. To this end, the two models were compared, and the XGBoost algorithm performed better on the RT-IoT2022 dataset, achieving an accuracy

of 99.88% and an F1 Score (weighted average) of 99.88%. It was chosen because of its speed during training (38.17 s response time, 0.34 s response time) and consuming less memory resources (443.12 MB) compared to the CNN model, which has 66.08 s training time, 1.940 s response time, and 1059 MB of memory space.

Yagmur Yigit and *al.* [28] have implemented an intelligent DDoS attack detection mechanism for autonomous core networks based on digital twinning. The mechanism, using online learning, employs the MLP algorithm and the YANG (Yet Another Generation) model, with the AutoFS module for data processing. The system was trained on two datasets: one from CICDDoS2019 and the other from ToN IoT. The system achieves a detection rate of 97% with a latency of 15 minutes. However, while the detection approach delivers excellent performance, deployment would be costly and maintenance time-consuming.

Ussatova and *al.* (2022) compare in [26] eight machine learning algorithms (naive Bayes, logistic regression, support vector machine, k-nearest neighbors, decision tree, random forest, XGBoost, and CatBoost) for DDoS detection in software-defined networking (SDN), showing that ensemble methods (RF, XGBoost) achieve 99% accuracy with SMOTE, surpassing SVM-based approaches. Their study highlights the importance of character selection (IG/F-tes) and data balancing, although limited to an SDN context only. The dataset comprises 104,345 rows with 63,335 benign labels and 40,504 malicious labels. It is sourced from various internet sources.

3 Comparison of AI models in detection

A thorough analysis of various Machine Learning and Deep Learning models, researched by numerous authors, is a crucial step in our research process. This has allowed us to identify the most relevant and effective ML/DL models for cyber threat detection.

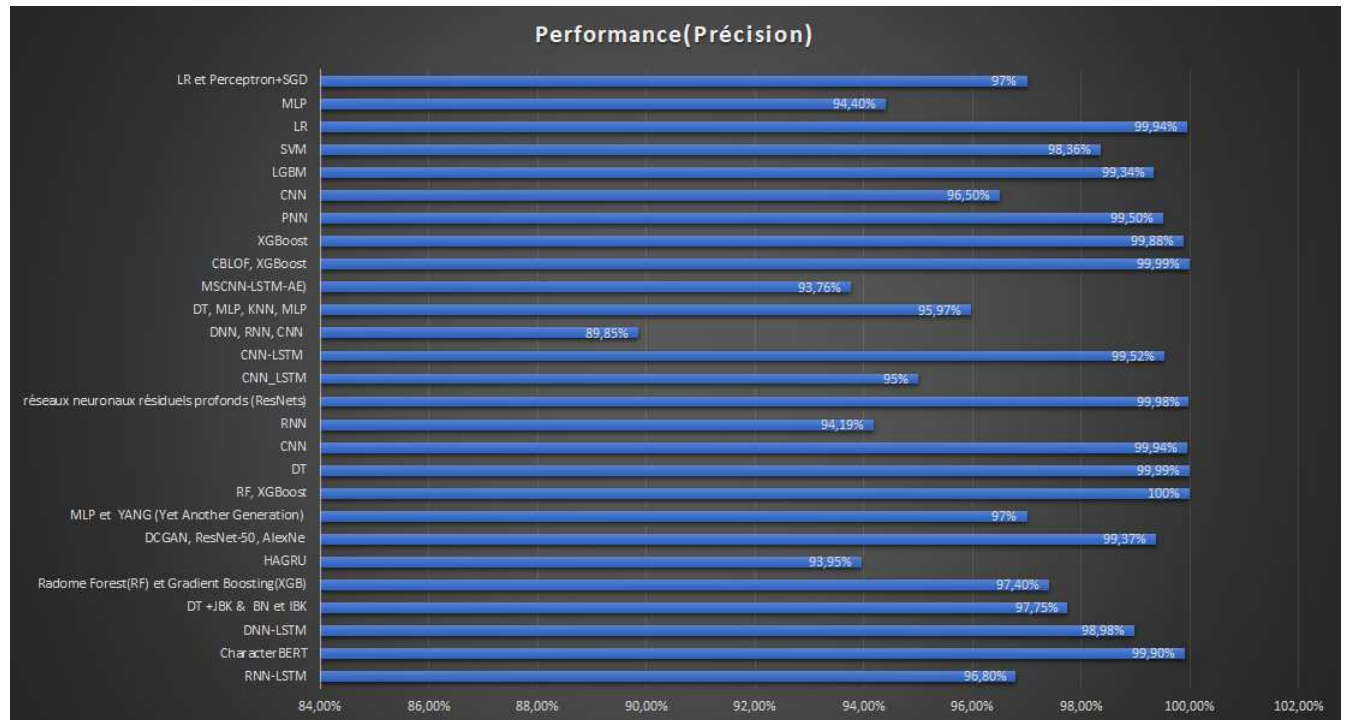


Figure 1: Comparison of anomaly detection models

The graph 1 presents the various artificial intelligence techniques applied in cybersecurity to detect threats. We evaluated the results of this research on the accuracy achieved by machine learning and deep learning models. The majority of the models presented achieve an accuracy greater than 94%, thus illustrating the effectiveness of AI techniques in cybersecurity. We distinguish two main classes: on the one hand, machine learning-based algorithms such as logistic regression (LR), stochastic gradient descent (SGD) perceptron, SVM, random forest (RF), XGBoost, and decision tree, which achieve accuracy ranging from 97% to 99.94%, further demonstrating their effectiveness; on the other hand, deep learning architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), LSTMs, or hybrid architectures such as CNN-LSTM, DNN-RNN-CNN, etc., which have an accuracy of 99%. Hybrid models such as Decision Tree and Naive Baye, or Random Forest and XGBoost, also demonstrate high accuracy, highlighting the appeal of ensemble approaches. However, these deep and hybrid architectures require significant computing power and substantial datasets to reach their full potential. Conversely, simpler models like Random Forest and XGBoost, while being more lightweight, offer high accuracy. They thus provide a good balance between robustness and speed of execution. They are less complex and more easily interpreted.

4 Comparative and critical study of the journal's solutions

The work of Zainab Sahib Dhahir [7], Amardeep Singha and *al.* [1], Amy THIANDOUM [25] and Yagmur Yigit and *al.* [28] highlights the diversity of intrusion detection approaches based on artificial intelligence.

Their results demonstrate that models such as LSTM and CNN have a high computational cost and significant interpretation complexity. Model performance depends on the data on which they are trained. The authors [7, 18, 23] use older datasets such as NSL-KDD, an enhanced version of KDD'99 (created in 1999) published in 2009, and CIC-IDS2017, which does not fully capture the complexity of current DDoS attacks. It should also be noted that the work in the review does not present a categorical detection solution for modern DDoS attacks.

