

Distributed denial of service attacks detection for software defined networks based on evolutionary decision tree model

Hasan Kamel, Mahmood Zaki Abdullah

Department Computer Engineering, College of Engineering, Mustansiriyah University, Baghdad, Iraq

Article Info

Article history:

Received Mar 21, 2022

Revised May 9, 2022

Accepted Jun 22, 2022

Keywords:

Distributed denial of service attacks

Evolutionary decision tree

Genetic algorithm

Machine learning

Software defined networks

ABSTRACT

The software defined networks (SDN) system has modern techniques in networking, it separates the forwarding plane from the control plane and works to collect control functions in a central unit (controller), and this separation process leads to many advantages, such as cost reduction and programming ability. Concurrently, because of its centralized architecture, it is prone to a variety of attacks. Distributed denial of service (DDoS) attack has a significant impact on SDN, it is characterized by its ability to consume network resources as well as its ability to turn off the entire network. The work in this study aims to improve and increase the security and robustness of SDN systems against the attack or intrusion, by using a machine learning model to detect attack traffic and classify traffic of SDN as (attack or normal), and optimization algorithm (genetic algorithm) for improving the accuracy of the classification. After preparing and preprocessing the dataset, we used the genetic algorithm (GA) to optimize the hyperparameters of the decision tree (DT) model, and the proposed evolutionary decision tree (EDT) model was used to classify traffic into normal and attack traffic. The results indicate that the suggested model achieved a high classification accuracy of 99.46.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Hasan Kamel

Department Computer Engineering, College of Engineering, Mustansiriyah University

Al-Bab Al-Muadham Street, Baghdad, Iraq

Email: Egma003@uomustansiriyah.edu.iq, drmzaali@uomustansiriyah.edu.iq

1. INTRODUCTION

The software defined networks (SDN) is a modern networks technique, it separates the data level from the control level and works to collect control functions in a central unit (controller) to provide several advantages such as cost reduction, programming ability and provide management of the entire network from a single point [1]. In SDN, the control level and redirect level are separated, thus realizing that the control level is managed using a programmable central controller where policies are configured for each device, be it a switch, router, or firewall, i.e., we leave from a network with a distributed control level to a network with a central control level. Consequently, the controller which has the capability to handle the entire-network from a centralized point, can quickly impose various network policies across the overall infrastructure [2], [3]. Despite the advantages offered by this modern structure, its centralized structure makes it vulnerable to attacks of its own, in addition to the well-known attacks against traditional networks [4]. Among the attacks that software-defined networks are exposed to, attacks on the central unit (controller) are among the most hazardous types of attacks. If an intruder (attacker) takes over the controller can possess the facility to manage or disable network traffic [5]. The most common attacks against the controller are distributed denial of service (DDoS) attacks, wherein users are rejected arrival to the network services. The attackers seek to generate a lot of traffic using multiple machines, exhaust the resources of the host computer, and balk it from serving via a DDoS

attack [6]. In the recent period, DDoS attacks have become one of the most famous and very dangerous attacks, and they can be devastating to a variety of network services [7]. Botnets, which are made up of zombie devices taken over by internet hackers, are used by attackers. DDoS attacks are difficult to identify and block because they involve a large number of devices [8], [9]. As a result, one of the most pressing issues for administrators and network service providers is the rapid identification and mitigation of DDoS attacks. DDoS attacks can disable different SDN layers by flooding communication-channels between the switch and the controller, or between the application layer and the controller, with excessive flow data. On the controller, there is no built-in security system that can differentiate between an intrusion and normal traffic. As a result, detecting an attack is extremely difficult. Volumetric attacks, resource-consuming attacks, and application layer attacks are common types of DDoS attacks [10]. As a result, their detection is difficult. This topic includes assaults against hypertext transfer protocol (HTTP) and domain name system (DNS) protocols [11]. Servers are rendered inaccessible in resource-consuming attacks by exploiting weaknesses in protocols implemented at the network layer. Transmission control protocol-synchronize (TCP-SYN) flood depletes the target machine's resources (memory, CPU, and storage) [12]. Its goal is to use volumetric attacks to devour the network's bandwidth. Common attacks like ICMP, user datagram protocol (UDP), and TCP-SYN flood take advantage of flaws in layer 4 and layer 3 protocols [13]. In this work, we focus on improving the security of SDN against intrusion by using a machine learning algorithm. We use a standard public dataset with a sum of 23 features for discovering DDoS attacks using machine learning for this purpose. We employed a machine learning method to classify SDN traffic into legitimate or illegitimate traffic. Next, we used an optimization algorithm to optimize (tuning) the hyperparameters of the model and improve the classification accuracy. The results appear that the proposed approach is more efficient than utilizing merely a machine learning model. The remainder of the work (paper) was arranged in this manner: the next portion demonstrates some of the previous-works. In section 3, we provide a brief explanation of the data set used. In addition, the proposed-method, machine-learning, and optimization-method are briefly discussed in this part. Section 4 contains the results and analysis. Section 5 provides the discussion and future-work.

