JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Utilizing Machine Learning Algorithms for Early Detection of DDoS Attacks in Software-Defined Networking.

Jean Paul Ndayizigiye,

Department of Information System and Technology, College of Informatics and Virtual Education, The University of Dodoma, Tanzania.

Abstract. Software-defined networking (SDN) disrupts the vertical integration of current Internet architecture and makes the network programmable from a logically centralized control point. Although centralized network control provides a lots advantages, it remains a challenge for researchers due to attacks against the SDN framework. Identifying DDoS attacks using machine learning is fundamentally a classification issue. In this article, we propose a model based on machine learning to detect a denial of service (DoS) attack in data plane devices, i.e., the OpenFlow switches, resulting from flow-table overflow. An SDN dataset was generated using Mininet and Ryu controllers. The split of the dataset was set at 70% training and 30% testing sets. The five features from a batch of 21 features are extracted using the ExtraTreesClassifier. Further, in this article, two algorithms, (i) Decision Tree (DT) and (ii) K-Nearest Neighbor (KNN), have been tested to detect DDoS attacks and classify the packet as either normal or an attack. The results showed an accuracy of 99.77of Decision Trees, and K-Nearest Neighbor achieved accuracy of 99.38%. We found that Decision Trees generates the best result.

Keywords: software defined networking, distributed denial of service attacks, machine learning algorithms, decision trees and k-nearest neighbor

@

1.Introduction

Software-defined networking (SDN) has been suggested as the internet architecture of the future due to the rising need for high-quality multimedia content. The control plane, which serves as the network's brain, and the data plane are separated in this network model.[1] . SDN controllers, southbound APIs, and northbound APIs are all parts of SDN models. With the help of this design, a centralized, programmable network that can dynamically offer services is made available. [2] . A centralized controller configures and oversees the network's control layer using OpenFlow (OF), a standard and open protocol used in SDN. A variety of complex switching and routing protocols are used to manage the data in SDN, which is stored in Mac tables and routing tables. In conventional networks, these tables are used to form the forwarding plane.[3]. The internet, which is crucial for communication, education, and business transactions, is highly utilized in modern culture. Physical security devices do not have the authority to make decisions since packets are sent according to flow rules, which allows attackers to get around security measures even before they are

deployed. The controller, which serves as the hub of the entire network, has access to numerous network status reports. The attacker can launch significant attacks by using the controller to gain direct access to the network's overall view. SDN has a distinct plane structure, and the attack objects are distinct at each plane. On the control plane, an attacker can do significant harm if they manipulate the controller, who controls the entire network. The majority of attacks in recent years have focused on controllers. DDoS is still one of the most significant security issues on the control plane among the known security flaws. [4]. Due to the significance of controllers in software-defined networks, DDoS assaults on controllers are quite harmful, and researchers are especially concerned with safeguarding controllers against attacks. DDoS assaults flood the network with traffic, suck up network resources, and clog up the network. DDoS assaults are frequently launched from dispersed hosts. [5]. DDoS assaults go through two stages. A distributed attack network of thousands of targeted computers—also referred to as zombies, robots, or attack hosts—is first built by the attacker. The target subsequently receives a flood of traffic from the assault site, either in the attacker's direction or automatically [6]. Attackers look for computers that are less secure, such as those that have not been properly patched, to establish an attack network. A DDoS assault is a violent and disruptive network attack that exhausts system resources and stops the system from functioning. It may completely eliminate the user's access to network resources, gravely endangering the network. Network resources are consumed when malicious data packets are delivered by online attackers, making it difficult or impossible to handle regular traffic. As a result, the network and servers get congested, interrupting regular services. DDoS attackers frequently target SDN due to its distinctive features. In this research, we suggest a method for identifying flow-table overflow-related Denial of Service (DoS) attacks on switches that are part of the data plane. In a flow table overflow attack, the attacker inserts extra flow rules to take advantage of the Ternary Content Addressable Memory (TCAM), where the flow table is stored, for rapid table look-up. When malicious flow-rules exceed flow tables, packets matching new flows are discarded at the switch due to a lack of available capacity. It is depicted in [7] that if the packet rate exceeds 800/second, a switch can become overwhelmed in under 10 seconds. In our method, we use the periodic states of flow tables as well as OpenFlow trace to look for fraudulent switches. System security features including authentication, access control, anti-jamming offloading, and virus detections are now effectively provided by machine learning. [8] .In this study, network traffic is labeled and classified for DDoS attacks using K-Nearest Neighbor (K-NN) and Decision Tree (DT). In the absence of labeled data, DoS assaults can be recognized using multivariate correlation analysis and Q-learning reinforcement learning (RL). The usage of ML in network and routing operations as well as for network security has increased as a result of technological improvement and the transition from traditional Internet architecture to SDN. Our key concept is the use of ML methods to identify DDoS attacks in SDN. To best detect DDoS attacks, we essentially create an ML-based system and train it using an existing dataset. The identification of malicious devices that serve as the channel for DDoS attacks is made possible by the intelligence of ML and the scalability of SDN. Before they cause any harm to the network, such devices can be banned and DDoS attacks can be neutralized. This paper presents a DDoS attack detection technique in an SDN environment using two machine learning algorithms. The main contributions of this paper are summarized below:

