Evaluating ML-based DDoS Detection with Grid Search Hyperparameter Optimization

Odnan Ref Sanchez*, Matteo Repetto†, Alessandro Carrega*, and Raffaele Bolla*‡
*CNIT S2N National Laboratory, Genoa, Italy
Email: {odnan.sanchez, alessandro.carrega}@cnit.it
†IMATI - CNR, Genoa, Italy
Email: matteo.repetto@ge.imati.cnr.it
‡DITEN, University of Genoa, Italy
Email: raffaele.bolla@unige.it

2021 IEEE 7th International Conference on Network Softwarization (NetSoft) | 978-1-6654-0522-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/NetSoft51509.2021.9492633

***Abstract*—Distributed Denial of Service (DDoS) attacks disrupt global network services by mainly overwhelming the victim host with requests originating from multiple traffic sources. DDoS attacks are currently on the rise due to the ease of execution and rental of distributed architectures such as the Internet of Things (IoT) and cloud infrastructures, which could potentially result in substantial revenue losses. Therefore, the detection and prevention of DDoS attacks are currently topics of high interest. In this study, we use traffic flow information to determine if a specific flow is associated with a DDoS attack. We used traditional Machine Learning (ML) methods in developing our DDoS detector and applied an exhaustive hyperparameter search to optimize their detection capability. Using lightweight approaches is suitable for resource-constrained environments such as IoT to reduce computing overhead. Our evaluation shows that most algorithms provide satisfactory results, with Random Forests achieving as high as 99% of detection accuracy, which is similar to the performance of current deep learning solutions for DDoS detection.**

***Index Terms*—DDoS Detection, Machine Learning, Network Security**

I. Introduction

Denial of Service (DoS) is a well-known cyberattack that targets a victim host (e.g., network servers, resources, or nearby infrastructures) mainly through excessive flooding of network requests to overload the victim that becomes unable to execute its usual services [1]. A DoS attack becomes more difficult to detect when the source is distributed over the network, which is known as Distributed DoS (DDoS). Over the years, DDoS attacks have grown due to the increase of distributed architectures such as the Internet of Things (IoT), distributed service paradigms (e.g., [2]), and the ease of renting resources. For instance, the MIRAI Botnet [3] was recently used to deploy a large-scale DDoS attack that has infected around 600,000 IoT devices worldwide.

With the growing sophistication of cyberattacks such as DDoS, traditional monitoring tools become inadequate to meet the required detection and mitigation strategies to protect critical network infrastructures. In this respect, DDoS detection based on Machine Learning (ML) (e.g., [4]) has become popular due to the increased detection performance with today's availability of data and computing resources.

Recent studies are leveraging Deep Learning (DL) solutions (e.g., [5], [6]) to further increase the detection performance. On the other hand, DL methods are usually more strenuous to deploy in real-world scenarios [7] and typically require much more input data and computational power than traditional ML methods. It becomes a concern in lightweight execution environments such as IoT. Thus, we aim to create a light ML-based DDoS detector using traditional ML methods and utilize traffic flow information as input data streams.

In developing our detector, we used grid search for finding the best hyperparameters to enhance the accuracy, which is a novelty compared to existing ML-based DDoS detectors. A similar approach has been used in [8] to improve accuracy but for BGP anomalies. While current trends focus on DL methods (e.g., [6]), this study looks into improving the capabilities of lightweight ML methods to their full potential using hyperparameter optimization. We evaluate classification techniques such as Naive Bayes, Logistic Regression, Decision Trees, Random Forests, K-Nearest Neighbors, Support Vector Machines, and Neural Networks to represent our detector.

We trained and evaluated the ML methods using the Canadian Institute of Cybersecurity (CIC) datasets[1]. These datasets are used for both research (e.g., [6], [9]) and industrial purposes. We utilized datasets from 2012, 2017, 2018, and 2019 that include DoS and DDoS attacks. This data includes, among others, the number of incoming and outgoing packets in a flow, packet inter-arrival times, header flag counts, and so on. The statistics include the ratio, total, mean, minimum, and maximum of every flow feature aforementioned.

Our results show that DDoS attacks can be detected using traditional ML algorithms with optimized hyperparameters, reaching an accuracy of over 98% by using Random Forests and Decision Trees across all the datasets. These results are similar to current DL approaches. Thus, tuning hyperparameter in traditional ML allows for increased performance similar to DL approaches with fewer resources needed.

The remainder of the paper is organized as follows. The next section, Section II, discusses related work on the current

[1]The details on the CIC datasets can be found on the website: https://www.unb.ca/cic/datasets/index.html

ML techniques for DDoS detection. Then, Section III presents the methodology and experimental setup. Section IV provides a detailed data analysis of the data features used in this study. Section V presents the paper evaluation results and the concluding remarks are discussed in Section VI.