These intelligent detection approaches, while generally effective, each have methodological and practical limitations that justify proposing a new solution. It is essential to develop a detection solution that is simultaneously high-performing, lightweight, and easy to explain—a security mechanism capable of proactively detecting cyber threats in real time. With this in mind, this paper proposes, in the following section, a methodological approach for DDoS attack detection. This solution combines efficiency, simplicity, and adaptability to the operational constraints of real-world environments.

5 Proposed method

5.1 General Architecture

Based on the CICDDoS2019 dataset, an intelligent intrusion detection system was developed. After applying a preprocessing step to the raw dataset, normalization was applied so that the values of all features were between 0 and 1. The training set was used to train the model, and the test set was used to evaluate the trained model. After preparing the data, we trained the machine learning algorithms: Random Forest, Decision Tree, Extra Trees, XGBoost, KNN, and a deep learning algorithm: MLP. We optimized each hyperparameter of the models using the GridSearchCV algorithm combined with cross-validation. Finally, the models were evaluated against the test set, and we compared the results to identify the models that performed best on these categories of DDoS attacks. The proposed system is illustrated in the figure.2.

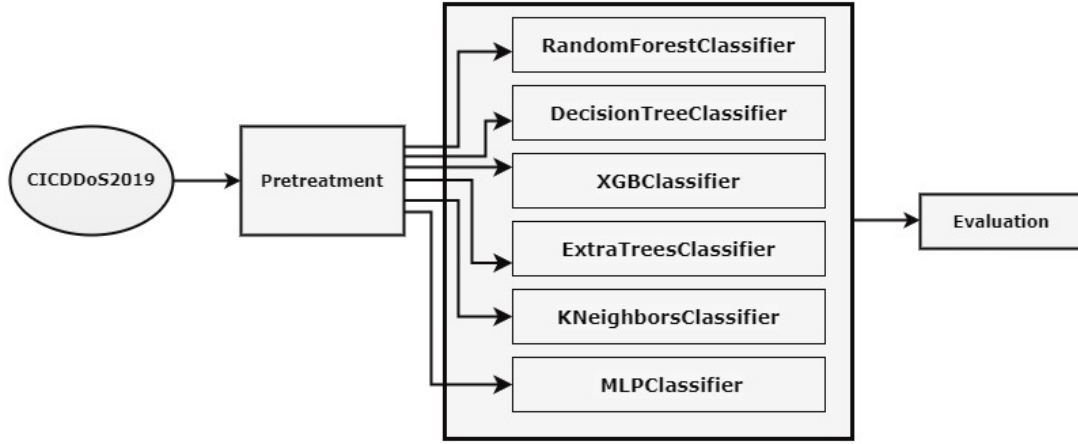


Figure 2: Proposed model architecture for DDoS detection

The architecture described above follows a flow, which we present in the figure ??, and we develop this flow in the following section, which represents the different steps up to the evaluation of our proposed detection model.

As data was recorded, including network traffic (Pcaps) and event logs (Windows and Ubuntu event logs) per machine. To extract features from the raw data, the authors used CICFlowMeter-V3 and extracted over 80 traffic features, then saved them in CSV format for each machine, making them directly usable and freely accessible.

The dataset consists of two zip files (2 GB each): a training dataset of 11 CSV files and an 876 MB test dataset of 7 CSV files. We manually analyzed the entire dataset to verify the presence of normal and malicious traffic. Some attack types in the training dataset, such as *Portmap*, were not present in the test dataset, and some attack types in the test dataset, such as *TFTP*, *NTP*, *DNS*, and *SNMP*, were not present in the training dataset. We therefore removed these files, allowing us to keep only the attacks present in both the training and the test, ensuring a consistent evaluation and avoiding biased learning. We thus have a total of 120,065 instances for training and 38,973 reserved for testing, with 78 variables.

5.2 Data preprocessing

This step aims to prepare the dataset for training and evaluating our different models. Figure 3 presents these necessary phases that we applied during the processing of our data, and we detail them later in this work.

5.3 Labeling resolution

The test data labels were incorrect. This data consists of 7 CSV files representing 8 classes. Of these classes, 7 correspond to DDoS attacks, while the 8th class is dedicated to normal traffic (the BENIGN class). Here are the changes made to the class labels in the test data so that they match those in the training dataset:

- The DrDoS_UDP class has been changed to UDP,
- The UDP-lag class has been changed to UDPLag,
- The DrDoS_MSSQL class has been changed to MSSQL,
- The DrDoS_LDAP class has been changed to LDAP,
- The DrDoS_NetBIOS class has been changed to NetBIOS,
- The Syn class has been changed to Syn,
- The Benign class has been changed to Benign.

The training dataset does not contain data on the WebDDoS attack, so we cannot use it for

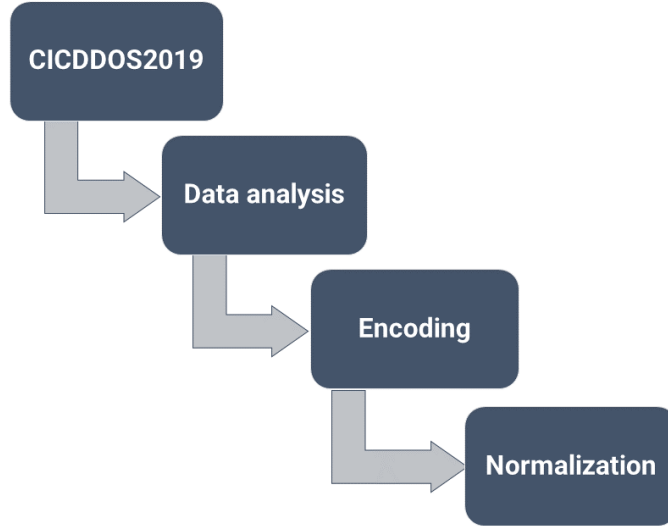


Figure 3: Data preprocessing phase

Table 1: Training and test dataset divided into classes

attack classes	Number of instances for the test	Number of instances for the train
Benign	10847	40980
Syn	533	47246
UDP	10420	17795
UDPLag	8872	55
NetBIOS	598	475
MSSQL	6212	8434
LDAP	1440	1885
WebDDoS	51	-
Total	38 973	116 870

training. We remove it from the test dataset. Therefore, the test dataset is reduced to **38,922** instances in total.

High-quality data always produces a high-quality model, meaning the performance of our models depends on the quality of the data on which they are trained. We therefore performed several preprocessing steps on the dataset.