2. LITERATURE REVIEW

In latest years, various investigations have been performed to protect SDN utilizing machine-learning methods and others techniques. This section discusses several research on DDoS security procedures based on machine and deep learning approaches and others techniques. The majority of SDN-based moving target defense (MTD) techniques have been created with a single SDN controller, which introduces a single point of failure and a scalability concern for large-scale networks. To assure both performance and security, the author proposes an SDN-based MTD architecture with several SDN controllers in paper [14]. A developing field of research is the use of "defensive cyber-deception" to improve the security and reliability of network-based systems. To deploy more effective "cyber deception", current honey technologies require more underlying infrastructure. The author investigates how to deploy "deception" in enterprise networks using SDN technology in paper [15]. Security mechanisms like the intrusion-detection-system (IDS) and the intrusion-prevention-system (IPS) are employed to enhance network security. Because of the increasing variety of attacks, statistical calculations must be used on these systems. Machine learning techniques have enabled intrusion detection systems to make useful predictions and remarks. A paper [16] developed an ensemble strategy for detecting DDoS attacks, they utilized four distinct machine learning methods. With (98.12%) accuracy, the SVM-SOM algorithm outperformed the other machine learning (ML) algorithms. DoS data set from the Canadian-Institute of cybersecurity (CIC) is used to test several classification algorithms (machine learning and deep learning algorithms). Out of all the machine learning algorithms tested, the multi-layer perceptron algorithm (MLP) produced the best results with 95% accuracy in paper [17]. Two models were used in the paper [18] to identify UDP flooding assaults in an SDN scenario. For traffic packet creation, they employed the Scapy software. The OpenFlow switch is used by their system to obtain flow stats. They tested the results of linear and polynomial-SVM models for classification after the phase of features extraction. The SVM (polynomial-SVM) method has a decreased false alarm rate of 34% and a higher accuracy of 3%, according to experimental results. proposed a security framework for detecting DDoS attacks in SDN architecture. The system is based on a paradigm of adaptive learning that classifies traffic using historic data. For efficient accuracy results, they applied a cross-validation approach. Although the results are encouraging, the adaptive security model needs to be evaluated on a variety of datasets from the real world to ensure that it is more realistic. In the SDN environment, paper [19] presented a novel security paradigm for DDoS attacks. The model is made up of two stages that use machine learning algorithms. The k-means technique is used in the data processing stage to choose the best features, and the k-nearest neighbor (kNN) approach is used in the detection stage to detect attack flows. Their approach has a 98.85% accuracy rate and a 98.47% recall rate. Ensemble technique was employed in the paper [20] to enhance IDS efficiency. The traditional NSL-KDD dataset is used to test multiple classification algorithms [21]. Karan *et al.* [22] presented a system for a DDoS attacks detection in SDN. The system

employed two levels of security. They began by employing Snort to detect signature-based attacks. They next classified attacks using the SVM model and the DNN classifier. The experimental results demonstrated that DNN has a higher classification accuracy rate than SVM, with a rate of 92.30%. To detect attack flows, a DDoS security system based on SDN architecture was proposed [23]. Their hybrid solution employs a combination of kNN and SOM algorithms. They use flow stats. gathered from SDN switches and classify the traffics into regular or malicious.

It is obvious from the summary of previous studies that the performance of intrusion-detection systems is highly dependent on the nature of the data sets. Several datasets have been used in previous studies such as (Cup'99, CAIDA 2016, CICIDS2017, UNB-ISCX, NSL-KDD, and CIC DoS). These data sets are outdated and attack characteristics are constantly changing. Therefore, there is an increasing need to use up-to-date data sets gained from SDN scheme. There are just a few available to the public datasets for use in SDN-based intrusion-detection systems [24], [25]. In our research, we used the "DDOS-ATTACK SDN DATASET.", which is a recent dataset created recently in a software defined networking environment.

3. PROPOSED METHODOLOGY

The proposed work (method) has three parts: dataset and preprocessing, optimization algorithm for hyperparameters optimization (tuning) and improving the accuracy of ML model, and evolutionary machine learning algorithm to classify traffic into normal traffic or attack traffic. Figure 1 shows an overview of this technique. The classes and features of the public dataset used in this section are clarified. The machine learning model used to classify network traffic and the optimization algorithm used to optimize and fine-tune the hyperparameters of the machine learning model that will rise the classification efficiency and accuracy are explained in detail in this section.