II. Related Work

Early comparison of traditional ML methods for DDoS detection is conducted by Bhamare et al. [4]. Using the UNSW dataset, Logistic Regression achieved best in terms of accuracy (i.e., 89%) with a 97% True Positive rate. Additionally, when these trained models were tested on another dataset (ISOT dataset), both J48 Decision Tree and Logistic Regression have achieved the best with 95% accuracy but had significantly reduced the True Positive rate.

He et al. [10] also studied DoS attacks but focused on the scenario in which the cloud environment is used for launching the attacks. They used hypervisors/virtual machine information and the results show that Support Vector Machines with a linear kernel performed best with 99.73% accuracy.

Given the increasing number of cyber attack types, Salman et al. [11] propose a two-step approach to identify the types of attack. First, they used ML to detect anomalies and proceed to a rule-based identification process to determine the attack type. Using Random Forest and Linear Regression, they achieved 99% of detecting anomalies while they achieved 93.6% accuracy in classifying the attack type.

With the increasing popularity of Neural Network (NN), Yuan et al. [9] adopted different DL methods such as Convolutional NN (CNN) and Recurrent NN (RNN) for DDoS attack detection called DeepDefense. They evaluated their classifier using the ISCXIDS (2012) [12] and showed a substantial reduction of error from using traditional approaches. For instance, the 7.5% error rate from Random Forests was reduced to 2.1% using DL methods.

Furthermore, Yao et al. [13] developed a DL feature extractor, DeepGFL, to classify attack and traffic flow through graph representation. It aims to detect various network attack types. For DoS Hulk, it reached 94.05% of F1-measure in the evaluation using CIC-IDS (2017) [14].

Doriguzzi-Corin et al. [6] also proposed LUCID DDoS detector, which is based on CNN. Using the ISCXIDS (2012) [12], CIC-IDS (2017) [14], and CSE-CIC-IDS (2018) [14] datasets from the CIC, they achieved high detection rates of up to 99.87% accuracy. Similarly, Roopak et al. [15] utilized CNN + Long Short-Term Memory (LSTM) for the detection of DDoS in IoT systems, reaching 97.16% accuracy tested on the CIC-IDS (2017) dataset.

Min et al. [16] also developed a Text-CNN and Random Forest-based Intrusion Detection System (TR-IDS) for IoT DDoS detection. Word embedding and Text-CNN are used for feature extraction of network traffic, which is fed to Random Forests for the final classification. It achieved 99.13% accuracy using the ISCXIDS (2012) [12] dataset.

Finally, Sanchez et al. introduce DLDDoS [5], which utilizes a 2-hidden layer feed-forward NN approach coupled with feature selection to reduce input data. The study used the most recent CIC datasets and achieved high detection performance.

TABLE I: The attack samples taken from CIC Datasets

Date	Duration	Event	attack type	attacks	benign
15/06/2010	60 mins.	DDoS	IRC Botnet	34760	34760
05/07/2017	17 mins.	DoS	Hulk	231073	440031
05/07/2017	13 mins.	DoS	GoldenEye	10293	-
05/07/2017	23 mins.	DoS	slowloris	5796	-
05/07/2017	21 mins.	DoS	SlowHTTPtest	5499	-
07/07/2017	20 mins.	DDoS	LOIC (TCP)	128025	97686
15/02/2018	43 mins.	DoS	GoldenEye	8851	2334
15/02/2018	41 mins.	DoS	Slowloris	2417	-
16/02/2018	34 mins.	DoS	Hulk	30287	97040
16/02/2018	56 mins.	DoS	SlowHTTPTest	30391	-
20/02/2018	65 mins.	DDoS	LOIC (HTTP)	125130	124914
21/02/2018	60 mins.	DDoS	HOIC (TCP)	400	77876
21/02/2018	34 mins.	DDoS	LOIC (UDP)	360	-
12/01/2019	13 mins.	DDoS	DNS	27065	2690
12/01/2019	10 mins.	DDoS	LDAP	12798	1280
12/01/2019	9 mins.	DDoS	MSSQL	15981	1573
12/01/2019	10 mins.	DDoS	NTP	115149	11454
12/01/2019	10 mins.	DDoS	NetBIOS	13530	1374
12/01/2019	11 mins.	DDoS	SNMP	12072	1198
12/01/2019	10 mins.	DDoS	SSDP	6016	615
12/01/2019	24 mins.	DDoS	UDP	17085	1702
12/01/2019	5 mins.	DDoS	TCP-SYN	3025	320
12/01/2019	220 mins.	DDoS	TFTP	202160	20250
12/01/2019	4 mins.	DDoS	UDP-lag	29635	2989
12/01/2019	11 mins.	DDoS	WebDDoS	39	-

The literature clearly shows that ML approaches do not implement hyperparameter optimization and recent works adopt DL models instead. This transition has increased the detection performance. However, DL methods are usually difficult to deploy in real-world scenarios [7] and typically require much more input data and computing capability than the traditional ML methods. This becomes a bottleneck for lightweight execution environments (e.g., IoT) and in centralized security architectures (e.g., [2]) that requires lightweight data collecting agents.