First, we filtered the variables in the dataset into numeric, categorical, and high-cardinality categorical columns. This step allows us to better tailor the processing to each variable, particularly for encoding, normalization, and feature selection.

The following categorical columns *Bwd PSH Flags*, *Fwd URG Flags*, *Bwd URG Flags*, *FIN Flag Count*, *PSH Flag Count*, *ECE Flag Count*, *Fwd Avg Bytes/Bulk*, *Fwd Avg Packets/Bulk*, *Fwd Avg Bulk Rate*, *Bwd Avg Bytes/Bulk*, *Bwd Avg Packets/Bulk*, *Bwd Avg Bulk Rate* have a unique value; they have been removed because they do not contribute to the variability of the dataset and do not provide any useful information for learning. Duplicate rows were removed from our dataset to avoid bias in model training and ensure data quality.

To avoid redundancy between explanatory variables, a correlation analysis was performed on all numerical variables in the dataset. A correlation matrix was calculated, with the following columns: 'Bwd Packets Length Total', 'Fwd Packet Length Mean', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow IAT Std', 'Flow IAT Max', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Std', 'Bwd IAT Max', 'Fwd Packets/s', 'Packet Length Min', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'RST Flag Count', 'Avg Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size', 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets', 'Subflow Bwd Bytes', 'Fwd Seg Size Min', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Max', 'Idle Min'. Columns exhibiting an absolute correlation greater than 0.8 with other variables were identified. These highly correlated columns were removed from the training and test datasets. This step aims to reduce multicollinearity, prevent overfitting, and improve the performance and stability of cyber threat detection models.

The categorical column, such as *Protocol*, has been converted to a numeric column. However, other categorical columns, such as IP Address and Timestamp, have been removed. These features were considered to be related to connection information and do not represent the properties of DDoS attacks, as the latter can be launched at any time by any machine against any victim machine. For this reason, the features 'Flow ID', 'Source IP', 'Destination IP', 'Timestamp', and 'Inbound' have also been removed, leaving only network traffic features.

The characteristics that remain for the rest of our study are presented in the table below.

Table 2: Set of characteristics used for DDoS detection

Features	Types	Features	Types
Protocol	int8	Bwd IAT Total	float32
Flow Duration	int32	Bwd IAT Mean	float32
Total Fwd Packets	int32	Bwd IAT Min	float32
Total Backward Packets	int16	Fwd PSH Flags	int8
Fwd Packets Length Total	float32	Fwd Header Length	int64
Fwd Packet Length Max	float32	Bwd Header Length	int64
Fwd Packet Length Min	float32	Bwd Packets/s	float32
Fwd Packet Length Std	float32	SYN Flag Count	int8
Bwd Packet Length Max	float32	ACK Flag Count	int8
Bwd Packet Length Min	float32	URG Flag Count	int8
Flow Bytes/s	float32	CWE Flag Count	int8
Flow Packets/s	float32	Down/Up Ratio	float32
Flow IAT Mean	float32	Init Fwd Win Bytes	int32
Flow IAT Min	float32	Init Bwd Win Bytes	int32
Fwd Act Data Packets	int16	Active Mean	float32
Active Std	float32	Idle Std	float32

We used `LabelEncoder()` to encode the label column.

5.4 Normalization

To harmonize the scaling of the numerical variables, normalization using the **MinMaxScaler** method was applied. This transformation brings the values of each attribute into the interval $[0, 1]$, without altering the original distribution. The scaler was adjusted on the training and test data to ensure consistency. This step is particularly important to guarantee the stability and performance of scale-sensitive algorithms, such as neural networks or distance-based methods.

The normalization is given by the following formula:

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (1)$$

avec

X : is the initial value.

$X_{\min}$: is the smallest value in the dataset.

$X_{\max}$: is the largest value in the dataset.

X_{normal} : is the normalized value, between 0 and 1.

5.5 Selection of algorithms for DDoS detection

The choice of our models was a crucial step in our research. We selected six classifiers: Decision Tree, KNN, Random Forest, Extreme Gradient Boosting, Extra Tree, and MLP. This choice was justified by previous work and their performance on imbalanced datasets. The work in Hamarsheh, I. Ashqar, and Hamarsheh [10], Jaiswal [13], Dhahir [7] highlighted the performance of these different models on evaluation metrics such as precision, accuracy, F1 score, etc. The models we propose are also recognized in Sharma et al. [21] like simple models, easy to understand and analyze, and have strong resistance to underfitting. The behavior of these models is influenced by their hyperparameters. A hyperparameter is a model parameter defined before training to optimize its performance. Each classifier has several hyperparameters, defined according to the type of data on which the model is to be evaluated. In our experiment, we considered hyperparameters applied in the context of multiclass classifications with unbalanced classes. Below, we define each hyperparameter and the mathematical equation associated with different classifiers.

1. Rando Forest and Extra Tree

- **n_estimators**: The number of trees in the forest. The more there are, the more stable the prediction, but the longer the training time.
- **max_depth**: It defines the maximum depth of each tree and allows control of the model's complexity.
- **min_samples_split**: Minimum number of samples for a node to be split. More splits make the model more complex.
- **class_weight**: Allows for automatic adjustment of the weight of classes with very few representations.

The algorithm's operation is described by the following mathematical equation:

$$\hat{y} = \frac{1}{N} \sum_{i=1}^N h_i(x) \quad (2)$$

Avec:

$\hat{y}$ The algorithm's final prediction

N , the total number of trees;

h_i : tree prediction i .

2. XGBoost

- **n_estimators**: The number of trees in the forest. The more there are, the more stable the prediction, but the longer the training time.
- **max_depth**: It defines the maximum depth of each tree and allows control of the model's complexity.
- **learning_rate**: Learning rate. Requires many trees and is stable if the value is small (0.01).
- **subsample**: defines the fraction of data for each tree.
- **colsample_bytree** : fraction of features for each tree. It allows for further regularization and prevents certain attributes from dominating.
- **gamma** : penalty for adding a split.

It is given by the following mathematical equation[10]:

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F} \quad (3)$$

3. MLPClassifier

- **hidden_layer_sizes** : Size of hidden layers. More neurons, a more powerful but more complex model.
- **activation** : activation function. *relu* : Fast, efficient for deep networks. *tanh* : better for small values, but slower
- **solver** : optimization algorithm. *adam* : Very effective for large datasets.
- **learning_rate_init** : initial learning rate.

4. Decision Tree

- **max_depth** : Maximum depth.
- **min_samples_split** : minimum number of samples for a split.

