

Benchmarking ML-Based DDoS Detection Algorithms: Efficiency, Effectiveness, and Generalization

Zhilbert Tafa*, Aurora Uka

University for Business and Technology, Prishtina, Kosovo

Abstract—Distributed Denial of Service (DDoS) attacks are among the most significant cybersecurity threats today. Traditional DDoS detection systems struggle to cope with emerging attack vectors, while Deep Learning (DL)-based approaches are limited by their high computational complexity. Shallow Machine Learning (ML) techniques have been widely explored, either as standalone solutions or to complement DL approaches. However, the existing literature remains ambiguous regarding the trade-offs among classification accuracy, computational complexity, generalization, and the consistency of ML-based methods across different network environments. In this work, we compared six representative models spanning supervised, semi-supervised, and unsupervised ML. We used five datasets to assess their DDoS identification capability, computational efficiency, generalization, and cross-domain performance. XGBoost demonstrated the best balance between DDoS identification capability ($F1 = 0.984$) and computational efficiency. Isolation Forest (IF) showed solid discriminative performance ($F1 = 0.673$) at a very low computational cost, making it suitable for resource-constrained devices. Other examined methods excel in certain aspects while remaining limited in other metrics. All models exhibited strong generalization performance. Apart from Stochastic Gradient Descent One Class Support Vector Machine (SGD-OCSVM), all other models showed high consistency in DDoS identification across different network setups.

Keywords—Anomaly detection; autoencoder; DDoS; isolation forest; LOF; machine learning; OCSVM; RF; XGBoost

I. INTRODUCTION

Distributed Denial-of-Service (DDoS) attacks are among the most prevalent cyberattacks in recent years, causing severe service disruptions or degradation. Their potentially harmful repercussions for organizations, governments, and individuals emphasize the importance of robust detection and mitigation strategies [1].

DDoS attacks have reached an unprecedented level, with vast growth in volume, intensity, frequency, and attack vectors [2]. In a DDoS attack, the attackers typically use botnets of compromised networked devices (e.g., computers, servers, IoT devices, etc.) to coordinate attacks through a command-and-control (C2) server (Fig. 1).

DDoS threats are commonly categorized into two main groups: volumetric and low-rate (stealthy) attacks.



Fig. 1. DDoS attacks orchestrated using C2 servers and botnets.

Volumetric attacks are usually launched by flooding the network infrastructure with an intense number of packets, which can result in the consumption of network resources [3]. On the other hand, low-rate attacks typically target the application layer. Through long-lasting sessions, they mimic legitimate user behavior, which makes them particularly hard to detect [4].

With the emergence of new technologies, computer networks are becoming more structurally complex, diverse, and heterogeneous [5]. This expands the attack surface, increasing the likelihood of sophisticated, multi-vector attacks that combine volumetric and application-layer exploits [6]. Moreover, the rapid advancements in the field of artificial intelligence (AI) have made attacks easier to launch and harder to detect. Current models for analyzing DDoS attacks are often inadequate for addressing the complexity of emerging DDoS campaigns [7].

To overcome the limitations of traditional defense strategies, modern Intrusion Detection and Prevention Systems (IDPSs) have increasingly incorporated intelligent adaptive decision-making mechanisms. However, as these systems are often devised to operate across resource-constrained devices, achieving a balance between low computational complexity and high classification performance remains a challenge. While Deep Learning (DL) methods excel in classification performance, they require significant computational resources and a large amount of training data. Hybrid IDPS may include only lightweight models, or be combined with DL methods, depending on the application.

With the exception of the autoencoder (for anomaly detection), in this study, only representative shallow ML algorithms that may be useful for incorporation into the hybrid system are considered. We developed a benchmarking framework to evaluate prominent supervised, unsupervised, and semi-supervised algorithms for DDoS detection in terms of

*Corresponding author.

classification performance, computational efficiency, generalization, and performance consistency across different network environments.

To the best of our knowledge, this study is the first to provide a broad comparative analysis, offering insights into the most suitable ML methods for network traffic classification.

II. RELATED WORK

Academic research has shown a high level of interest in identifying combinations of data-driven models that best suit DDoS detection in diverse implementation scenarios, exploring statistical, entropy-based, ML, and DL methods. As our work focuses on “lighter” data-driven models, this section reviews recent and relevant studies in this domain.

Trigui et al. [8] measure training time, inference latency, peak memory usage, and model size of Extreme Gradient Boosting (XGBoost), Random Forest (RF), AdaBoost, K-Nearest Neighbours (KNN), Naïve Bayes (NB), and Support Vector Machine (SVM) models across two datasets. Although the models achieve impressive results, the choice of evaluation datasets may not reflect the real-world traffic environment.

In [2] authors evaluated RF, XGBoost, NB, Logistic Regression, Linear SVM, Decision Tree (DT), and Hidden Markov Models (HMM) against the LR-HR DDoS 2024 and the INSDN datasets. The optimized XGBoost model achieved the best performance. However, the analysis was limited to SDN environments and relied on a restricted set of features.

Hirsi et al. [9] use RF as a feature selection model and SVM for classification of SDN traffic. The models were evaluated on a self-generated SDN-DDoS dataset and NSL-KDD, outperforming standalone SVM, NB, FNN, and RF models. However, NSL-KDD is not representative of SDN traffic and is now considered outdated, which limits the applicability of the results.

An unsupervised approach combining blockchain technology, Isolation Forest, and three feature-selection methods was proposed in [10]. It outperforms models that use individual feature-selection techniques. However, evaluation solely on the (outdated) CICDDoS2019 limits its generalizability.

Das et al. [11] evaluated five supervised and five unsupervised algorithms across three datasets. They proposed a hybrid ensemble model combining supervised and unsupervised methods, achieving promising results in both standalone and hybrid configurations. The study is limited by outdated data and an inconsistent feature set across the compared models.

Reference [12] employs K-Means, GMM, and Stochastic Gradient Descent One Class SVM (SGD OCSVM) on the CICDDoS2019 dataset for zero-day attack detection, and Gaussian Random Projection (GRP) for feature selection. The proposed combination demonstrates high accuracy, but it is limited to only a specific and somewhat outdated networking environment.