In this work, different from the DL trends, we focus on lightweight and traditional ML models and utilize hyperparameter search to obtain their optimum parameters that yield higher detection performance. This work also uses recent CIC datasets and reports a comparison of current DL approaches in detecting DDoS attacks.

III. System Model

In this section, the datasets and ML models in this study are discussed, followed by the training and evaluation procedures.

A. Datasets

The datasets used in this study are from the CIC, University of New Brunswick (UNB), which have been used in numerous studies (e.g., [5], [6]). We used the latest datasets that contain both DoS and DDoS attacks, namely, ISCXIDS (2012) [12], CICIDS (2017) [14], CSE-CIC-IDS (2018) [14], and CICDDoS (2019) [17], which are reported in Table I. The same set of datasets has also been used in [5]. The attacks extracted are composed of a Botnet attack, DoS attacks such as Hulk, GoldenEye, sloworis and slowHTTPtest, network stress testing tools such as Low Orbit Ion Cannon (LOIC), and DDoS

attacks from utilizing different protocols and applications (i.e., DNS, LDAP, MSSQL, NTP, NetBIOS, SNMP, SSDP, UDP, TCP (Syn), TFTP, UDP-lag, and WebDDoS).

Table I shows that ISCXIDS (2012) includes an Internet Relay Chat (IRC) Botnet DDoS attack [12]. This dataset was generated using the IBM QRadar appliance. Also, multiple DoS and DDoS attacks from the CICIDS (2017), CSE-CIC-IDS (2018), and CICDDoS (2019) include flow features, which are generated by the CICflowmeter tool [18]. Given the tool differences, the real data traces in PCAP format of the ISCXIDS (2012) dataset was used to generate flow features using the CICflowmeter tool to have similar data features across all datasets.

Each data sample contains information about traffic flow statistics. The complete description of these features is provided in the documentation of the CICflowmeter [18]. The 76 features include the header information, number of incoming and outgoing packets, Inter-Arrival Times (IAT) of these packets, packet length information, and so on. The statistics include the ratio, total, mean, minimum, and maximum of every flow information. The use of flow statistics for DDoS detection means that it has to wait for the flow to finish before the classification. However, a DDoS attack is composed of numerous flows. For instance, the 2012 botnet attack produced 34,000 attack flows for a 60-minute duration. In this case, the detector can evaluate flows that have been completed from the first few seconds and can still provide an early alarm trigger.

B. Training Configuration

Each dataset in Table I is randomly distributed into 75% training and 25% test sets. Furthermore, 10% of the training samples are used as the validation set during the training phase. The table shows that only the 2019 dataset has a data imbalance, where there were more attack samples than benign samples. We preprocessed the training data by removing noise and scaling the features. We used the z-score normalization [19] for the feature scaling to have similar ranges for all features using the scikit-learn's standard scaler function[2]. It transforms individual feature mean to zero and unit variance.

We compared multiple classification techniques, namely, Logistic Regression (LR), Naive Bayes (NB) classifier, Decision Trees (DT), Random Forests (RF), K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Multi-layer Perceptron (MLP). The output binary classes include anomaly and benign, which stands for the DDoS attack and regular traffic flows, respectively.

Each of the ML algorithms has its own set of hyperparameters that needs to be fixed before training except for the NB classifier, which does not have a tuning parameter and will be used as a benchmark. These hyperparameters are reported in Table II. For LR, we explore Ridge (L1) and Lasso (L2) regression [20] for regularization, which prevents overfitting and reduces model complexity. The C parameter [21], which is the inverse of regularization strength, is also explored for LR. Similarly, we also explore the C parameter for the SVM.

TABLE II: GridSearchCV hyperparameter values

Algorithm	Hyperparameters
NB	None
LR	penalty: [Ridge (L1), Lasso (L2)], C : [0.001, 0.01, 0.1, 1, 10, 100]
KNN	K: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100], metric: [euclidean, manhattan, chebyshev, minkowski]
DT	impurity :[gini, entropy], max depth: [1:20]
RF	impurity :[gini, entropy], max depth: [1:20], no. of trees: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
SVM	C: [0.001, 0.01, 0.1, 1, 10, 100]
MLP	activation: [logistic, relu, tanh], solver: [lbfgs, adam , sgd], hidden layers size : [20, 40, 60, 80, 100]

For KNN, we explore different values of K neighbors, together with the different metrics for computing the distances among samples. For DT, we use different criteria for computing the impurity such as Gini impurity [22] and entropy [22], with tree depth up to 20 levels. RF also takes the same hyperparameters with the addition of the number of tree estimators ranging from 10 to 100. We explore different sizes of hidden layers for MLP together with their activation functions such as logistic [23], rectified linear unit (relu) [23], and hyperbolic tangent (tanh) [23]. We also explore different weight optimizers, which include the Limited memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithm [24], Stochastic Gradient Descent (SGD) [24], and Adam optimizer [25]. The hyperparameter search is conducted using scikit-learn's GridsearchCV function[3]. The model training and validation are implemented using 3-fold cross-validation.