1.To design a topology and apply machine learning algorithms on testing environment of linear with one controller SDN networks.

2.Generate SDN dataset using Mininet emulator and Ryu controllers

3.To design a model that detect DDoS attacks using machine learning, evaluate and compare different ML algorithms such as KNN and DT the accuracy of our proposed DDoS detection with other related works.

4.Implement a machine learning model to detect DDoS attacks with two ML algorithms selected such as DT and KNN.

The remainder of the paper is organized as follows: Section II discussed the work related to detecting DDoS in the SDN framework. Our approach to detecting DDoS on SDN switches is described in Section III. The experimental set-up, dataset generation, and performance analysis are provided in Section IV. Finally, conclusions are drawn in Section V.

2.Related Works

In [9] presented an Advanced-SVM technique for SDN networks to detect DDoS assaults using UDP and SYN floods. Using the SDN-Traffics DS and KDDCUP99 datasets for testing and training, the suggested system produced evaluation performance averages for precision, recall, and F1-score of 87%, 84%, and 93%, respectively.

In [10] presented a method based on the fast K-NN model which is effective and accurate at spotting DDoS attacks. A fictitious NSL-KDD dataset was used to test and train the suggested approach. In order to detect DDoS attacks, the proposed approach improved the K-NN detection efficiency, reaching high accuracy, precision, and stability.

In [11] suggested a solution for identifying DDoS attacks based on an upgraded K-NN algorithm that runs on the SDN controller and likewise had good performance in doing so.

Then, to detect abnormal behavior ,in [12] suggested an ensemble ML based on K-NN, naive Bayes (NB), SVM, and self-organizing map (SOM) algorithms. The method tests and prioritizes the model using the CAIDA 2016 dataset. However, both the ensemble and single ML algorithms used in the ensemble approach had low detection accuracy and false-positive rates.

For use in a real-time detection system, in [13] investigated P4 programmable and K-NN, RF, SVM, and ANN algorithms for implementation in a real-time detection system. They proposed an automated DDoS attack detection (DAD) method. The DAD approach detects SYN flood attacks locally on SDN switches with an overall performance of 98%.

In [14] RF, SVM, K-NN, naive Bayes (NB), and decision tree (DT) algorithms were used to defend the SDN controller against DDoS attacks. The method was tested on the NSL-KDD dataset, with DT attaining a high accuracy of 99.97% and SVM achieving a very low accuracy of 60.19%.

To assess and detect TCP-SYN flood DDoS attacks against the SDN controller, in [15] investigated a range of ML classification models, including DT, random forest (RF), AdaBoost (AB), multilayer perceptron (MLP), and logistic regression (LR). The experiment's findings demonstrate that all categorization models performed well.

To categorize SDN network traffic as regular or DDoS attacks, in [16] used ML techniques such K-NN, DT, ANN, and SVM. Among classification algorithms, they demonstrated that DT has the highest accuracy rate (99.75%), while SVM has the lowest accuracy rate (81.48%).

DT and SVM-based algorithms were proposed in [17] as a detection method for DDoS attacks. The KDD CUP dataset was used to examine and test the suggested strategy. They only managed to perform poorly, though. For instance, the accuracy rates for DT and SVM are just 78% and 85%, respectively.

In [18] KNN, SVM, ANN, and NB, four ML classification algorithms, were employed to identify DDoS attacks in an SDN context. With a synthetic dataset used to test the proposed methods, KNN was found to have a high detection accuracy of DDoS attack detection (98.3%). The remaining ML classifiers, in comparison, only managed to reach a minimal level of detection accuracy.

A method based on SVM, DT, NB, and logistic regression (LR) was suggested in [19] to identify DoS and DDoS attacks in the SDN network. The method's accuracy was tested on a fictitious dataset and was found to be 97.5% for SVM, 96% for NB and DT, and 89.98% for LR.