Hybrid models often combine supervised methods for their high accuracy with unsupervised methods for their ability to detect novel attacks and adapt to unseen patterns. Recent

approaches have used DL models combined with unsupervised models to effectively detect zero-day attacks [13].

It is evident that the literature lacks a thorough analysis of less complex (shallow ML) models across diverse networking conditions, comparing them under the same or very similar assumptions. This work aims to address this gap by benchmarking six representative supervised and anomaly detection approaches. The evaluation was performed across datasets acquired from traditional computer networks, IoT, SDN, and Cloud infrastructure.

III. MATERIALS AND METHODS

Fig. 2 shows the flowchart of the proposed methodology. The acquired datasets were first cleaned and encoded. Hyperparameters for all models were optimized using inner 3-fold cross-validation, and the best settings, selected based on average F1 score, were then applied for training with outer 5-fold cross-validation. Feature selection/extraction and data scaling were performed within each training cross-validation fold to prevent data leakage.

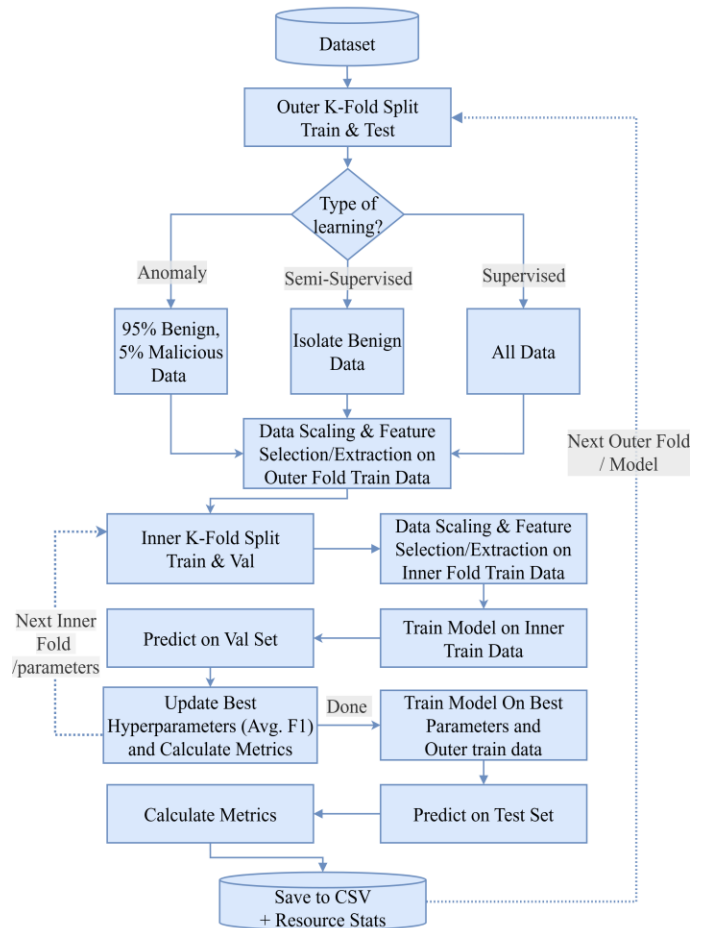


Fig. 2. ML pipeline for DDoS detection.

Preprocessing for different models involved different approaches. The data for supervised models were standardized using the Min-Max statistical method, whereas unsupervised/anomaly models used the Z-Score standardization method.

The dimensionality of the datasets was reduced using Chi-Squared feature selection for supervised models and Principal Component Analysis (PCA) feature extraction for the other machine learning models, excluding the autoencoder model, for which no feature selection or extraction method was used. This approach of combining MinMax with Chi-Squared is essential, as Chi-Squared cannot be computed on negative data, whereas Z-Score can produce negative values but preserves the variance of the features. PCA, on the other hand, is sensitive to scaling and does not require the target feature, making it suitable for unsupervised models. A fixed random seed of 42 was used for sampling data during outer and inner folds, as well as for PCA feature extraction.

To execute the ML pipeline, we used a Hetzner Cloud dedicated server with 4 vCPUs, 16 GB RAM, and 160 GB storage. The 4 vCPUs were based on an AMD EPYC Milan processor with a 64-bit x86_64 architecture. The server ran the

Ubuntu 24.04.4 LTS (Linux kernel 6.8.0-101-generic) operating system. The programming environment and the respective list of libraries are presented in Appendix A (Table IV).

The following subsections provide a detailed description of each step of the DDoS detection model development.

A. Datasets

The datasets were selected based on several criteria. Our goal was to cover diverse network traffic patterns, DDoS attack types, and networking frameworks. Therefore, the datasets include traffic from conventional networks, IoT environments, SDN controllers, and Cloud environments. We also balanced recency with adoption in the literature. However, in this process, some widely used datasets were excluded due to insufficient documentation on the data acquisition setup or outdated content. We finally limited the scope of the materials to the five datasets summarized in Table I.

TABLE I. DATASETS

Dataset	Year	Environment	Records	Features	Key Strength	Main Limitation
ASEADOS SDN IOT [14]	2026	SDN + IoT	457,044	84	Slow-rate DDoS attacks	DDoS attacks mainly with stress test tools
BCCC-cPacket-CloudDDoS-2024 [15]	2024	Cloud	540,494	319	Large feature set	DDoS methods unspecified. Data imbalance
SDN Dataset [16]	2020	IoT + SDN	27.9M / 30.2M	34	Large-scale dataset	Duplicates; undocumented DDoS attack types
TON-IoT [17]	2020	IoT + Cloud + SDN	830,302	47	Traffic diversity	DDoS attacks are unspecified, despite using UFOnet and custom scripts.
INSDN [18]	2020	SDN	343,889	84	SDN-specific and slow rate attacks	Duplication after removing the irrelevant features

B. Data Preprocessing

A data cleaning audit was performed on all datasets. The sensitivity analysis of the deduplication strategy showed a non-significant reduction in precision and recall after data deduplication, except for the dataset obtained from the cloud environment, where a small increase in both metrics was observed. This implies that deduplication is safe in all cases.