C. Test Evaluation

In addition to Accuracy (Ac), the evaluation measures also include the F1-score (F1), which is a more appropriate measure for imbalanced datasets such as the 2019 dataset since it is derived from Precision (Pr) and Recall (Rc). The formulas for the evaluation measures are the following:

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

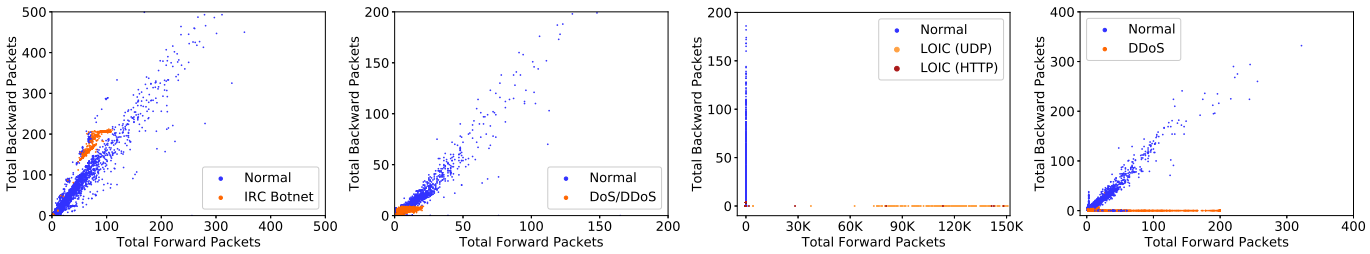
$$Precision\ (Pr) = \frac{TP}{TP + FP} \tag{2}$$

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

$$F1 - score\ (F1) = \frac{2 * PR * RC}{PR\ +\ RC} \tag{4}$$

[2]StandardScaler function: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[3]GridSearchCV function https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html



(a) DDoS IRC Botnet (2012) (b) DoS + DDoS LOIC TCP (2017) (c) LOIC UDP and HTTP (2018) (d) DDoS attacks (2019)

Fig. 1: The number of forward versus backward packets of regular and attacks flows

In the equations above, the True Positives (TP) and True Negatives (TN) are the numbers of traffic flows correctly classified as DDoS attacks and regular flows, respectively. Otherwise, incorrectly classified traffic flows are counted as False Positives (FP) and False Negatives (FN) for DDoS attacks and regular flows, respectively.

Each ML model is also evaluated using the Receiver Operating Characteristic (ROC) curve. This curve graphically illustrates the classification capability of the ML model by plotting the TP and FP rates for various thresholds. The Area Under the ROC Curve (AUC) ranges from zero to one (i.e., $\in [0, 1]$), in which larger values signify better classification capability. These metrics are mostly used in evaluating binary classification approaches (e.g., [5], [8]).

IV. Data Analysis

DDoS attacks are known to overwhelm the server victims with both incoming traffic (e.g., volumetric attacks) and outgoing traffic (e.g., reflection attacks) that are more than they can handle. As a first step, we investigated the total packet flows for both forward (attacker to the victim) and backward (victim to the attacker) directions. These features are included among the multiple features used in this study.

Figure 1 shows the total forward versus backward packets in a flow differentiated according to regular and DDoS flows. Even only showing these two features, we can see a pattern for the different DDoS attacks. Normal traffic mostly has an equal number of forward and backward traffic, even for Figure 1c, where we skewed the figure to show a large difference between incoming and outgoing traffic.

Figure 1a shows the DDoS Botnet from 2012, which has a slightly increased number of backward packets versus the forward packets. In this scenario, seven users managed to infiltrate the servers and force them to download and run the HTTP GET command, which resulted in full and partial inaccessibility [12].

For DoS attacks (e.g., Hulk, GoldenEye, slowloris, and slowHTTPtest), the features do not show significant change as depicted in Figure 1b. The same goes for DDoS LOIC (TCP) attacks. However, these attacks affected the servers in terms of the number of TCP flows in a small duration. For instance, DoS Hulk only happened for 17 minutes and produced 230,124 attack flows. Similarly, GoldenEye, slowloris, and slowHTTPtest attacks have caused thousands of flows in only 13, 23, and 21 minutes, respectively. The DDoS LOIC (TCP) attack also occurred in under 20 minutes. We do not show the DoS attacks in the 2018 dataset since they have the same pattern as in the 2017 DoS attacks.

The difference between the number of forward and backward packets is more evident in the 2018 LOIC (UDP and HTTP) attacks. Figure 1c shows a clear separability between DDoS and regular flows. Note that for a single regular flow, the largest number of forward packets only achieved 20,000 packets. A single attack flow can reach up to 200,000 forward packets for an HTTP attack and up to 300,000 forward packets for a UDP attack. These are respectively 10 and 15 times the maximum number of forward packets in a regular flow.