5. KNN

- **n_neighbors** : Number of neighbors for classification. If k is smaller, the model is sensitive to noise; if k is very large, the model smooths but risks ignoring small classes.
- **p** : defines the choice of distance. $p=1$ for the Manhattan distance (L1) and $p=2$ for the Euclidean distance (L2).
- **weights** : It measures the disparity between two instances. This affects how the algorithm defines the neighborhood of a new instance. *uniform*: all neighbors have the same weight, *distance*: the closest ones have more influence, often better for noisy data.

It is generally based on the calculation of Euclidean distance given by the following formula:

$$d(x_i, y_i) = \sqrt{\sum_{j=1}^n (x_{ij} - y_{ij})^2} \quad (4)$$

Avec :

x_i :vector of the point to be classified

y_i :vector of a point in the dataset

n :number of features

5.6 Cross-Validation with GridSearchCV

Cross-validation is defined as an essential technique in machine learning for evaluating a model's performance more reliably, reducing the risk of overfitting. It is a statistical validation method where the data is divided into several subsets to repeatedly test and train a model on different portions of the training dataset. In our case, we chose a 5-fold cross-validation (CV=5). Specifically, the dataset is divided into five equal parts. (figure 4): At each iteration, four layers are used to train the model and the fifth to validate it, alternating until each layer has served as a test once. This approach provides a more reliable evaluation than a simple separation of training and testing, as it reduces the risk of overfitting. The choice of five layers represents a good compromise: it limits computation time while offering a stable performance estimate. This methodical approach is recommended in the work of Meenu Bhagat. [20] and Victor Wandera [6]. Their studies allowed us to highlight the advantages of this mechanism:

- **Increased model accuracy:** Cross-validation allows us to evaluate the performance of a model on unknown data, thereby increasing the accuracy of the model.
- **Prevents over-adjustment:** Cross-validation makes it possible to detect models that overfit training data by evaluating model performance on invisible data.
- **Hyperparameter settings:** To determine the ideal hyperparameter values that produce the highest performance.
- **Model selection:** By comparing the performance of several models and choosing the best one, cross-validation improves the overall performance of the model.
- **Robustness assessment:** By evaluating the performance of a model across multiple datasets and parameters, cross-validation helps to determine the robustness of the model.
- **Improves the generalizability of the model:** By evaluating a model's performance on unseen data, cross-validation improves a model's ability to generalize to new data.

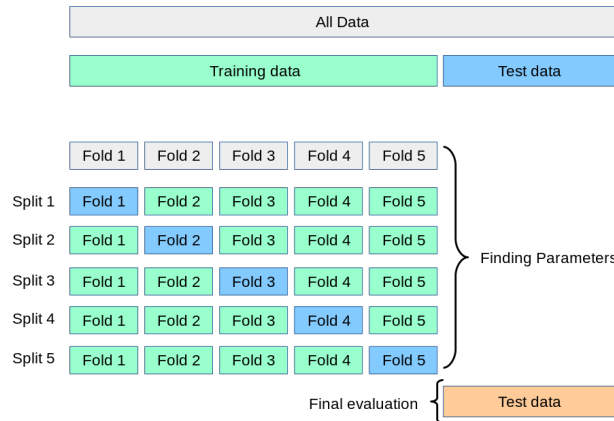


Figure 4: Cross-Validation in 5 splits [24]

5.7 Hyperparameter Optimization with GridSearchCV

Hyperparameters play a crucial role in the performance of machine learning models. These are parameters or configurations that are not learned from the data, but defined before training. These parameters influence the behavior of the models and impact their predictive ability.

GridSearchCV (GridSearch Cross-validation) is a technique used in machine learning for hyperparameter optimization. It's a method for finding the best combination from a user-defined dictionary of hyperparameters for each model. To evaluate model performance, GridSearchCV uses the cross-validation technique described earlier. The following table presents the hyperparameter dictionary and the best hyperparameters selected by GridSearchCV.

Table 3: The best hyperparameters obtained by GridSearchCV

Algorithms	Hyperparameter dictionaries	Best hyperparameters	F1_score CV
Random Forest	n_estimators: [100, 200, 300], class_weight: [None,'balanced'] ,max_depth: [10, 20, 30], min_samples_split: [2, 5]	max_depth: 30, 'class_weight': 'balanced', min_samples_split: 5, n_estimators: 200	0.9869
Decision Tree	max_depth: [None, 10, 20, 30], class_weight: [None,'balanced'], min_samples_split: [2, 5]	max_depth: 20, class_weight: balanced, min_samples_split: 2	0.9869
Extra Trees	n_estimators: [100, 200, 300], class_weight: [None,'balanced'], max_depth:[10, 20, 30]	max_depth: 30, 'class_weight': 'balanced', n_estimators: 200	0.9884
KNN	n_neighbors: [3, 5, 7, 10], weights: ['uniform', 'distance'], p: [1, 2]	n_neighbors: 7, p: 2, weights: uniform	0.9843
MLP Classifier	hidden_layer_sizes: [(100), (128, 64)], learning_rate_init: [0.001, 0.01], alpha: [0.0001, 0.001, 0.01], activation: ['relu', 'tanh'], solver:['adam'], max_iter: [500]	activation:relu, learn- ing_rate_init: 0.001, hid- den_layer_sizes: (128,64), max_iter: 500, solver: adam, alpha: 0.001	0.9725
XGBoost	n_estimators: [100,200,300], max_depth: [3,6,10,20], learn- ing_rate: [0.01,0.05,0.1], colsam- ple_bytree: [0.8], gamma: [0, 0.1, 0.2], subsample: [0.8]	learning_rate: 0.1, max_depth: 6, n_estimators: 100, subsample: 0.8	0.9854

6 Evaluation metric

The validation of a machine learning model relies on evaluating its predictive performance. This evaluation must be representative of the model’s performance in a real-world situation. Predictive performance is defined by metrics that are considered both according to the types of problems (classification or regression) to be solved and according to the data. In our case study, where the data is unbalanced, according to Gaël and *al.* [27], and Mathieu [15], The validation of a machine learning model relies on evaluating its predictive performance. This evaluation must be representative of the model’s performance in a real-world situation. Predictive performance is defined by metrics that are considered both according to the types of problems (classification or regression) to be solved and according to the data. In our case study, where the data is unbalanced, according to Gaël and.

The F1 score is calculated from the accuracy and recall values. The model’s accuracy can be measured by calculating the percentage of true positives among all the model’s positive predictions. Recall corresponds to the percentage of true positive samples in the dataset that include genuine positives, compared to the total number of actual positive samples. The F1 score ranges from 0 to 1, with higher scores indicating superior performance. A perfect F1 score of 1.0 indicates that the model’s accuracy and recall are perfect, while a score of 0 indicates that the model’s results are completely inaccurate. The F1 score is given by the following key mathematical formula:

Score F1 : The F1 score is the harmonic mean of accuracy and recall.

$$\text{Score F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

False Positive Rate (FPR): The false positive rate (FPR) measures the proportion of negative instances that are incorrectly classified as positive by the model. In other words, it’s the probability that the system will falsely alert when there is no actual threat. A high FPR indicates an excess of false alarms, which can lead to a loss of confidence in the detection system and increase verification costs. Mathematically, the FPR is defined as follows:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

or FP represents false positives and TN the real negatives.