Label encoding was applied to the label and protocol columns, assigning a unique integer to each class. To prevent ordinal bias, source and destination ports were transformed into frequency-based and port-group features (Unknown, Well-known, Registered, and Private). The resulting port-group features were then encoded using One-Hot encoding.

To prevent data leakage, standardization and feature selection were performed within each fold of the k-fold cross-validation process, while the validation and test data remained unseen during these steps (Fig. 2). For supervised models, Min-Max normalization was applied, while feature selection was performed using the Chi-square (χ^2) method. For unsupervised methods, Z-Score standardization was performed before PCA feature extraction because PCA is sensitive to the scale and variance of the input features. Feature selection and extraction reduce dataset dimensionality and improve both model efficiency and classification performance.

Min-Max is a data standardization statistical method used for converting data within a range [0,1]. Min-Max normalization is calculated as follows:

$$X_{\text{scaled}} = \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} \quad (1)$$

where:

X is an original value from the dataset.

$X_{\min}$, $X_{\max}$ represent the minimum and maximum values of a given feature, respectively.

Z-score is also a statistical data standardization method used to convert data so that the features contribute equally to the analysis. Z-Score can be described by Eq. (2):

$$Z = \frac{x - \mu}{\sigma} \quad (2)$$

where:

x – observed value

μ – mean

σ – standard deviation

These methods are necessary for model training because features with larger numerical ranges can influence the model and create a biased prediction.

Chi-squared feature selection is a statistical method used to select features that are correlated with the target feature. The Chi-Squared statistic is calculated via the χ^2 value:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \quad (3)$$

where:

O_i – observed frequency

E_i – expected frequency

χ^2 – Chi-Squared statistic

Higher values of χ^2 indicate greater relevance.

Since the Chi-square (χ^2) method requires non-negative feature values, Min–Max normalization is applied beforehand.

The elbow method was used to determine the optimal number of features by identifying the point beyond which additional features provide negligible improvement. The process of identifying the optimal number of features by using the elbow methodology is presented as pseudocode in Algorithm 1.

Principal Component Analysis (PCA) is a statistical method used for feature extraction. It reduces the number of features transforming the feature space into new uncorrelated linear components. This reduces the complexity of the dataset and improves training efficiency.

Algorithm 1: Elbow Method for selecting optimal number of features for χ^2 and PCA

Input:

X_{train} : Training features,
 y_{train} : labels,
model: model type,

Output:

k: Optimal number of features/components,

If the model is supervised, then

 Compute χ^2 score for each feature
 using (X_{train} , y_{train})

 Rank features according to their χ^2 scores

 Compute cumulative feature importance

 Generate cumulative importance curve

$x \leftarrow$ Feature indices,

$y \leftarrow$ Cumulative feature importance

else

 Fit PCA on X_{train}

 Compute explained variance ratio

 Compute cumulative explained variance

 Generate cumulative variance curve

$x \leftarrow$ Component indices,

$y \leftarrow$ Cumulative explained variance

end if

Determine the knee point using KneeLocator with:

 curve $\leftarrow$ "concave"

 direction $\leftarrow$ "increasing"

$k \leftarrow$ KneeLocator(x , y , curve, direction)

return k

Fig. 3 and Fig. 4 show examples of elbow point identification within one fold of the outer 5-fold cross-validation process for χ^2 feature selection and PCA feature extraction, respectively.

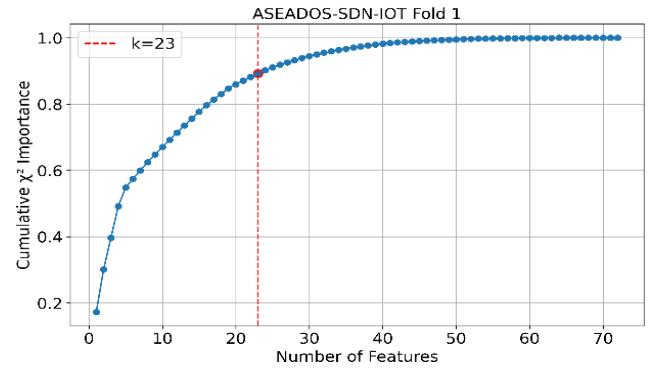


Fig. 3. An Example of the selected number of features for χ^2 feature selection (ASEADOS-SDN-IOT at fold 1).

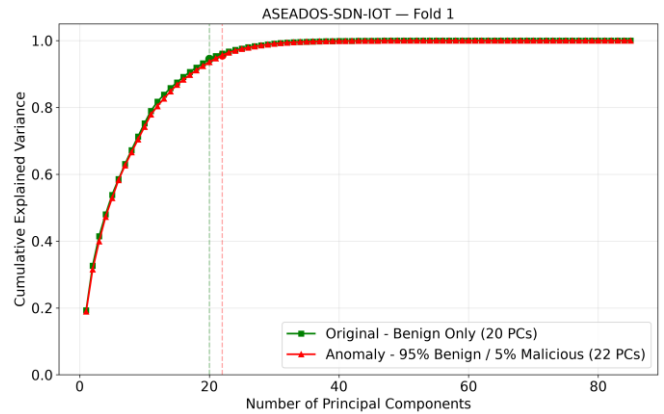


Fig. 4. Example of the selected number of features for PCA feature extraction (ASEADOS-SDN-IOT at fold 1).

C. Model Selection and Implementation

Although DL has become dominant in data science, practical deployments of DDoS detection mechanisms often require a balance between effectiveness and efficiency. Consequently, combined lightweight ML approaches, as well as hybrid approaches that combine DL with traditional ML techniques for DDoS identification, have been extensively explored in the literature. To evaluate the potential of traditional ML techniques in such settings, we selected prominent supervised, unsupervised, and semi-supervised methods that are widely used in DDoS detection. In this subsection, we briefly present their essence.

Supervised ML algorithms operate on labeled data. In our analysis, we included Random Forest (RF) and Extreme Gradient Boosting (XGBoost). These algorithms were selected based on their strong performance in DDoS detection reported in the surveyed literature, outperforming other methods across a wide range of evaluation metrics.

RF randomly divides the feature set into a specific number of decision trees and trains them on different subsets of data. Then, the output is classified through majority voting. The algorithm is robust, resistant to overfitting, and performs well on unbalanced data.