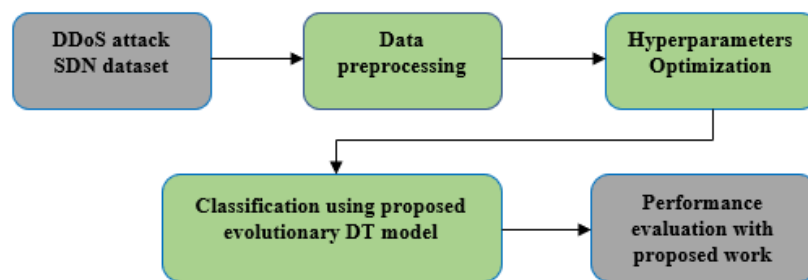


Figure 1. Overview of the proposed work

3.1. Dataset

The new-dataset ("DDOS-attack SDN dataset"), which was generated within the SDN framework (environment) and made publicly available to researchers to be used in machine learning research, was employed in this study [26]. The dataset contains 1,04,345 traffic-flows, 23 features, and consist of (UDP, TCP, and ICMP) protocols as attack and normal traffics. Except for the features that define the source and target, the dataset contains statistical (numerical) features such as packet per flow, byte count, packet rate, and duration sec. The data must be preprocessed before beginning machine learning model training. Several preprocessing techniques were applied to the data set. Missing value handling, null value removal, categorical value encoding, and other pre-processing techniques are used. Categorical values with no numerical values, such as source-destination internet protocol (IP) and protocol, were encoded using one-hot encoding [27]. We then attempted to find the correlation (correlation) between output and the input features using a variety of machine learning methods, heatmap graphs, and correlation techniques. As a result of this procedure, the column containing time data that was displayed with the "dt" feature was determined to be useless and was deleted from the data set. By applying adjustment (normalization) to numeral (numeric) data, the data preprocessing phase was completed.

3.2. Hyperparameter optimization using optimization algorithm (genetic algorithm)

For ML models, choosing the appropriate hyperparameter configuration has a direct effect on the performance of the model. It frequently necessitates extensive knowledge of ML algorithms as well as appropriate hyper-parameter optimization techniques. Although multiple automatic optimization approaches exist, when applied to various types of problems, they have varied strengths and disadvantages. Building an

efficient ML model is a time-consuming and complex process which involves finding the best algorithm and tuning hyper-parameters to obtain the best model architecture. The genetic algorithm (GA) [28] is a popular metaheuristic algorithm based on the-evolutionary hypothesis that individuals with the better survival and environmental adaptability are more able to live and passing on their qualities to future generations. The characteristics of their parents will be passed down to the following generation, which may include both good and bad individuals. Better individuals will have a higher chance of surviving and having more capable offspring, while the worst will eventually fade away. The-individual with the best adaptability will be selected as the global optimum after multiple generations [29]. To apply the genetic algorithm to hyperparameter optimization problems, each individual or chromosome represents a hyperparameter, and its decimal-value is the input value for the hyper-actual parameter in each evaluation. Every chromosome contains several genes, which are binary digits, and the genes of this chromosome are then subjected to crossover and mutation operations. The population represents all possible combinations within the initialized chromosome/parameter ranges, whereas the fitness function denotes the parameter evaluation measures [30]. Because the parameter values that are randomly initialized typically do not contain the best parameter ranges, several-operations, such as selection stage, crossover stage, and mutation stage, must be performed on the well-performing chromosomes to identify the optimums [31]. Chromosome selection is carried out by selecting chromosomes with high fitness function values. To keep the population size constant, chromosomes with high fitness function values are more likely to be carried on to the next generation, where they generate new chromosomes with the best characteristics of their parents. Chromosome selection ensures that the best traits of each generation are passed down to future generations. Crossover is a method of creating new individuals (chromosomes) by exchanging a proportion of genes between chromosomes. Mutation operations can also be used to generate new chromosomes by randomly changing one or more genes on a chromosome. Mutation and crossover operations allow for different characteristics in later generations and reduce the probability of missing good characteristics [32]. The following are the main genetic algorithm procedures [33]: i) initialize the population, chromosomes (each chromosome represent set of hyper-parameters), and genes at random, representing the whole search space, hyper-parameters, and hyper-parameter values, respectively, ii) compute the fitness-function, which represents the objective-function of an ML model, to examine (evaluate) the performance of each individual in the current generation, iii) run selection, crossover, and mutation operations on the chromosomes to generate a new generation containing the next hyper-parameter configurations to be tested, iv) repetition steps 2 and 3 until the stop condition is satisfied, and v) end the program and display the optimum hyper-parameter configuration.