False Negative Rate (FNR): The false negative rate (FNR) measures the proportion of positive instances that are incorrectly classified as negative. In other words, it is the probability that the model misses a real threat and considers it normal. A high FNR is particularly critical in cybersecurity, as it means that real attacks are going undetected. The FNR is defined by the following formula:

$$\text{FNR} = \frac{\text{FN}}{\text{FN} + \text{TP}}$$

or FN represents false negatives and TP the real positives.

ROC AUC Score : The area under the learning curve measures the distance between the learning curve and the validation curve. That is, the distance between the learning algorithm’s understanding of the data and its validation by the test data.

7 Reference and implementation details

The field of machine and deep learning requires significant hardware resources. We chose to work on Google Colab.

Google Collaborate, Python, short for Google Colaboratory, is a platform offered free of charge by Google for writing and running Python code, accessible via a browser at the URL

<https://colab.research.google>. Based on Jupyter notebooks, Google Colab facilitates access to computing resources and common machine learning libraries. Although we used the free version, it proved extremely useful for computationally intensive tasks, such as using Deep Learning models (MLP models). This accelerated the model training process, thus reducing the time required to obtain results. We chose to use Python as our programming language. Python is an interpreted, multi-paradigm, and multi-platform programming language, and it is also a more common and popular language for machine learning and artificial intelligence due to its flexibility and the large number of open-source software libraries available.

jupyter Notebook :Jupyter Notebook is an open-source web application for creating and sharing documents. Formerly known as IPython Notebooks, it is a web-based interactive computing environment that allows Python users to create notebook documents. The term "notebook" refers to different entities; it can be the Jupyter web application, the Jupyter Python web server, or a Jupyter document.<https://jupyter.org> **The libraries used in our project:**

Scikit-Learn allows us to experiment with different predefined machine learning and data analysis techniques and algorithms quickly and easily.

Pandas et Numpyare used for data manipulation (loading, reorganizing, and processing data).

Matplotlib :Matplotlib is a Python library used to create animated, 2D statistical graphs such as histograms, scatter plots, etc. Its documentation is available on its official website: <https://matplotlib.org>. We use it to visualize our data.

8 Results and discussion

8.1 F1 Metric Analysis: Score by Class

The figure 5, This section presents the comparative results of the evaluation of the six models by attack class. The Decision Tree, KNN, and MLP models, with f1 scores of 0.995, 0.994, and 0.995 respectively, stand out in the classification of the benign class. The LDAP attack is very well classified by the Random Forest (0.908), KNN (0.906), and Extra Tree (0.901) models, but Random Forest performs slightly better than the others. All our models perform well on the MSSQL class, but the best is RandomForest (0.948). For NetBIOS, Random Forest (0.802) and KNN (0.801) are ranked as the models that perform best on this attack category. XGBoost is the only model that reaches 0.274 on the Syn class, but remains very weak, which means that the patterns of this class are complex for our models to detect. For the UDP class, all models achieve an f1 score of 0.895, but Random Forest stands out with an f1 score of 0.897. The UDPLag class was very poorly ranked by our models; only Decision Tree obtains a mediocre f1 score of 0.043, which could be caused by its minority in the test set.

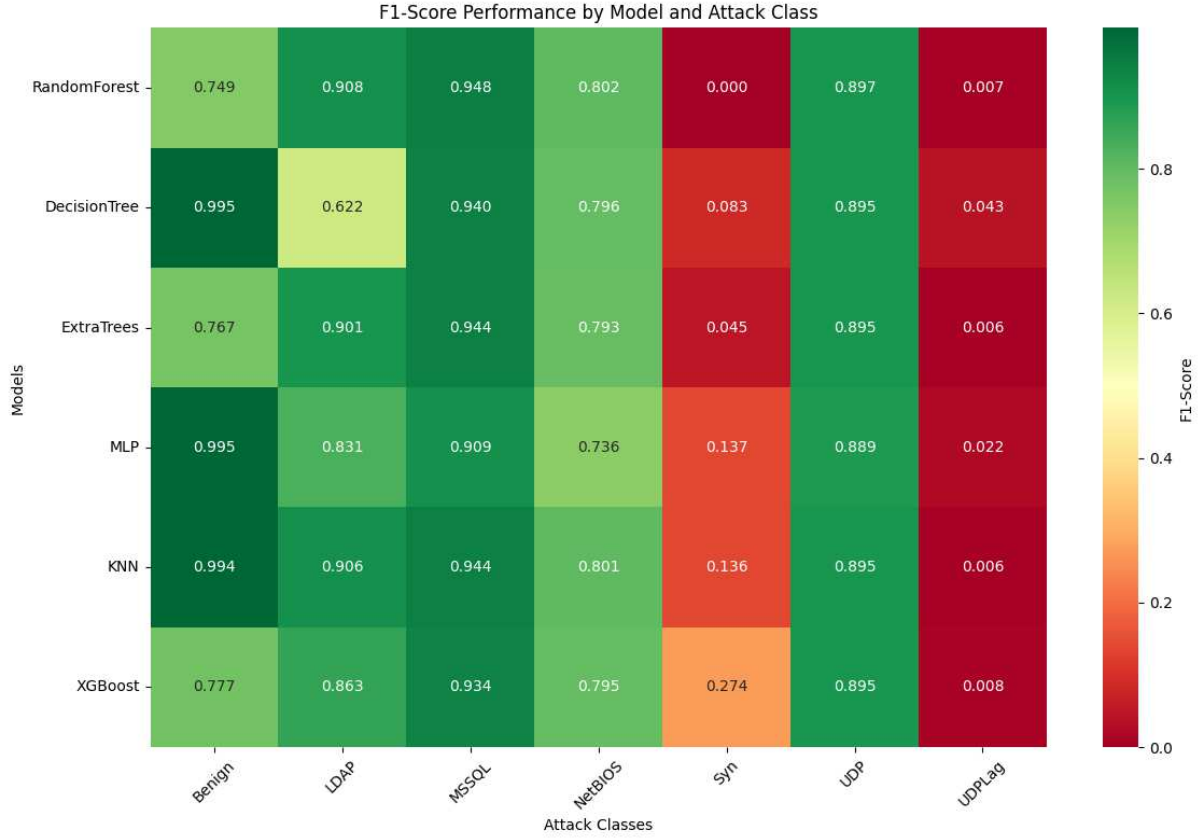


Figure 5: Comparison of DDoS detection models by class using the f1 score

8.2 Analysis of the AUC/ROC curve by class

The figure 6 The graph presents the Receiver Operating Characteristic (ROC) curves for each model. The derived metric is Area Under the Curve (AUC); the closer the curve is to the upper left corner, the better the performance. These curves allow for a visual analysis of the balance between the true positive and false positive rates of the model, thus highlighting its ability to differentiate between benign and malicious traffic.