XGBoost uses bootstrapping as an ensemble method, i.e., each decision tree is trained on the residuals of the one before it. This iterative process improves accuracy and enables effective

modeling in a complex feature space. XGBoost is robust to missing values and can handle imbalanced datasets effectively.

Unsupervised ML operates on unlabeled data. In the context of DDoS detection, statistical and unsupervised methods are essential for detecting unseen (zero-day) attacks. Semi-supervised approaches learn on partially labeled benign data, and then detect patterns that deviate from benign behavior.

Based on their results from comparative analyses, we have chosen to examine Local Outlier Factor (LOF) and Isolation Forest (IF) for unsupervised anomaly detection, and autoencoder and SGD-OCSVM for semi-supervised training. Although autoencoders belong to the DL group of algorithms, we have included them in our study because autoencoders are often used for anomaly detection and are therefore of interest for comparison with the other presented anomaly detection algorithms.

LOF is a density-based method that measures the local isolation of data points compared to their neighbors. Specifically, if a data point is surrounded by a lower density compared to the average density of its neighbors, that point is considered an anomaly. This approach is particularly effective in identifying local anomalies.

IF employs a tree-based structure similar to RF. Each tree recursively partitions the data using randomly selected features and split thresholds. The average path length of data points across the trees is evaluated. Since anomalous instances are typically easier to isolate, they tend to have shorter average path lengths than legitimate traffic and are therefore identified as outliers.

An autoencoder is a DL structure typically used for feature extraction and anomaly detection. It consists of an encoder and a decoder. In the context of DDoS detection, the encoder encodes the pattern learned from benign data. Then, the decoder tries to reconstruct the original features from the encoded ones. If the data are malicious, the reconstruction error will be high.

OCSVM is an anomaly detection algorithm based on SVM, except that it is trained exclusively on unlabeled data to find a hyperplane that encloses the majority of benign data. When applied to mixed traffic, anomalies are differentiated as belonging to the other side of the hyperplane. In this work, the surveyed literature motivated us to use SGD OCSVM.

We trained supervised learning algorithms on the mixed benign and malicious data, while semi-supervised models were trained only on benign samples. The autoencoder used a fully connected encoder-decoder architecture with two ReLU-activated hidden layers and a linear output layer. The model was trained using Adam optimization and MSE loss with early stopping, where patience was 5 to prevent overfitting. The autoencoder model creation algorithm is provided in Appendix B (Table V). The anomaly detection models were trained on a subset comprising 95% benign data and 5% malicious data, randomly selected from each training fold. All algorithms were tested on mixed benign and malicious samples. For LOF, IF, and OCSVM, the labels were obtained from their prediction methods, where outliers (-1) were mapped to the malicious class, and inliers (1) were mapped to the benign class.

Autoencoder, on the other hand, identified anomalies using a reconstruction-error threshold tuned to the 95th, 97th, or 99th percentile. Parameters used for model implementation and hyperparameter tuning, along with their selected values for all models, are presented in Appendix C (Table VI).

D. Evaluation Objectives

Our evaluation objectives focused on measuring the models' effectiveness, computational efficiency, generalization, and performance consistency across different network environments. We also analyzed the datasets' content and their discriminatory capacity.

Effectiveness was measured through standard classification metrics, such as accuracy, precision, recall, and F1.

Accuracy measures the proportion of the correctly classified instances out of the total predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

where:

TP – True Positive is the sum of all instances correctly classified as belonging to the positive class.

TN – True Negative is the sum of all instances defined as benign traffic by the model and are actually benign traffic.

FP – False Positive is the sum of all instances defined as DDoS attack traffic by the model, but are actually benign traffic.

FN – False Negative is the sum of all instances defined as benign traffic by the model, but are actually DDoS attack traffic.

Precision measures the ratio of the true positives to the total positives. In our case, it represents how many of the predicted attacks were actual attacks.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

Recall, sometimes referred to as sensitivity, measures the proportion of all the correctly predicted DDoS attacks among all the DDoS attacks in the whole class of DDoS attacks.

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

F1-score is the harmonic mean between precision and recall.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

Computational efficiency was measured through training time, tuning time, CPU utilization, and peak memory requirements. CPU utilization was calculated as the average value across four CPU cores. These measurements were performed on the same platform to allow for comparison under the same assumptions. However, computational efficiency should not be interpreted in terms of absolute values, as it is platform-specific. Therefore, the results serve only for comparative analysis.

Generalization was measured by the ability of the models to classify unseen data in different folds of the cross-validation process.

Performance consistency across different network environments was measured using the standard deviation of models' classification performance across the datasets.

Discriminatory capacity within datasets was measured through mean normalized confusion matrices, which are suitable for evaluating model performance across multiple groups. In this case, we measured the performance of different models applied to each dataset and calculated mean normalized confusion matrices with relative (percentage) values.

IV. RESULTS

We divide this section into two parts. The first part presents the statistical and visual characteristics of the cleaned and preprocessed data. The second part presents the performance of the applied algorithms.

A. Exploratory Data Analysis (EDA)

To ensure consistent assumptions across models, we applied a standardized process of data cleaning and preprocessing.

We first cleaned the dataset by retaining only benign and DDoS traffic, eliminating irrelevant variables (e.g., identifiers), and handling infinite, missing, and invalid values.

The final number of features (including the target feature) and rows, as well as the distribution of benign and DDoS traffic across the analyzed datasets, are presented in Table II.

Table II shows that all datasets are relatively well-balanced. While this is normally desirable, it requires certain changes to the data splitting process during the development of anomaly detection models. In particular, due to the assumption of anomalies being infrequent events, their performance may not be representative if tested on balanced datasets.

We observe that features related to time, packet frequency, and volume follow a right-skewed distribution in each dataset (Fig. 5). Since the flow duration parameter has a different time scale in two of the datasets, its distribution is not shown.