In [20] suggested a method for identifying DDoS assaults based on SVM, DT, K-NN, and BN classifiers. The NSL-KDD dataset was used to test and train the method, and the results showed that the DT classifier (95.16%) had the highest detection rate.

In [21] developed a method to categorize and detect DDoS (i.e., HTTP, UDP flooding attacks, and Smurf) based on seven ML algorithms (i.e., K-NN, RF, NB, SVM, linear regression (LR) DT, and ANN). The suggested method is put into practice at the SDN controller and produces high average detection accuracy across all categorization algorithms.

In [22] suggested a method for detecting DDoS assaults in SDN based on four ML classification approaches (KNN, DT, RF, and KNN). High detection accuracy was attained by the proposed method with 99.89% (KNN), 99.50% (DT), 99.90% (RF), and 99.95% (ANN).

3.Background

Machine learning algorithms be divided into four categories: supervised, unsupervised, semi-supervised, and reinforcement learning. Table 2 explains the main differences between them.

Table 1. Machine Learning Techniques

ML types	Learning	Example
1.Supervised ML	Labeled dataset	KNN, DT, SVM,……
2.Unsupervised ML	Unlabeled dataset	PCA, SVD, K-means
3.Semi-supervised ML	Some labeled and other unlabeled	Linear Logistic, Linear Regression
4.Reinforcement ML	Trial and error (no dataset)	Q-learning, Markov Decision Support……..

We pre-trained the model on a known classed training sample in this study, and the generated data is labeled with 1 and 0 to denote a normal or attack flow. As a result, supervised machine learning techniques were applied. The machine predicts the class of the input data based on the training sample during the testing phase when the input is coupled with a label that denotes a classification class.

Next is a further explanation of the supervised algorithms that have been used in this work:
1.K-nearest neighbor (KNN): The KNN algorithm is a method that uses the supervised algorithm [23]. KNN includes instance-based learning groups. Its method is simple: it compares how similar the test sample and training sample are to find the KNN [24]. KNN [25] is the process of identifying groups of (k) items in learning data that are most similar to the item in fresh data. KNN is a simple classification technique, yet it works effectively [26]. Normally, the distance formula is used to calculate the distances between two x and y objects as indicated in Equation (2):

$$d(x,u) = \sqrt{\sum_{i=1}^{n}(x_i - u_i)^2}$$

Figure 1 depicts the KNN decision rule for a collection of samples split into two classes with K= 1 and K= 4. One known sample is utilized in Figure 1(a) to classify an unknown sample, while multiple known samples are used in Figure 1(b). The parameter K is set to 4 in the final scenario, allowing the classification of the unknown sample to be based on the four samples that are closest to it. Only one of them is a member of the other class, while three of them are members of the same class. The unidentified sample is categorized as belonging to the class on the left in both instances.

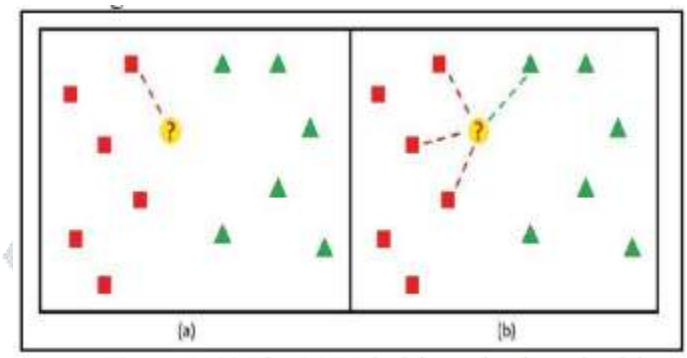

Fig 1.(a) The 1-NN decision rule: the point ? is assigned to the class on the left,(b) the KNN decision rule , with k=4 ,the point ? is assigned to the class on the left as well.

2.Decision Tree: The root node of the decision tree algorithm indicates the strongest predictor, and subsequent predictors are accessed by stem nodes. The decision tree method uses a tree structure that adheres to a set of criteria to identify a class or value. The algorithm eventually reaches leaf nodes, which often stand for a classification or decision[27]. Despite their utility, decision trees have a number of disadvantages, such as instability brought on by even slight changes in the data, which can result in a substantial shift in the tree's structure. Decision tree calculations can occasionally become much more difficult compared to other techniques.