For most classes, we observe a very high AUC (greater than 0.95), indicating that the models distinguish the classes well. Random Forest stands out in particular with a macro AUC score of 0.9528.

8.2.1 Overall comparative analysis of models with F1 Score and AUC

To better understand the performance of the models in the face of data imbalances, we compare the macro F1 score of the models, which corresponds to the unweighted arithmetic mean of the F1 scores calculated for each class, and the weighted F1 score, which corresponds to the weighted arithmetic mean of the F1 scores, and the AUC curve, which plots TPR (true positive rate) as a function of FPR (false positive rate) (figure). 7).

On the left of the figure 7, we observe the MLP and KNN models, which are two of the best models, having respectively F1-Score (Macro): 0.6456, F1-Score (Weighted): 0.7093 and F1-Score (Macro): 0.6688, F1-Score (Weighted): 0.7162. The best models for AUC macro are Random Forest (AUC (Macro): 0.9528) and XGBoost (AUC (Macro): 0.9445).

This analysis resulted in two robust and stable models, namely XGBoost with an acceptable F1-Score (Macro): 0.6494 and F1-Score (Weighted): 0.6549 and an AUC (Macro) reaching 0.9445, and Random Forest with an acceptable F1-Score (Macro) of 0.6158 and F1-Score (Weighted):

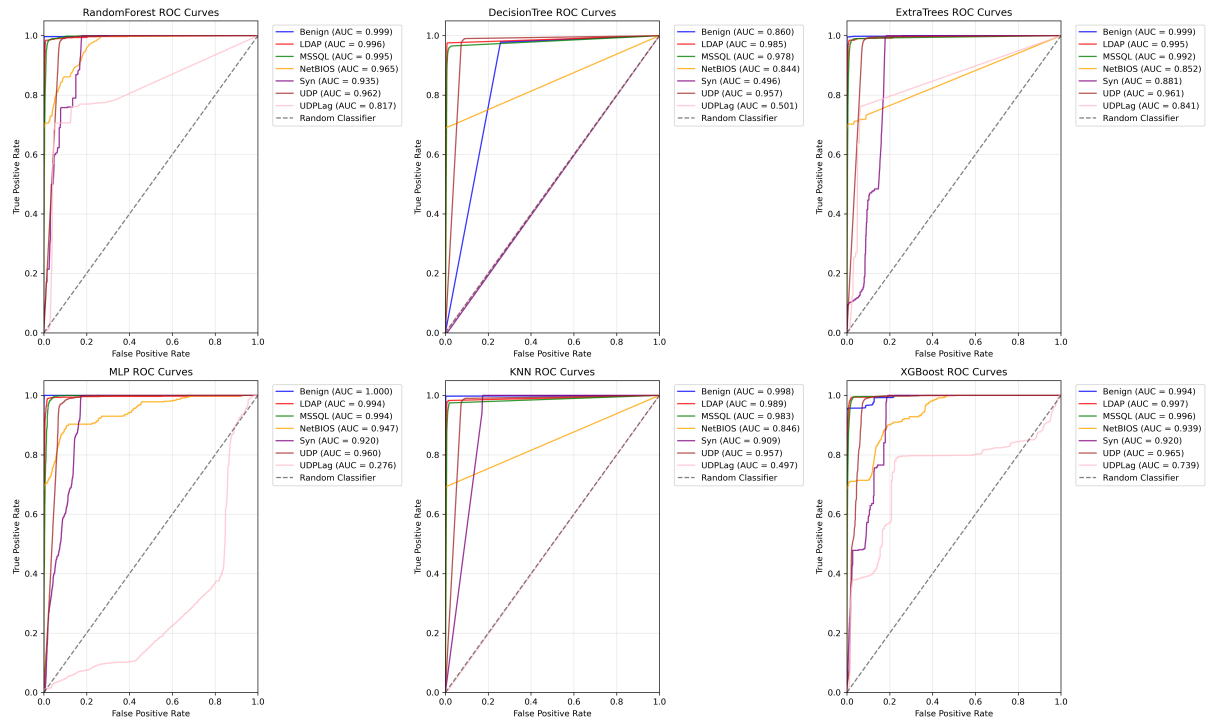


Figure 6: Comparing models by class using the AUC curve

0.6476 and an AUC (Macro): 0.9528.

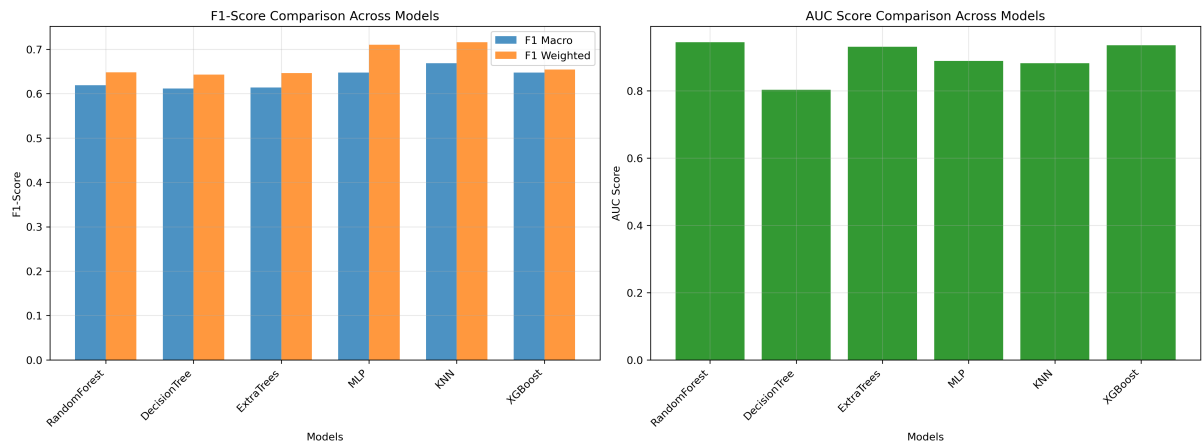


Figure 7: Overall comparison of models using F1 score and AUC

8.3 Comparative study of FPR, FNR and prediction time metrics

Our study focused on evaluating the FPR and FNR metrics, which define the false positive (false alert) and false negative (undetected positive case) rates. In the table 4, we observe that all our models produce very few false alerts and make fewer errors, but models such as Random Forest and ExtraTrees have false negative rates of 0.2 and 0.3, respectively. Some models, like XGBoost, MLP, KNN, and Decision Tree, produce fewer false negatives, but KNN has a high prediction time (68.5182 s), making it slow and less suitable for detection. The Decision Tree, Random Forest, and XGBoost models have acceptable prediction times of 0.0068, 0.1233, and 0.0788, respectively.