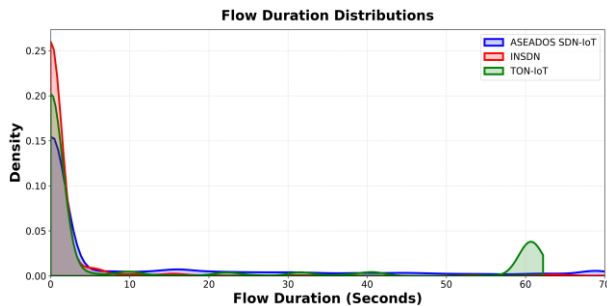


Fig. 5. Flow duration distribution across three datasets.

PCA on datasets with highly correlated columns results in a smaller number of extracted features (Fig. 6). Semi-supervised ML uses only the benign data for training. Therefore, the number of extracted features for semi-supervised learning is naturally lower compared to other unsupervised methods. An exception is the number of extracted features for the Cloud dataset, which is likely due to the diversity of benign traffic and the similarity of benign and malicious traffic. Chi-squared, which is used for supervised ML algorithms, shows the degree to which the features are correlated with the label column.

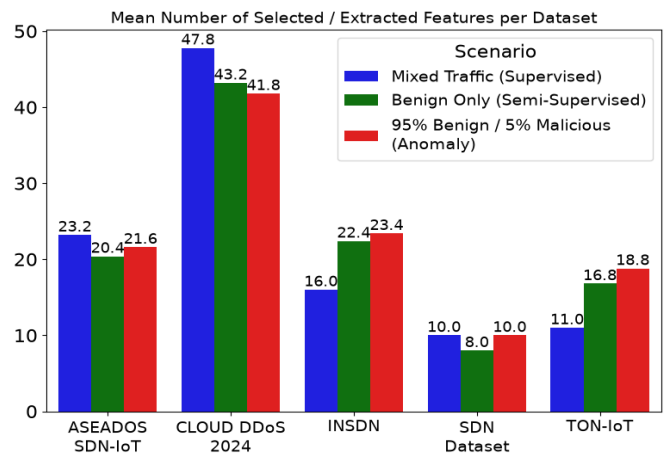


Fig. 6. Mean number of selected/extracted features across datasets.

The most important features as selected by χ^2 are presented in Appendix D (Table VII).

TABLE II. DATASET AFTER CLEANING

Cleaned dataset	Benign	DDoS	Features	Total
ASEADOS SDN IOT	246.071 (58.15%)	177.056 (41.85%)	86	423.127
BCCC-cPacket-CloudDDoS-2024	260.205 (81.45%)	59.246 (18.55%)	300	319.451
SDN dataset	1.029.143 (65.66%)	538.220 (34.34%)	30	1.567.363
TON-IoT	410.476 (50.26%)	406.190 (49.74%)	50	816.666
INSDN	58.758 (53.20%)	51.673 (46.8%)	86	110.431

B. Model Results

Most studies in this field consider accuracy a leading classification performance metric. However, in highly imbalanced learning environments, such as those for anomaly detection, where anomalies are expected to be rare events, relying on accuracy can be misleading. This is due to the fact that accuracy is highly influenced by class distribution and may remain high even when false positives and false negatives are considerable.

To ensure fairness and consistency in our analysis, we use the F1-score as the primary metric for model performance evaluation, as it balances precision and recall. Fig. 7 presents the mean F1 scores and standard deviations of the models on each dataset.

Among the selected models, XGBoost demonstrated the best classification results, whereas SGD-OCSVM performed the worst regarding classification metrics. The small standard deviation observed for all models indicates that the classification results provide a reliable estimate of the expected model performance, suggesting good generalization across different runs and dataset splits.

Table III presents the performance of all models across all five datasets, whereas Fig. 8 illustrates the trade-off between the model classification performance and computational efficiency across all datasets. All values were obtained from an isolated dedicated cloud environment.

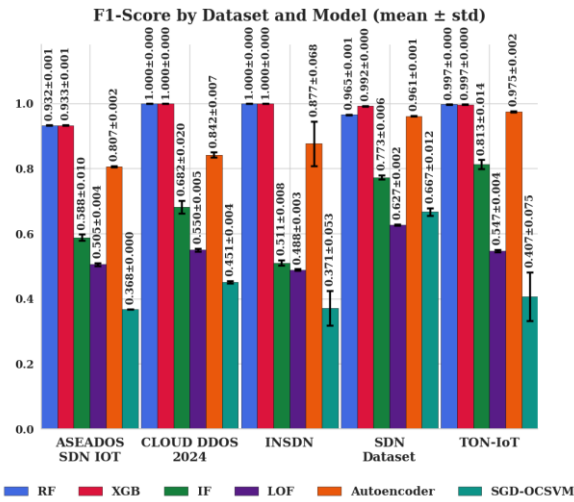


Fig. 7. F1 scores and standard deviations of models on each dataset.

As shown in Fig. 8, XGBoost achieves the best balance between DDoS identification and computational efficiency. Among unsupervised methods, IF stands out by combining solid classification performance with high computational efficiency. The autoencoder demonstrates high classification performance at the cost of higher resource utilization and runtime. SGD-OCSVM outperforms all other methods in terms of runtime. However, it ranks last in DDoS identification performance across the selected datasets and consumes considerable CPU resources.

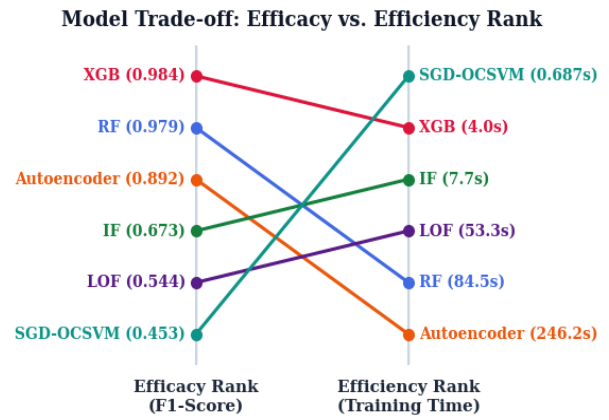


Fig. 8. Efficacy (F1) vs Efficiency (Training Time) trade-off.

Except in the case of SGD-OCSVM, where accuracy and F1 vary significantly, the low standard deviation in classification for all other models shows that they maintain high performance consistency across different networking setups. Higher variability in computational metrics across the datasets is expected, since computational efficiency highly depends on the feature space.