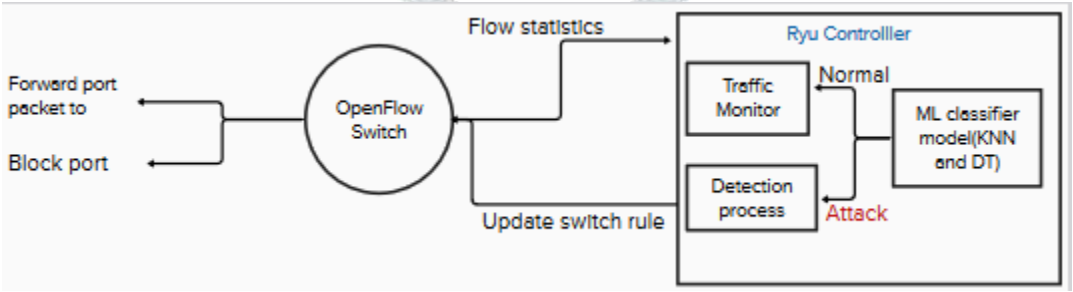

Fig 2. Proposed model

In this work two machine learning algorithms has been proposed to secure SDN networks from DDoS attacks, KNN and DT. The proposed system works as illustrated in Figure 2, the controller extracts the network traffic characteristics from switch flow table and uses it as input to the selected ML classifier to identify the traffic as normal or attack.

4.Methodology

4.1. Experimental environment setup and traffic generation

The experiment setting was carried out on a HP laptop running the Ubuntu 20.04 LTS operating system with a Core i7 CPU and 12 GB of RAM. RYU controller and Mininet emulation were used for the testbed. In this study, Open VSwitches using the OpenFlow protocol version 1.3 were employed. Using the ping command tool, which sends a request over the network to a certain host, regular traffic is produced. A successful ping returns a response to the initiating host from the host that was pinged. The attack traffic is produced using the network utility Hping3, which may transmit unique TCP/UDP/ICMP packets. With the help of this tool, you can manage the size, volume, and packet fragmentation in order to overwhelm the target.

Table 2. Normal and attack traffic specification

Traffic parameters	Normal	Attack
Packet type	TCP	TCP flag
Source IP	10.0.0.1 or any host	Spoofed Random IP
Destination IP	10.0.0.2 or any host	10.0.0.2
Traffic rate (packets/sec)	3	More than 2000

4.2. Network topology

The network topology was constructed using python script to evaluate the effect of topology on the detection techniques efficiency. Linear with one controller topology was the network implemented with 6 hosts but with 3 switches and one controller. Figure 3 illustrate the topologies architecture.

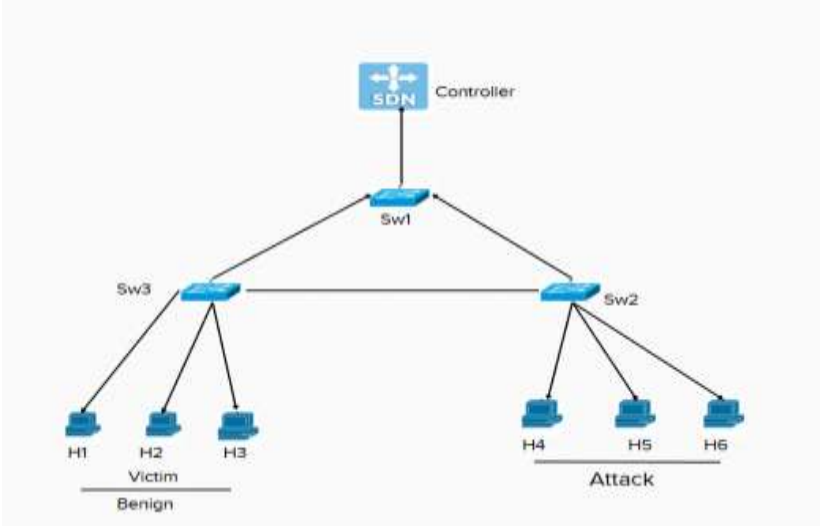

Figure 3. Simulation topology.

A test topology made with Mininet for the evaluation is shown in Figure 3. Each switch has tables that include information about the incoming and outgoing paths of a packet for that switch. The controller has access to the tables. The OpenFlow switch looks for a match in the flow table whenever a new packet arrives. The packet is passed to the controller for additional processing if a match is not discovered in the flow table.