Table 4: Evaluation of the trade-offs between false positives, false negatives, and prediction time

Modèles	Average FPR	Average FNR	Forecast times
Random Forest	0.0506	0.2427	0.1233
Decision Tree	0.0400	0.2742	0.0068
Extra Trees	0.0498	0.3371	0.3112
MLP	0.0409	0.2093	0.3504
KNN	0.0393	0.2023	68.5182
XGBoost	0.0491	0.1095	0.0788

8.4 Interpretation of the results

Based on our results, two models, namely Random Forest and XGBoost, stood out and are ranked as the best models for DDoS attack detection. These models were trained and tested on an unbalanced test dataset. We chose to leave the classes unbalanced to obtain true detection performance. These models demonstrated their robustness in classifying certain attack classes, including LDAP, MSSQL, NetBIOS, and UDP. The data imbalance problem is a recurring issue in machine learning, and set-based methods (Random Forest, XGBoost) are recognized as common techniques for improving the performance of machine learning models on unbalanced datasets. [14].

Our models offer ideal efficiency for integration into intrusion detection systems to enhance network security. These evaluated models (Random Forest, XGBoost) can therefore be used to strengthen existing defenses and reduce response time to emerging threats.

8.5 Limitations and challenges of our research

Achieving a perfect academic research project is virtually impossible; our work is no exception.

The research we present has limitations and challenges that we encountered during its execution. In this section, we will list the gaps identified in our results and outline some of the challenges we faced during our research. We will also propose possible solutions to overcome these challenges and improve future work.

1. The gaps identified in our approach are:

The lack of data and resources: The lack of data in certain categories of DDoS attacks was extreme; our models focused on classifying only a specific type of attack. This is insufficient for a cybersecurity system; we need more data on Syn and UDPLag attacks. Throughout our research, we used the free version of Google Colab, which limits access to computing power resources for developing a general-purpose model capable of detecting multiple types of cyberattacks.

The presence of false positives and false negatives : The accuracy of our models is not sufficient to guarantee the completeness of the defense; that is to say, the models can make prediction errors, which could cause false alerts to the system.

2. The challenges we encountered during our research :

- The lack of data: simulating and obtaining personal data poses a significant challenge to our approach. We could have implemented a simulation system to conduct DDoS attack types on a server to collect this data for training our models, but this scientifically valuable approach requires substantial resources. Therefore, we focused on cybersecurity benchmark data to find the CICDDoS2019 dataset, which represents real-world DDoS attacks on networks.
- Rapid technological evolution: As technology evolves, new cyberattacks emerge and challenge cyber defenders. Therefore, defending target systems requires more than just a specific model for a number of attacks.

To address the identified gaps and challenges, it would be possible to improve the quality and representativeness of the analysis data. The Syn and UDPLag classes have very complex patterns that our models struggle to detect; therefore, we propose exploring hybrid (combined) models (Random Forest and XGBoost) or neural networks. Despite their computational cost, these models could better capture the patterns of these distributed denial-of-service attack categories in future work.

8.6 Discussion of our results

The use of machine learning algorithms is the subject of research by several scientists. In our study, we chose to compare them in the table 5 the performance of our best models with the work carried out by Ahmad Harmarshe and *et al.* [1].

Their study on DDoS detection using the CICDDoS2019 dataset relies on data preprocessing and an undersampling approach to balance benign and malicious flows. They evaluated four supervised learning models (Random Forest, Decision Tree, XGBoost, SVM) on the test set. The reported performance reached an overall accuracy of 68.9% for Random Forest, 63.5% for XGBoost, and only 47.8% for Decision Tree. Compared to our methodological approach, which relies on hyperparameter optimization with GridSearchCV, we obtained better results. Their methodology evaluates models using accuracy, which can be misleading in the case of unbalanced classes; nevertheless, we chose to compare our best models with the work in the review. Our models have improved accuracy, Random Forest achieved 73.94% against 68.9% and the XGBoost model achieved 74.14% against 63.6%.

Algorithmes	Accuracy	Accuracy [1]
Random Forest	73.94%	68.9%
XGBoost	74.14%	63.5%

Table 5: Comparisons of our results with those of Ahmad Hamarshe et *al.* [1]

Conclusion

This work allowed us to detail a methodology focused on using Machine Learning and Deep Learning algorithms to detect Distributed Denial-of-Service (DDoS) attacks. DDoS attacks are a variant of Denial-of-Service attacks, but unlike DDoS attacks, they are carried out by multiple botnets, which complicates their detection by traditional intrusion detection systems. Using the Google Colab environment and the Python programming language, we trained six algorithms—Decision Tree, Extra Tree, Random Forest, XGBoost, KNN, and MLP—on the CICDDoS2019 cybersecurity reference dataset. Analysis of this dataset enabled us to identify seven classes: six classes representing malicious traffic and one class representing normal traffic. These six classes include Syn, UDP, UDPLag, MSSQL, LDAP, and NetBIOS attacks. These classes are unbalanced in the training and test data. We balanced only the training data by increasing the number of samples from minority classes using the SMOTE technique. We then optimized the hyperparameters of each model for good performance with the GridSearchCV algorithm by applying K-FOLD cross-validation (with $cv=5$). We considered the F1 score, the AUC/ROC curve, and the FPR and FNR metrics to evaluate the performance of our models. Our results show that the Random Forest and XGBoost models are best suited for detecting LDAP, MSSQL, NetBIOS, and UDP attacks, but they fail considerably against Syn and UDPLag attacks due to the complexity of these attack patterns. The Random Forest model obtained F1 scores of 0.896, 0.944, 0.902, and 0.798, respectively. The AUC (macro) was 0.9528, the FPR was 0.05, and the FNR was 0.2. XGBoost achieved F1 scores of 0.896, 0.934, 0.862, and 0.793, respectively, along with an AUC(macro) of 0.9445, an FPR of 0.04, and an FNR of 0.2. These results indicate that detection based on these models is more accurate and effective in identifying types of distributed denial-of-service (DDoS) attacks. Our findings pave the way for proactive security measures in cybersecurity; they are promising, and we recommend integrating these models into intrusion detection systems.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