To assess the discriminative power of each dataset, we analyzed the mean normalized confusion matrices across datasets (Fig. 9). The TON-IoT dataset was the most discriminative compared with the other datasets, whereas ASEADOS-SDN-IoT showed less discriminative power.

TABLE III. AVERAGE CLASSIFICATION PERFORMANCE AND EFFICIENCY OF THE MODELS ACROSS FIVE DATASETS

Model/Metric	RF	XGB	IF	LOF	Autoencoder	SGD-OCSVM
Accuracy	0.980 ± 0.029	0.985 ± 0.029	0.938 ± 0.024	0.913 ± 0.010	0.911 ± 0.062	0.649 ± 0.132
Precision	0.981 ± 0.030	0.984 ± 0.030	0.673 ± 0.126	0.543 ± 0.053	0.912 ± 0.057	0.564 ± 0.238
Recall	0.978 ± 0.029	0.985 ± 0.026	0.673 ± 0.126	0.544 ± 0.054	0.885 ± 0.078	0.542 ± 0.068
F1-Score	0.979 ± 0.030	0.984 ± 0.029	0.673 ± 0.126	0.544 ± 0.054	0.892 ± 0.074	0.453 ± 0.125
Training Time (s)	84.493 ± 120.613	3.997 ± 3.042	7.696 ± 5.757	53.291 ± 40.846	246.215 ± 274.419	0.687 ± 0.466
Tuning Time (s)	5860.820 ± 7326.411	668.243 ± 373.508	175.759 ± 135.280	1081.502 ± 984.672	18138.425 ± 12948.144	16.450 ± 12.545
Training CPU (%)	25.266 ± 0.407	28.610 ± 3.922	24.998 ± 0.016	42.853 ± 10.265	34.579 ± 3.347	49.402 ± 25.011
Tuning CPU (%)	26.018 ± 0.931	32.591 ± 6.241	25.000 ± 0.000	31.620 ± 4.687	33.553 ± 2.751	66.481 ± 19.096
Training Memory Peak (MB)	69.836 ± 46.018	3.956 ± 3.507	37.491 ± 26.140	347.488 ± 415.666	659.358 ± 593.578	46.669 ± 25.931
Tuning Memory Peak (MB)	1041.916 ± 646.464	1003.888 ± 668.147	598.580 ± 504.337	721.308 ± 532.039	1349.199 ± 843.267	1163.579 ± 857.727

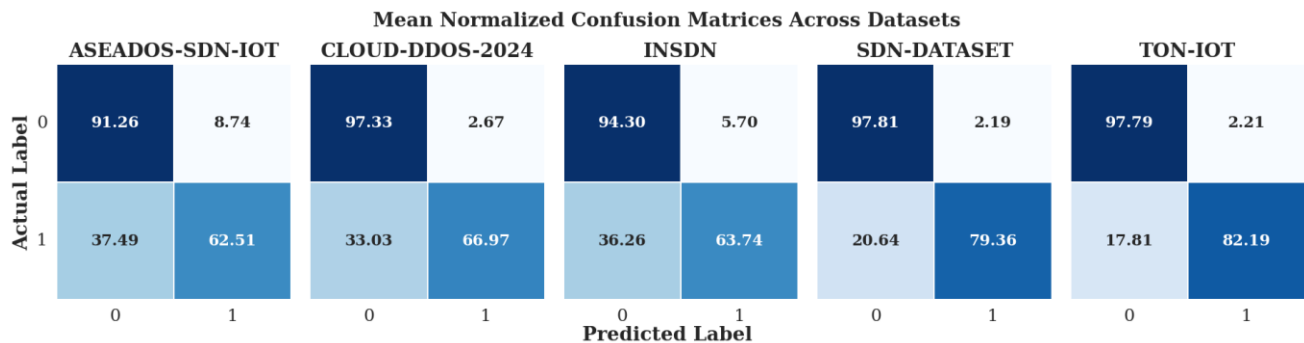


Fig. 9. Mean normalized confusion matrices for all models across each dataset.

V. DISCUSSION

We analyzed and compared the performance of several ML approaches across five different datasets for DDoS detection. While our focus was on providing a clearer assessment of the effectiveness, efficiency, generalization, and performance consistency of shallow ML methods for different network environments, we also provided insight into the datasets used for model training.

The EDA confirmed that packet timing, frequency, and packet length metrics follow a right-skewed distribution across all datasets. We noticed shifted peak positions in the timing statistics derived from datasets containing IoT traffic compared with those from other environments. The results in Fig. 9 also indicate that SDN and TON-IoT datasets contain more discriminative features than the other datasets. The highest false-negative rates are evident in the ASEADOS-SDN-IOT dataset (37.49%) and INSDN (26.26%). This is likely due to the higher presence of slow-rate attacks in these datasets. The normalized values for the Cloud dataset also show a relatively high false-negative rate (33.03%). This is likely due to the “ambiguity” introduced by traffic labeled as “Suspicious”. A more thorough analysis of datasets remains a subject for future work.

In terms of effectiveness and computational efficiency, the results indicate that XGBoost is the most balanced supervised method ($F1=0.984$, Training Time=4s) for DDoS detection, whereas IF performs best ($F1=0.673$, Training Time=7.7s) among the evaluated unsupervised methods (Fig. 8). This result is in line with the existing literature and confirms the potential of IF for integration into defense systems in IoT environments. As IF primarily focuses on global anomalies, LOF may be used as a complementary technique to detect local anomalies (e.g., slow rate attacks).

SGD-OCSVM achieves a short runtime at the cost of relatively high CPU utilization. Autoencoders demonstrate high anomaly detection capabilities and are highly useful in many IDS systems, but their computational requirements make them less applicable to resource-constrained devices.

All models demonstrated high generalization capabilities. Apart from the $F1 (0.453 \pm 0.125)$ and accuracy (0.649 ± 0.132) scores of the OCSVM, the low standard deviations of all other classification measures indicate that the models also exhibit high performance consistency across diverse networking scenarios.

The performed analysis underscores the need for combined approaches while contributing to the selection of their components.