Similarly, DDoS attacks of the 2019 dataset also show similar characteristics to the 2018 LOIC attacks. The total number of forward packets has a vast difference compared to the number of backward packets, which is shown in Figure 1d. For instance, a single DDoS traffic flow using the DNS protocol reached a maximum of 100,000 packets. Thus, we expect high detection rates for these datasets given their clear separability, even when using traditional ML methods.

V. Experimental Results

We report the training evaluation of the ML models during the hyperparameter search in this section. Then, we show the classification results of all the ML methods using both the accuracy measures and the ROC curves of the final testing evaluation. Finally, we compare our work with the existing DL solutions utilizing the same datasets.

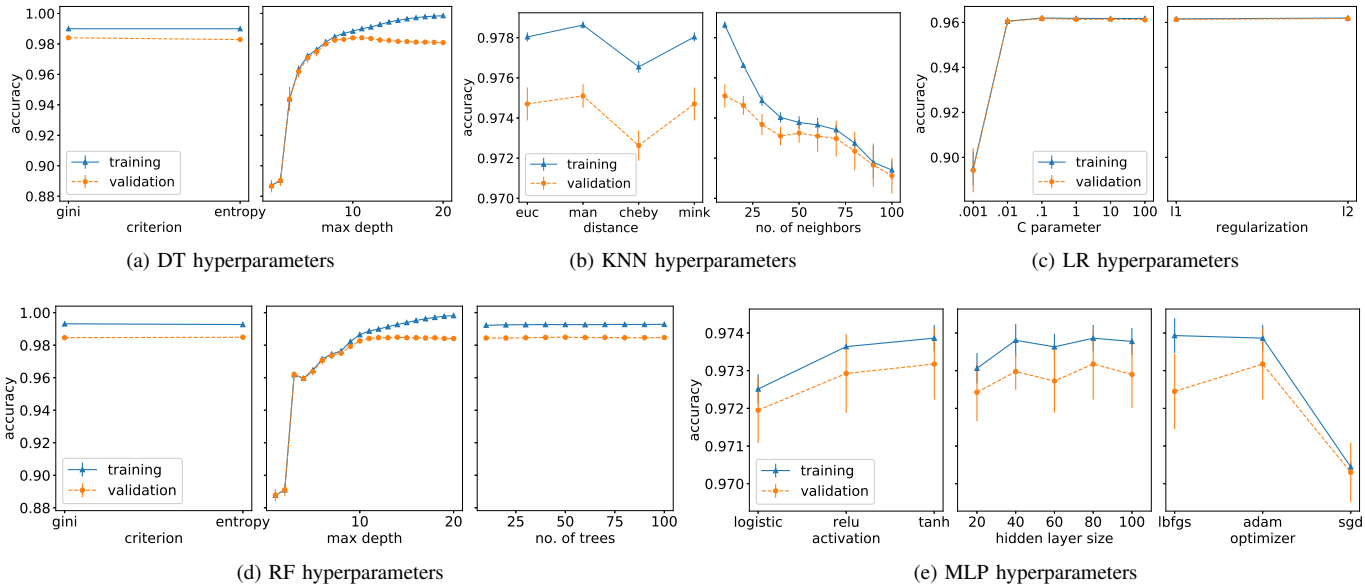
The ML models are developed and evaluated using the tools from scikit-learn[4]. The system running the experiments is composed of 32 CPU cores and 64 GB of RAM.

A. Training Evaluation

The training evaluation is shown in Figure 2, depicting the average training and validation accuracies for each hyperparameter. Figure 2 shows only the result for the ISCXIDS 2012 dataset, which has patterns similar to the other datasets.

For the DT, the figure shows that choosing either Gini or entropy to measure node impurity does not incur a significant

[4]https://scikit-learn.org/stable/index.html



(a) DT hyperparameters (b) KNN hyperparameters (c) LR hyperparameters

(d) RF hyperparameters (e) MLP hyperparameters

Fig. 2: ML model hyperparameter trends affecting accuracy (showing only results for ISCXIDS 2012)

change in detection accuracy. On the other hand, the depth of the tree plays a huge role in improving the accuracy. As the depth increases, the training and validation accuracy increases. However, the validation accuracy stops earlier, which means that tree depth also tends to overfit. The optimum value chosen by GridSearchCV is at max depth = 11, where the validation accuracy is the highest (instead of the training accuracy). The same pattern can be observed for the other datasets.

For the RF, similar patterns can be seen for the depth and criterion from DT, as shown in Figure 2d. The number of estimators does not have a strong influence on increasing accuracy. The patterns for these three hyperparameters are also similar to the other datasets.

For the KNN, the number of neighbors, K, has more influence on accuracy than the distance metric. As K increases, the accuracy decreases, which is the same on all datasets. Thus, K=10 is optimum for all the KNN models. For the distance metric, Manhattan achieved the best accuracy while Chebyshev achieved the least. This result is also the same for the 2017 and 2019 datasets, while Chebyshev achieved best for the 2018 dataset.