4.3 Dataset generation

The dataset was created using simulation rather than the publicly accessible datasets, which are unrealistic and lack numerous extracted data attributes, in accordance with the situations shown in Figure 3. the datasets produced by utilizing the Mininet emulator to collect the characteristics of network flows during both normal and attack traffic. The flow's extracted features are listed in Table 3. The topology's typical traffic between hosts will be generated at random by the topology's python script using the "Ping" traffic generating command tool. Using the "Hping3" traffic generation command tool, another Python script will randomly generate the attacks flood traffic, which is a DDoS traffic, between hosts in the topology. These are the common command tools used to generate traffic in a simulation test. Data of (434.4 M Byte/5479917 flow) for linear topology is generated throughout the generation process, which lasts for two hours of normal traffic and 25 minutes of attack traffic.

Table 3. Dataset features

Field name	Description
dt	Arrival time of packet
swtch	Switch ID
src_ip	Source IP address
dst_ip	Destination IP address
pkt-count	Number of packets per second
byt-ecount	Number of bytes per second (protocol
dur	Duration of flow
dur-sec	Duration of flow in seconds
dur-tot	The total duration of flow in seconds
flow	Flow identifier
packet-ins	Packet entry
pkt-per-flow	Number of packets per flow
byte-per-flow	Number of bytes per flow
pkt-rate	Number of packets rate
pair-flow	Number of flow entry
protocol	Protocol type
port-no	Number of switch port
tx-bytes	Number of bytes per second
rx-bytes	Number of bytes per nano second
tx-kbps	Number of bytes per kbps
label	0 as normal, 1 as attack

4.4. Feature Selection and Classification

The pre-processed data is utilized to choose features using the ExtraTreesClassifier. To prevent overfitting and overlearning, the ExtraTreesClassifier randomizes various decisions and data subsets. Out of a batch of 21 features, the ExtraTreesClassifier selects the top 5 features. The ExtraTreesClassifier's top 5 features for speeding up calculation. The classification models are then trained using data that has only the top 5 features. To choose the optimal model with the highest accuracy and performance results, two classification models were trained. 1) K-Nearest Neighbors was one of the categorization models that was taken into consideration. Decision Tree 2. Scikit library is used to train the machine learning-based classification models. The default parameters were used to train the Decision Tree model.. The k-value was set to 3 for the KNN classifier model.

4.5. Model Testing and Training

The machine learning classifier model is trained on each dataset when they are created. The dataset was divided at the training stage into 70% training sets and 30% testing sets. The controller will use the learned model during testing phase real-time traffic to determine if the flow is normal or under assault. The trained model is assessed using the confusion matrix and training period. Where the confusion matrix is made up of the true positive (TP), the false positive (FP), the false negative (FN), and the true negative (TN).

Table 4. Confusion Matrix

		Predicted	
		Positive	Negative
Actual	Positive	TP	TN
	Negative	FP	FN

Eq. (1) calculate the accuracy of the model depending on confusion matrix. It's the measure of classified dataset features to the total dataset:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} * 100\%$$

Eq.(2) calculate the precision of the model depending on confusion matrix. This measure is predominantly used when the dataset is imbalanced. It's the ratio of correctly classified data to the sum of correctly classified and incorrectly classified.

$$Precision = \frac{TP}{TP + FP}$$

Eq.(3) calculate the recall of the model depending on confusion matrix. It is the ratio of correctly classified data to the sum of significant attacks. It is additionally called positive sensitivity value:

$$Recall = \frac{TP}{TP + FN}$$

Eq.(4) F1-score: It is defined as the number of observations that have been erroneously categorized.

$$F1 - score = \frac{2*TP}{(2*TP+FN+FP)}$$

4.6. Detection process

After training is complete, the system will be able to recognize and categorize traffic produced by any host inside the topology. The trained model will be called by the RYU controller's Python code, which will then begin testing it on real-time traffic. Every 3 seconds, it will gather the flow statistics from each switch and give them as input to the classification model. The classifier makes a prediction about whether it is an attack or regular traffic, and the controller terminal then shows the outcome. The mitigation phase begins once the controller displays the victim if the traffic is malicious (DDoS). It has been discovered that RYU terminal accurately detects a real traffic event while creating regular traffic between any two sites. The correct detection in the controller topology is shown in Figure 4. We test the detection of attack traffic while the regular flow of traffic is still ongoing. Using the hping3 program, a DDoS attack traffic is formed between two hosts for the topology, with host 6 attacking host 2, using random faked IPs. RYU will identify attack traffic after a brief period and print the victim host. Figure 5 displays the outcome of DDoS detection for the Ryu Controller Topology alone.