VI. CONCLUSION AND FUTURE WORK

As Internet technologies become more diverse, the complexity of DDoS attacks increases, and the need for adaptive intelligent DDoS detection systems becomes more apparent. DL-based methods are superior in many tasks, but they are computationally complex for resource-constrained devices (such as IoT devices and many SDN controllers).

In this research, we developed a benchmarking framework to evaluate six prominent supervised, unsupervised, and semi-supervised learning algorithms for DDoS detection across five datasets acquired from different networking environments.

The comparative analysis highlights the capacity of simple models to balance computational efficiency and DDoS detection performance while maintaining strong generalization and consistency across the diverse network environments, emphasizing their potential for integration into hybrid systems.

Future research will use these findings to examine the adaptability of resource-efficient hybrid models to concept drift, data drift, and zero-day attacks.

REFERENCES

- [1] H. Olimboyev, S. Khamrayeva, H. Olimboyev, O. Khalmuratov, K. Rakhimboev and B. Ibragimov, "Using Deep Learning to Detect DDoS Attacks at the Application Layer," 2025 IEEE 26th International Conference of Young Professionals in Electron Devices and Materials (EDM), Altai, Russian Federation, 2025, pp. 2010-2016, doi: 10.1109/EDM65517.2025.11096735.
- [2] Ö. Tonkal and J. A. Mgungile, "Evaluating Machine Learning Models for DDoS Detection in SDNs," J. Adv. Inf. Technol., vol. 17, no. 3, pp. 596–610, 2026, doi: 10.12720/jait.17.3.596-610.
- [3] J. Pebrianto and V. Suryani, "Adaptive DDoS Attack Detection: Entropy-Based Model With Dynamic Threshold and Suspicious IP Reevaluation," IEEE Access, vol. 13, pp. 55858–55876, 2025, doi: 10.1109/ACCESS.2025.3553144.
- [4] E. Cambiaso, M. Aiello, M. Mongelli, and I. Vaccari, "Detection and Classification of Slow DoS Attacks Targeting Network Servers," Proc. of 15th International Conf. on Availability, Reliability and Security, Virtual event, Ireland, August, 2020, doi: 10.1145/3407023.3409198.
- [5] Z. Tafa & V. Milutinovic, "Machine Learning in Congestion Control: A Survey on Selected Algorithms and a Roadmap to Their Implementation," arXiv:2112.15522. 2021.
- [6] O. Ebrahim, S. Dowaji, and S. Alhammoud, "On the Limited Generalizability of a Universal Features Set for DDoS Detection Across Network Environments," IEEE Access, vol. 14, pp. 7932–7974, 2026, doi: 10.1109/ACCESS.2026.3653648.
- [7] F. Luo, S. Fan, and G. Shao, "A Survey of Emerging DDoS Threats in New Power Systems," Sensors, vol. 26, no. 4, p. 1097, Jan. 2026, doi: 10.3390/s26041097.
- [8] M. M. Trigui, M. S. Islam, M. D. Gambo, M. S. Alam, and T. Helmy, "Resource-Aware Ensemble Framework for DDoS Detection in SDN via Novel Game-Theoretic Feature Selection," IEEE Access, vol. 14, pp. 1666–1676, 2026, doi: 10.1109/ACCESS.2025.3618519.
- [9] A. Hirsi et al., "HSF: A Hybrid SVM-RF Machine Learning Framework for Dual-Plane DDoS Detection and Mitigation in Software-Defined Networks," IEEE Access, vol. 13, pp. 112303–112323, 2025, doi: 10.1109/ACCESS.2025.3583712.
- [10] S. Peng, J. Tian, X. Zheng, S. Chen, and Z. Shu, "DDoS Defense Strategy Based on Blockchain and Unsupervised Learning Techniques in SDN," Future Internet, vol. 17, no. 8, p. 367, Aug. 2025, doi: 10.3390/fi17080367.
- [11] S. Das, M. Ashrafuzzaman, F. T. Sheldon, and S. Shiva, "Ensembling Supervised and Unsupervised Machine Learning Algorithms for Detecting Distributed Denial of Service Attacks," Algorithms, vol. 17, no. 3, p. 99, Mar. 2024, doi: 10.3390/a17030099.
- [12] M. Roopak, S. Parkinson, G. Y. Tian, Y. Ran, S. Khan, and B. Chandrasekaran, "An Unsupervised Approach for the Detection of Zero-Day DDoS Attacks in IoT Networks," Authorea, vol. 2024, no. 0114, doi: 10.22541/au.170526630.07302484/v1.
- [13] N. U. Ain, M. Sardaraz, M. Tahir, M. W. Abo Elsoud, and A. Alourani, "Securing IoT Networks Against DDoS Attacks: A Hybrid Deep Learning Approach," Sensors, vol. 25, no. 5, p. 1346, Jan. 2025, doi: 10.3390/s25051346.
- [14] T. Lakshan Yasarathna and N.-A. Le-Khac, "ASEADOS-SDN-IoT: A novel SDN-IoT network intrusion detection dataset and framework," Internet Things, vol. 36, p. 101891, Mar. 2026, doi: 10.1016/j.iot.2026.101891.
- [15] M. Shafi, A. H. Lashkari, V. Rodriguez, and R. Nevo, "Toward Generating a New Cloud-Based Distributed Denial of Service (DDoS)

Dataset and Cloud Intrusion Traffic Characterization,” Information, vol. 15, no. 4, p. 195, Apr. 2024, doi: 10.3390/info15040195.

- [16] A. Kaan Sarica and P. Angin, “A Novel SDN Dataset for Intrusion Detection in IoT Networks,” in 2020 16th International Conference on Network and Service Management (CNSM), Izmir, Turkey: IEEE, Nov. 2020, pp. 1–5. doi: 10.23919/CNSM50824.2020.9269042.
- [17] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, “TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems,” IEEE Access, vol. 8, pp. 165130–165150, 2020, doi: 10.1109/ACCESS.2020.3022862.
- [18] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, “InSDN: A Novel SDN Intrusion Dataset,” IEEE Access, vol. 8, pp. 165263–165284, 2020, doi: 10.1109/ACCESS.2020.3022633.