For the LR, both L1 and L2 regularization parameters have a minuscule effect on accuracy. In contrast, the C parameter has a significant influence on accuracy across all datasets. For the SVM, the C parameter is uniform across all the datasets, where accuracy decreases as the C parameter increases.

For MLP, the weight optimizer has more influence on accuracy than the hidden layer numbers and their activation functions. For the optimizer, SGD always achieved the least among all the datasets. Adam optimizer achieves best for 2012 and 2017 datasets while LBFGS achieves best for the remaining datasets. Although the differences are minuscule, relu has achieved the highest for the 2017 dataset while tanh achieved best for the remaining datasets.

Finally, the average fitting and validation time during the training phase is shown in Figure 3 using the logarithmic scale. SVM, MLP, and KNN have to take a large training time while DT, RF, and LR have a significantly lesser training time. For the validation time, KNN takes a larger time compared to the rest given its non-parametric nature. KNN does not learn parameters during training (i.e., lazy learner) and uses the training data for prediction. Thus, large numbers of training samples increase the time for a prediction. The figure shows only the results for the 2017 dataset since it has the most number of samples, although the patterns are similar across all datasets.

B. Testing Evaluation

Table III shows the results of ML models for the four main test datasets, reporting the overall accuracy (Ac), F1-measure (F1), precision (Pr), and recall (Rc). For the 2012 dataset, the best result was achieved by RF, reaching 98.3% Ac using entropy with a tree depth of 14. Similarly, DT also provides high Ac, which reached 98.2% using Gini and a tree depth of 11. Linear SVM and LR both achieved 96.3% Ac using a C

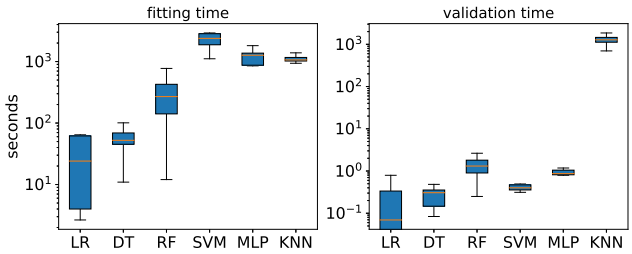
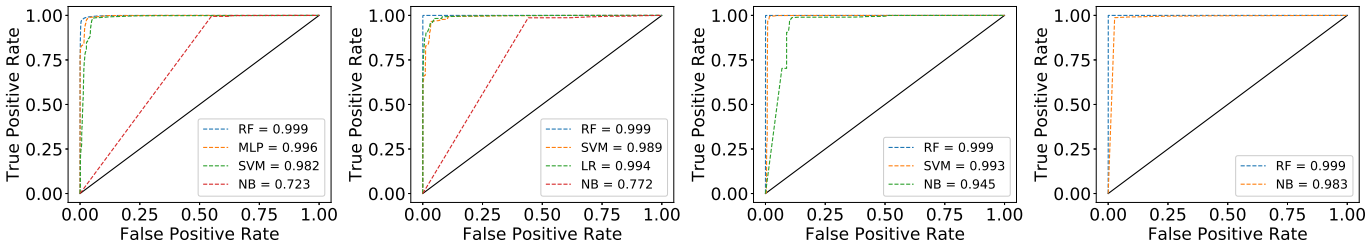


Fig. 3: CICIDS2017 training time evaluation

TABLE III: Evaluation Results of different ML methods for each test dataset

ML Models	ISCXIDS 2012				CICIDS 2017				CSE-CIC-IDS 2018				CICDDoS 2019			
	Ac	F1	Pr	Rc	Ac	F1	Pr	Rc	Ac	F1	Pr	Rc	Ac	F1	Pr	Rc
Naive Bayes	70.601	77.198	63.079	99.460	63.418	69.173	53.248	98.688	91.800	91.932	90.446	93.468	98.337	99.080	99.726	98.441
Logistic Regression	96.332	96.391	94.928	97.901	96.157	95.441	94.186	96.730	98.682	98.697	97.539	99.883	99.797	99.888	99.921	99.855
Decision Trees	98.230	98.221	98.787	97.661	**99.952**	**99.942**	99.908	99.975	**99.988**	**99.988**	99.993	99.983	99.972	99.985	99.990	99.979
Random Forest	**98.335**	**98.329**	98.760	97.901	99.927	99.912	99.877	99.947	99.890	99.890	99.817	99.963	**99.979**	**99.988**	99.996	99.980
K-Nearest Neighbors	97.652	97.647	97.949	97.347	99.839	99.807	99.729	99.885	99.928	99.928	99.902	99.955	99.920	99.956	99.962	99.950
Support Vector Machines	96.430	96.496	94.794	98.261	95.573	94.640	95.317	93.972	97.508	97.565	95.329	99.908	99.792	99.885	99.900	99.870
Multi-Layer Perceptron	97.547	97.570	96.730	98.426	98.949	98.751	97.605	99.924	99.878	99.878	99.784	99.972	99.933	99.963	99.967	99.959



(a) ISCXIDS 2012 (b) CICIDS 2017 (c) CSE-CIC-IDS 2018 (d) CICDDoS 2019

Fig. 4: Evaluating ML model robustness using ROC curves and AUC values (showing only the distinct values)

parameter of 0.01 and 0.1, respectively. All of the algorithms yield high accuracy except for NB.