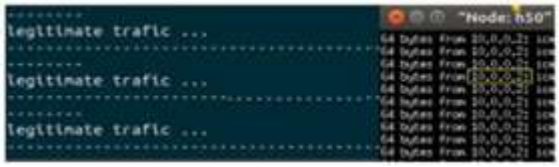

Fig.4 Detecting normal traffic in ryu controller

Fig.5 Detecting attack traffic in ryu controller

The detection process steps are explained in algorithm .

Algorithm : Detection Process

Select the classifier (DT or KNN)
Hard_time = t #block port timeout
1: Create a trained model based on generated dataset
2: Capture the packets every 3 second and process it to get necessary fields(*collect_flow_stat*)
3: Classify the packet using the model
4:If classifier classifies the flow as anomaly, then
5: Display the traffic as attack and specify the victim

5. Results and discussion

The proposed algorithms were evaluated for their capacity to recognize and counteract DDoS attacks in the simulation networks and to demonstrate their accuracy in classifying incoming traffic for the controller. Confusion metrics (Accuracy, Precision, F1, and Recall) are employed during the examination.

Table 5: Accuracy of KNN and DT

Models	*Accuracy*
KNN	99.38%
DT	99.77%

Accuracy values in terms of percentage for KNN and DT are shown in table 5. Accuracy is defined as how accurately the classifier classifies both positive and negative instances into positive class and negative class respectively. KNN algorithm gives accuracy of 99.38%. DT algorithm gives the accuracy of 99.77%.

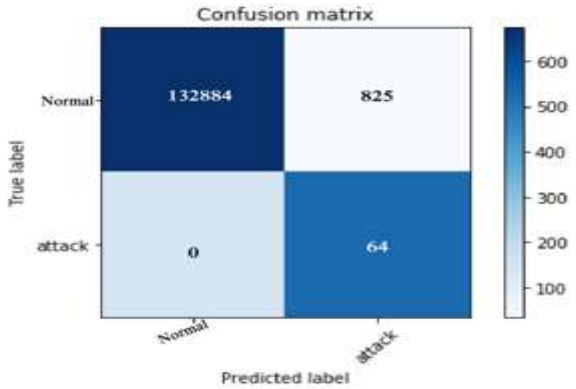

Fig.6.Confusion Matrix of KNN

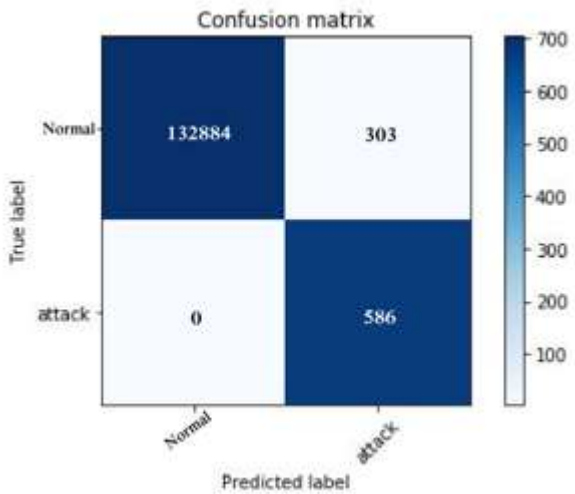

Fig.7 Confusion matrix of D.T

Table.6 Comparison of parameters of the different classification models

Model	TP	TN	FP	FN
KNN	132884	825	0	64
Decision Tree	132884	303	0	586

Table 7. Comparison of the confusion metrics of the two models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1(%)
KNN	99.38	100	99.95	99.95
DT	99.77	100	99.56	99.95