APPENDICES

TABLE IV. APPENDIX A: LIBRARY VERSIONS

Library	Version	Usage
Python	3.12.3 with GCC 13.3.0	Experimental environment
Pandas	2.3.3	Data preprocessing and analyses
NumPy	2.4.2	
matplotlib	3.10.8	Graph generation
seaborn	0.13.2	
scikit-learn	1.8.0	ML models, preprocessing, and model evaluation metrics
XGBoost	3.3.0	XGBoost model building
TensorFlow	2.21.0 (Keras API)	To build Autoencoder model
PyTorch	2.13.0	
kneed	0.8.6	For knee-point detection
psutil	7.2.2	Resource utilization monitoring.

TABLE V. APPENDIX B: AUTOENCODER ARCHITECTURE AND OPTIMIZATION

ALGORITHM: Autoencoder model creation
Input: input_dim, encoding_dim inputs $\leftarrow$ Input(input_dim) encoded_1 $\leftarrow$ Dense(encoding_dim, ReLU)(inputs) encoded_2 $\leftarrow$ Dense(encoding_dim / 2, ReLU)(encoded_1) decoded_1 $\leftarrow$ Dense(encoding_dim, ReLU)(encoded_2) outputs $\leftarrow$ Dense(input_dim, Linear)(decoded_1) Autoencoder $\leftarrow$ Model(inputs, outputs) Optimizer $\leftarrow$ Adam Loss $\leftarrow$ Mean Squared Error Compile Autoencoder using Optimizer and Loss Return Autoencoder

TABLE VI. APPENDIX C: HYPERPARAMETERS USED FOR HYPERPARAMETER TUNING

Model	Hyperparameter	Values Tested
Random Forest	n_estimators	100, 200
	max_depth	5, 10, 15
	min_samples_split	2, 5, 10
	min_samples_leaf	1, 2, 4
	max_features	sqrt
	random_state	42

XGBoost	n_jobs	1
	n_estimators	100, 200
	max_depth	3, 6, 10
	learning_rate	0.01, 0.1, 0.2
	subsample	0.8, 1.0
	colsample_bytree	0.8, 1.0
	objective	binary
	eval_metric	logloss
	tree_method	hist
	random_state	42
Local Outlier Factor	n_jobs	1
	n_neighbors	10, 20, 30
	algorithm	auto,ball_tree, kd_tree
	Contamination	0.05
Isolation Forest	novelty	True
	n_estimators	100, 200, 300
	max_samples	auto, 0.8, 0.5
	contamination	0.05
SGD One-Class SVM	random_state	42
	n_jobs	1
	eta0	0.001, 0.01
	nu	0.05, 0.1, 0.2
	Learning_rate	optimal
Autoencoder	average	True
	random_state	42
	encoding_dim	8, 12,16, 24, 32, 48
	epochs	30, 50
	batch_size	128, 256
	threshold_percentile	95, 97, 99

TABLE VII. APPENDIX D: FEATURES SELECTED ACROSS 5 FOLDS

Dataset	Nr. of Features	Selected features
CLOUD-DDOS-2024	47	max_header_bytes, min_header_bytes, mean_header_bytes, median_header_bytes, mode_header_bytes, bwd_std_header_bytes, bwd_cov_header_bytes, bwd_init_win_bytes, syn_flag_percentage_in_total, fwd_syn_flag_percentage_in_total, fwd_ack_flag_percentage_in_total, bwd_syn_flag_percentage_in_total, fwd_syn_flag_percentage_in_fwd_packets, fwd_ack_flag_percentage_in_fwd_packets, bwd_syn_flag_percentage_in_bwd_packets, bwd_ack_flag_percentage_in_bwd_packets, bwd_rst_flag_percentage_in_bwd_packets, packets_IAT_mean, packet_IAT_max, packet_IAT_min, packet_IAT_total, packets_IAT_median, packets_IAT_cov, packets_IAT_mode, fwd_packets_IAT_cov, bwd_packets_IAT_mean, fwd_packets_IAT_cov, bwd_packets_IAT_max, bwd_packets_IAT_min,,

		bwd_packets_IAT_total, bwd_packets_IAT_median, bwd_packets_IAT_mode, handshake_state, min_bwd_packets_delta_time, max_bwd_packets_delta_time, mean_packets_delta_time, mode_packets_delta_time, median_packets_delta_time, cov_packets_delta_time, mean_bwd_packets_delta_time, mode_bwd_packets_delta_time, median_bwd_packets_delta_time, cov_fwd_packets_delta_time, std_header_bytes_delta_len, activity, Src_Port_Freq, Src_Port_Group_Private, Src_Port_Group_Registered
TON-IOT	11	ts, conn_state, dns_qtype, dns_AA, dns_RA, Src_Port_Freq, Dst_Port_Freq, Src_Port_Group_Well-known, Dst_Port_Group_Private, Dst_Port_Group_Registered, Dst_Port_Group_Well-known

SDN-DATASET	10	Pkts, Bytes, Srate, Drate, TnP_PSrcIP, TnP_PDstIP, TnP_PerProto, TnP_Per_Dport, N_IN_Conn_P_DstIP, N_IN_Conn_P_SrcIP
ASEADOS-SDN-IOT	23	Flow Duration, Fwd Pkt Len Max, Flow IAT Max, Fwd IAT Tot, Fwd IAT Std, Fwd IAT Max, Bwd IAT Tot, Bwd IAT Std, Bwd IAT Max, Bwd PSH Flags, FIN Flag Cnt, SYN Flag Cnt, RST Flag Cnt, PSH Flag Cnt, Idle Mean, Idle Max, Idle Min, Src_Port_Freq, Src_Port_Group_Unknown, Dst_Port_Group_Private, Dst_Port_Group_Registered, Dst_Port_Group_Unknown, Dst_Port_Group_Well-known
INSDN	16	Protocol, Flow Duration, Bwd Pkt Len Min, Fwd IAT Tot, Fwd IAT Std, Fwd IAT Max, Bwd IAT Tot, Pkt Len Min, FIN Flag Cnt, SYN Flag Cnt, ACK Flag Cnt, Src_Port_Freq, Dst_Port_Freq, Src_Port_Group_Registered, Src_Port_Group_Well-known, Dst_Port_Group_Private