For the 2017 dataset, DT, RF, and KNN reached over 99% Ac. DT achieved the highest with a maximum tree depth of 14. RF also achieved high Ac through a maximum tree depth of 20 levels and 100 tree estimators. Furthermore, MLP reached 98.9% Ac with relu as the activation function in the hidden nodes and Adam weight optimizer. LR and SVM also achieved high Ac with a C parameter equal to 10 and 0.001, respectively. NB still performed the least in this dataset.

All of the algorithms yield high detection Ac in the 2018 dataset. NB reached 91.8% Ac while the rest achieved greater than 97.5%. DT achieved best with 99.9% Ac followed closely by KNN. RF also achieved 99.8% Ac using 20 tree estimators. Both DT and RF used entropy with a maximum tree depth of 17 and 20 levels, respectively. MLP also reached 99.8% Ac with 100 hidden layers. Linear models, SVM and LR achieved 97.5% and 98.6% accuracy using 0.001 and 0.1 of the C parameter, respectively.

The models yield the highest detection accuracy using the 2019 DDoS dataset. NB achieved 98.3% Ac while the rest obtained over 99%. RF achieved best, which reached 99.97% Ac followed closely by DT. RF and DT models use 20 and 19 levels of tree depth, respectively. MLP also achieved 99.93% using 80 hidden layers with tanh activation function and LBFGS weight optimizer. Linear models, LR and SVM reached the same accuracy of 99.79% using a C parameter value of 100 and 0.1, respectively.

The results have shown accordance with the analysis in Section IV where the 2018 and 2019 datasets show clear separability for the number of forward and backward packets, allowing them to have better Ac results compared to the others.

C. ROC curves

In addition to accuracy metrics, the AUC of each ML model is also reported in Figure 4, showing only the distinct values. Figure 4a shows the AUC of the ML models for the 2012 dataset and shows high AUC values for most of the ML models except for NB. It confirms the results of the accuracy metrics and also shows that RF is the most robust classifier. MLP, KNN, DT, and LR also achieved 0.99 of AUC.

For the 2017 dataset, Figure 4b shows that RF has achieved the largest AUC, which also achieved the highest detection accuracy for this dataset. DT, RF, MLP, and KNN also achieved similar results. NB still has the least performance in detection confidence.

Figure 4c also shows that RF achieved the highest AUC for the 2018 dataset, which also achieved the highest Ac of over 99% for this dataset. DT, KNN, and MLP have also achieved similar results with RF. Finally, Figure 4d shows the robustness of all ML models for the 2019 DDoS dataset, which achieved greater than 0.98 of AUC.

The accuracy metrics and AUC evaluation show that RF and DT achieved outstanding results for DDoS attack detection. On average, DT outperformed RF only by 0.002% Ac while RF outperformed DT only by 0.002 on average AUC. Since RF is an ensemble method composed of multiple DTs to overcome DT's tendency of overfitting, we choose RF to represent our DDoS detection scheme.

D. Literature Comparison

The evaluation concludes with the comparison of current DL approaches that utilize the same dataset. As shown in Figure 5, our results using RF, called MLDDoS, are comparable to current approaches. For the 2012 dataset, DLDDoS [5] use Deep Neural Networks (DNN), LUCID [6] and TR-IDS [16]

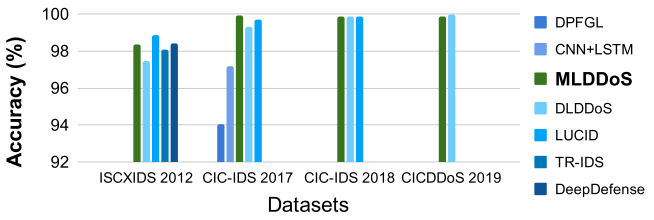


Fig. 5: Comparison with current Deep Learning models

used CNNs, while we show the best result of DeepDefense [9] using LSTM. The figure shows that our approach has similar detection performance to existing DL methods.

For the 2017 dataset, we have outperformed DLDDoS [5], LUCID [6], DeepGFL [13], and the DL approach from [15] using CNN+LSTM. For the DeepGFL, we reported the result for detecting DDoS Hulk in the figure. LUCID, DLDDoS, and MLDDoS have similar detection performance in the 2018 dataset. Finally, only DLDDoS and MLDDoS have utilized the most recent 2019 dataset at the time of writing, which both achieved high detection performance.