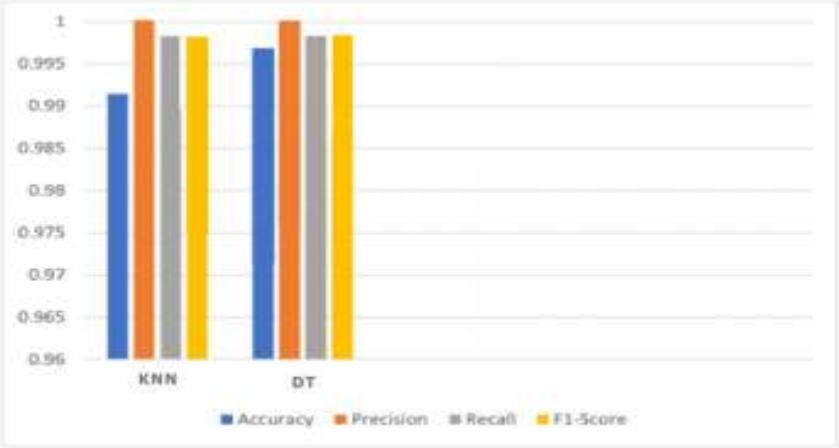

Fig.8 Result of machine learning algorithms with metrics classification

The DT model has the highest accuracy rating, with 99.77%. The KNN model performed the worst, with a substantially greater false negative rate than the KNN model. One of the models' standout characteristics was how rare false positives were, compared to how frequent erroneous negatives were yet still very few in number. While some legitimate traffic may be blocked as a result, this guarantees that no malicious traffic will be mistakenly labeled as legitimate. Table 8 compares the classification models used in this study and other comparable studies.

Table 8. Comparison of the classification models with other related works in terms of accuracy.

Ref.	Classification Model	Accuracy from other works	Accuracy from this research
16	DT (DDoS, SDN)	78%	99.77%
19	DT (DDoS, SDN)	95.16%	99.77%
17	KNN (DDoS, SDN)	98.30%	99.38%
21	KNN (DDoS, SDN)	99.89%	99.38%

Table 8 shows that the DT model, which had an accuracy of 99.77%, performed much better than the linear-based ML models. The DT model is an effective classification model for DDoS detection since it also offers a reasonable trade-off between precision and recall rates.

6.**Conclusion and Future work**

SDN is a framework that is ideal for the dynamic nature of today's applications since it is controllable, economical, and adaptive. One of the current SDN security's biggest issues is the detection of DDoS attacks. Traffic is recorded on the simulated SDN topology and an SDN dataset is developed to solve this issue. Using the trained model, the incoming data packets are categorized as normal or attack. In this study, DDoS attacks are detected using a machine learning method. The SDN dataset, which has 21 features and was produced with the aid of Mininet and Ryu controller, is used, and the top 5 features are taken from it. In this study, two different algorithms—KNN and Decision Tree—are compared to see which categorization model is more effective at spotting fraudulent IP addresses. With an accuracy of 99.77%, the Decision Tree model performs better than the KNN classifier models.

In terms of future work, we suggest employing a real network rather than a virtual one and adding more SDN controllers to the multi-controller topology. Additionally, deep learning algorithms can be used and reinforcement learning that don't require a dataset can also be tested.

7. References

[1] T. E. Ali, A. H. Morad, and M. A. Abdala, "Load balance in data center SDN networks," *Int. J. Electr. Comput. Eng.*, vol. 8, no. 5, pp. 3086–3092, 2018, doi: 10.11591/ijece.v8i5.pp.3086-3092.

[2] T. E. Ali, M. A. Abdala, and A. H. Morad, "Sdn implementation in data center network," *J. Commun.*, vol. 14, no. 3, pp. 223–228, 2019, doi: 10.12720/jcm.14.3.223-228.

[3] T. E. Ali, A. H. Morad, and M. A. Abdala, "Traffic management inside software-defined data centre networking," *Bull. Electr. Eng. Informatics*, vol. 9, no. 5, pp. 2045–2054, 2020, doi: 10.11591/eei.v9i5.1928.

[4] T. Xu, D. Gao, P. Dong, H. Zhang, C. H. Foh, and H. C. Chao, "Defending Against New-Flow Attack in SDN-Based Internet of Things," *IEEE Access*, vol. 5, no. February, pp. 3431–3443, 2017, doi: 10.1109/ACCESS.2017.2666270.

[5] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *J. Inf. Secur. Appl.*, vol. 50, p. 102419, 2020, doi: 10.1016/j.jisa.2019.102419.

[6] C. Bouras, A. Kollia, and A. Papazois, "SDN & NFV in 5G: Advancements and challenges," *Proc. 2017 20th Conf. Innov. Clouds, Internet Networks, ICIN 2017*, pp. 107–111, 2017, doi: 10.1109/ICIN.2017.7899398.

[7] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, "Defending against flow table overloading attack in software-defined networks," *IEEE Trans. Serv. Comput.*, vol. 12, no. 2, pp. 231–246, 2019, doi: 10.1109/TSC.2016.2602861.

[8] M. W. Nadeem, H. G. Goh, V. Ponnusamy, and Y. Aun, "DDoS Detection in SDN using Machine Learning Techniques," 2022, doi: 10.32604/cmc.2022.021669.