- [1] “A hybrid DNN-LSTM model for detecting phishing URLs”. In: *SI: APPLICATION OF MACHINE LEARNING FOR SECURITY* (8.08.2021). URL: [https://doi.org/10.1007/s00521-021-06401-z\(0123456789\)](https://doi.org/10.1007/s00521-021-06401-z(0123456789)).
- [2] Sahil Arora, Pranav Khare, and Sandeep Gupta. “AI-driven DDoS mitigation at the edge: Leveraging machine learning for real-time threat detection and response”. In: *2024 International Conference on Data Science and Network Security (ICDSNS)*. IEEE. 2024, pp. 1–7.
- [3] S Asha Varma and K Ganesh Reddy. “An AI based IDS framework for detecting DDoS attacks in cloud environment”. In: *Information Security Journal: A Global Perspective* 33.6 (2024), pp. 613–625.
- [4] Radjaa Bensaid et al. “SA-FLIDS: secure and authenticated federated learning-based intelligent network intrusion detection system for smart healthcare”. In: *PeerJ Computer Science* 10 (2024), e2414.
- [5] Radjaa Bensaid et al. “Toward a Real-Time TCP SYN Flood DDoS Mitigation Using Adaptive Neuro-Fuzzy Classifier and SDN Assistance in Fog Computing”. In: *Security and Communication Networks* 2024.1 (2024), p. 6651584.
- [6] Meenu Bhagat and Dr Brijesh Bakariya. “A comprehensive review of cross-validation and machine learning techniques”. In: *International Journal on Science and Technology (IJSAT)* Volume 16, number 1 (2025).
- [7] Zainab Sahib Dhahir. “A hybrid approach for effective DDoS detection in the Traffic network using feature engineering based on CBLOF and XGBoost”. In: *Journal of Future Artificial Intelligence and Technologies 2024 (September)* vol. 1, n° 2 (2024). DOI: [DOI: 10.62411/faith.2024-33](https://doi.org/10.62411/faith.2024-33).
- [8] Sneha Ghosh, Raman Kumar Goyal, and Kuntal Chowdhury. “Explainable AI-Driven Intrusion Detection System for DoS Attack Classification Using Deep Learning and Optimization Techniques”. In: *IEEE Access* 14 (2026), pp. 5618–5642.
- [9] Mohammed Sayeeduddin Habeeb and T Ranga Babu. “Network intrusion detection system: a survey on artificial intelligence-based techniques”. In: *Expert Systems* 39.9 (2022), e13066.
- [10] Ahmad Hamarshe, Huthaifa I.Ashqar, and Mohammad Hamarsheh. “Detection of DDoS Attacks in Software Defined Networking Using Machine Learning Models”. In: (2023).
- [11] Jim Holdsworth and Matthew Kosinski. “Cybersecurity”. In: (). Accessed online on 02/12/2024. URL: <https://www.ibm.com/fr-fr/think/topics/ddos>.
- [12] INTERPOL. *INTERPOL 2024 REPORT ON THE ASSESSMENT OF CYBER THREATS IN AFRICA*. 2024.
- [13] Sejal Jaiswal. “Multilayer Perceptrons in Machine Learning: A Complete Guide”. In: (2024). Accessed online on 08 juin 2025. URL: <https://www.datacamp.com/fr/tutorial/multilayer-perceptrons-in-machine-learning>.
- [14] Victor Wandera Lumumba et al. “Comparative analysis of cross-validation techniques: LOOCV, K-folds Cross-Validation, and Repeated K-folds Cross-Validation in Machine Learning Models”. In: *American Journal of Theoretical and Applied Statistics* 13, Issue 5 (2024).
- [15] Mathieu. “F1 Score in Machine Learning (Score F1 dans l’apprentissage automatique)”. In: (Janvier 2025). Accessed online on 15 août 2025. URL: <https://nugg.ad/fr/glossaire-ia/score-f1-apprentissage-automatique/>.

- [16] Sajid Mehmood et al. “Distributed Denial of Services (DDoS) attack detection in SDN using Optimizer-equipped CNN-MLP”. In: *PloS one* 20.1 (2025), e0312425.
- [17] Ashfaq Ahmad Najar and S Manohar Naik. “AE-CIAM: a hybrid AI-enabled framework for low-rate DDoS attack detection in cloud computing”. In: *Cluster Computing* 28.2 (2025), p. 103.
- [18] Alper Ozcan et al. “A hybrid DNN-LSTM model for detecting phishing URLs”. In: *SI: APPLICATION OF MACHINE LEARNING FOR SAFETY* (8.08.2021). URL: [https://doi.org/10.1007/s00521-021-06401-z\(0123456789\(\)\)](https://doi.org/10.1007/s00521-021-06401-z(0123456789())).
- [19] Sekoia. “Artificial Intelligence in Cybersecurity”. In: (). Accessed online on 02/12/2024. URL: <https://www.sekoia.io/fr/glossaire/ia-intelligence-artificielle-dans-la-cybersecurite/>.
- [20] Iman Sharafaldin et al. “Development of a realistic dataset and taxonomy on distributed denial-of-service (DDoS) attacks”. In: *53rd IEEE Carnahan International Conference on Security Technologies, Chennai, India, 2019*. 2019.
- [21] Vishwas Sharma et al. “Artificial intelligence-based intrusion detection system – Detailed investigation”. In: *Web ITM Conferences*. 2024. DOI: <https://doi.org/10.1051/itmconf/20246504002>.
- [22] S Siva Shankar et al. “A novel optimization based deep learning with artificial intelligence approach to detect intrusion attack in network system”. In: *Education and Information Technologies* 29.4 (2024), pp. 3859–3883.
- [23] M. Amaad UL Haq Tahir et al. “A hybrid model for detecting phishing sites using supervised learning algorithms”. In: (2020).
- [24] Prexawanprasut Takorn and Banditwattanawong Thepparit. “Improving memorization in minority classes through a new cluster-based approach: Oversampling technique”. In: (2024). DOI: <https://doi.org/10.3390/informatics1102003>.
- [25] Amy THIANDOUM. “Artificial intelligence for intrusion detection in the Internet of Things: the case of the MQTT protocol”. MA thesis. UFR of Sciences and Technologies, 2024.
- [26] Olga Ussatova et al. “Comprehensive DDoS attack classification using machine learning algorithms”. In: *Computers, Materials and Continua* 73.1 (2022), pp. 577–594.
- [27] Gaël Varoquaux and Olivier Colliot. “Evaluation of machine learning models and their diagnostic value”. In: (2023).
- [28] Yagmur Yigit et al. “Intelligent DDoS detection mechanism for standalone core networks based on digital twinning”. In: *IEEE Communications Standards Magazine* 6 (2023).