VI. Conclusion

In this paper, we show that DDoS attacks can be detected with high accuracy using only traditional ML algorithms. The search for the optimum hyperparameters also supported the development of the ML models to yield high detection performance. Our results show that RF and DT achieved the best performance and worst for NB since it does not have hyperparameters for tuning. We adopted RF as our DDoS detector since it is an ensemble technique composed of DTs that would combat a single DT's overfitting tendencies. We also compared our results to existing approaches in the literature that utilize DL methods with the same datasets. This paper also provides detailed data and model analysis, which are missing in most ML studies. We found patterns from attacks by analyzing the raw data and understand the ML model parameters that are important for tuning to increase detection performance. For future work, we will extend the detection to multiclass to identify the type of attack and deploy it in a real-world environment.

Acknowledgment

This work is developed under the H2020 ASTRID and GUARD projects, which receive funding from the European Commision's Horizon 2020 research and innovation program under Grant Agreement 786922 and 833456, respectively.

References

[1] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *Computer Journal*, vol. 35, no. 10, pp. 54–62, 2002.

[2] M. Repetto, A. Carrega, and A. Duzha, "A novel cyber-security framework leveraging programmable capabilities in digital services," in *ITASEC*, 2020, pp. 1–11.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[4] D. Bhamare, T. Salman, M. Samaka, A. Erbad, and R. Jain, "Feasibility of supervised machine learning for cloud security," in *Int. Conf. on Information Science and Security (ICISS)*. IEEE, 2016, pp. 1–5.

[5] O. R. Sanchez, M. Repetto, A. Carrega, R. Bolla, and J. F. Pajo, "Feature selection evaluation towards a lightweight deep learning ddos detector," in *International Conference on Communications (ICC)*. IEEE, 2021, pp. 1–6.

[6] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa, "LUCID: A practical, lightweight deep learning solution for ddos attack detection," *Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[7] B. Sliwa, N. Piatkowski, and C. Wietfeld, "LIMITS: Lightweight machine learning for iot systems with resource limitations," in *International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[8] O. R. Sanchez, S. Ferlin, C. Pelsser, and R. Bush, "Comparing machine learning algorithms for bgp anomaly detection using graph features," in *Proc. 3rd ACM CoNEXT Workshop on Big Data, Machine Learning and Artificial Intell. for Data Commun. Netw.*, 2019, pp. 35–41.

[9] X. Yuan, C. Li, and X. Li, "Deepdefense: identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2017, pp. 1–8.

[10] Z. He, T. Zhang, and R. B. Lee, "Machine learning based ddos attack detection from source side in cloud," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2017, pp. 114–120.

[11] T. Salman, D. Bhamare, A. Erbad, R. Jain, and M. Samaka, "Machine learning for anomaly detection and categorization in multi-cloud environments," in *2017 IEEE 4th Int. Conf. on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2017, pp. 97–103.

[12] A. Shiravi, H. Shiravi, M. Tavallaee, and A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *comput. & security*, vol. 31, pp. 357–374, 2012.

[13] Y. Yao, L. Su, and Z. Lu, "Deepgfl: Deep feature learning via graph for attack detection on flow-based network traffic," in *IEEE Military Communications Conference (MILCOM)*. IEEE, 2018, pp. 579–584.

[14] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[15] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *9th IEEE Annual Comput. and Commun. Workshop and Conf. (CCWC)*. IEEE, 2019, pp. 452–457.

[16] E. Min, J. Long, Q. Liu, J. Cui, and W. Chen, "TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest," *Security and Commun. Netw.*, vol. 2018, no. 4943509, pp. 1–9, 2018.

[17] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in *International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[18] A. Lashkari, Y. Zang, G. Owhuo, M. Mamun, and G. Gil, "Cicflowmeter (formerly iscxflowmeter)- a network traffic flow analyzer." [Online]. Available: https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter

[19] A. Jain, K. Nandakumar, and A. Ross, "Score normalization in multimodal biometric systems," *Pattern Recognition*, vol. 38, no. 12, pp. 2270–2285, 2005.

[20] S. e. a. Aseervatham, "A sparse version of the ridge logistic regression for large-scale text categorization," *Pattern Recognition Lett.*, vol. 32, pp. 101–106, 2011.

[21] C. P. Diehl and G. Cauwenberghs, "SVM incremental learning, adaptation and optimization," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4. IEEE, 2003, pp. 2685–2690.

[22] L. E. Raileanu and K. Stoffel, "Theoretical comparison between the gini index and information gain criteria," *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, 2004.

[23] M. M. Lau and K. H. Lim, "Investigation of activation functions in deep belief network," in *2nd international conference on control and robotics engineering (ICCRE)*. IEEE, 2017, pp. 201–206.

[24] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng, "On optimization methods for deep learning," in *ICML*, 2011.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, 2015, pp. 1–15.