[9] M. M. Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Analysis of features dataset for DDoS detection by using ASVM method on software defined networking," *Int. J. Networked Distrib. Comput.*, vol. 8, no. 2, pp. 86–93, 2020, doi: 10.2991/IJNDC.K.200325.001.

[10] Y. Xu, H. Sun, F. Xiang, and Z. Sun, "Efficient DDoS Detection Based on K-FKNN in Software Defined Networks," *IEEE Access*, vol. 7, pp. 160536–160545, 2019, doi: 10.1109/ACCESS.2019.2950945.

[11] S. Dong and M. Sarem, "DDoS attack detection method based on improved KNN with the degree of DDoS attack in software- defined networks," *IEEE Access*, vol. PP, p. 1, 2019, doi: 10.1109/ACCESS.2019.2963077.

[12] V. Deepa, K. M. Sudar, and P. Deepalakshmi, "Design of Ensemble Learning Methods for DDoS Detection in SDN Environment," *Proc. - Int. Conf. Vis. Towar. Emerg. Trends Commun. Networking, ViTECoN 2019*, pp. 1–6, 2019, doi: 10.1109/ViTECoN.2019.8899682.

[13] F. Musumeci, "Machine - Learning - Enabled DDoS Attacks Detection in P4," 2022, doi: 10.1007/s10922-021-09633-5.

[14] M. W. Nadeem, H. G. Goh, V. Ponnusamy, and Y. Aun, "Ddos detection in sdn usingmachine learning techniques," *Comput. Mater. Contin.*, vol. 71, no. 1, pp. 771–789, 2022, doi: 10.32604/cmc.2022.021669.

[15] R. Swami, M. Dave, and V. Ranga, "Detection and Analysis of TCP - SYN DDoS Attack in Software - Defined Networking," *Wirel. Pers. Commun.*, vol. 118, no. 4, pp. 2295–2317, 2021, doi: 10.1007/s11277-021-08127-6.

[16] Ö. Tonkal, H. Polat, E. Başaran, Z. Cömert, and R. Kocaoğlu, "Machine learning approach equipped with neighbourhood component analysis for ddos attack detection in software-defined networking," *Electron.*, vol. 10, no. 11, 2021, doi: 10.3390/electronics10111227.

[17] K. M. Sudar and M. Beulah, "Detection of Distributed Denial of Service Attacks in SDN using

Machine learning techniques," pp. 0–4, 2021.

[18] H. Polat and O. Polat, "Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models," 2020.

[19] A. Ahmad, E. Harjula, M. Ylianttila, and I. Ahmad, "Evaluation of Machine Learning Techniques for Security in SDN," *2020 IEEE Globecom Work. GC Wkshps 2020 - Proc.*, no. September, 2020, doi: 10.1109/GCWkshps50303.2020.9367477.

[20] N. Satheesh *et al.*, "Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network," *Microprocess. Microsyst.*, vol. 79, p. 103285, 2020, doi: 10.1016/j.micpro.2020.103285.

[21] K. S. Sahoo, "A Machine Learning Approach for Predicting DDoS Traffic in Software Defined Networks," no. June 2021, 2018, doi: 10.1109/ICIT.2018.00049.

[22] S. Pande, A. Khamparia, D. Gupta, and D. N. H. Thanh, "DDOS Detection Using Machine Learning Technique," *Stud. Comput. Intell.*, vol. 921, no. 1, pp. 59–68, 2021, doi: 10.1007/978-981-15-8469-5_5.

[23] T. Edition, *No Title*.

[24] S. K. Dey, M. Raihan Uddin, and M. Mahbubur Rahman, *Performance Analysis of SDN-Based Intrusion Detection Model with Feature Selection Approach*, no. May. Springer Singapore, 2020. doi: 10.1007/978-981-13-7564-4_41.

[25] S. B. Imandoust and M. Bolandraftar, "Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events : Theoretical Background," *Int. J. Eng. Res. Appl.*, vol. 3, no. 5, pp. 605–610, 2013.

[26] D. Gunawan, T. Hairani, and A. Hizriadi, "Botnet identification based on flow traffic by using K-nearest neighbor," *2019 Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS 2019*, pp. 95–100, 2019, doi: 10.1109/ICACSIS47736.2019.8979738.

[27] B. A.O., B. A.M., S. P.O, and A. L.B., "An Ensemble Approach Based On Decision Tree And Bayesian Network For Intrusion Detection," *Comput. Sci. Ser.*, vol. 15, no. January, pp. 82–91, 2017.