In the steps, the initial population of hyperparameter configuration-candidates is generated using random initialization with stochastic (random values) in the specified search space. Our objective-function (accuracy (ACC)) is a maximization problem as shown in (1), in the execution of the genetic algorithm, the evaluation, selection, and recombination processes represent one generation. Several open-source libraries exist to implement evolutionary algorithms such as Genetic Algorithm in practice, in our work we used distributed evolutionary algorithms in python (DEAP) library for hyperparameter optimization. DEAP [34] is a novel Python evolutionary computation package that includes several evolutionary algorithms such as genetic algorithm and differential evolution. It works with parallelization mechanisms such as multiprocessing and machine learning packages such as sklearn, DEAP built-in functions were used for evaluation, mutation, crossover (one-point crossover), and tournament selection. The genetic algorithm was run with the following hyperparameters (population_size=10, mutation_probabilty=0.10, crossover_probabilty=0.5, tournament_size=3 and generations_number=15). After executing all the steps, we will get the best possible accuracy and the better possible-combination of the hyperparameters of DT model which are shown in Table 1. Figure 2 shows the flow chart and the steps of genetic algorithm.

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

Where TP, FP, tn and fn represent the elements of the confusion matrix, which will be explained later.

Table 1. The hyper-parameters and configuration space for DT model

Hyper-parameter	Type	Search space
criterion	Categorical	['gini', 'entropy']
splitter	Categorical	['random', 'best']
max-depth	Discrete	[4,50]
max-features	Discrete	[1,22]

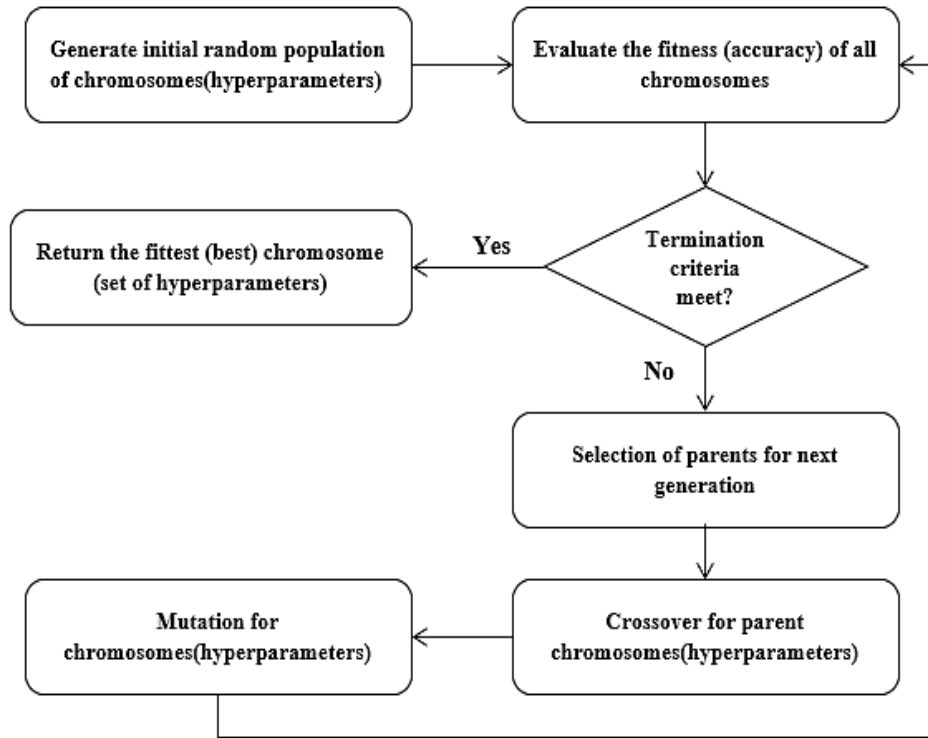


Figure 2. Flow chart of genetic algorithm

3.3. Classification using proposed evolutionary decision tree

For regression and classification of real-world situations, the decision tree machine learning algorithm is utilized. This model is based on the structure of a tree. The tree's root, on the other hand, is at the very top. The branches are built using objective rules based on the dataset's features and the decision tree is also evolved gradually [35]. The processes outlined can be used to generate a decision tree [36]: i) the entire dataset is split into two sections: training and test sets, ii) the training set is used as an input to the tree's root, iii) as shown in (2), the root is found using information theory, iv) the prone-procedure is followed, and v) the steps from 1 to 4 are repeated until all nodes have turned into leaf nodes.

$$Entropy (P) = -\sum_{i=1}^N p_i \log (p_i) \quad (2)$$

Where p stands for the dataset's probability distribution. In order to get an efficient decision tree, other hyperparameters must be tuned (optimized). After conducting many experiments, we concluded that the most important hyperparameters that greatly affect the efficiency of the model results and that need to be tuned (optimized) such as (criterion: the function for determining a split's quality, splitter: the method for selecting the split at each node, max-depth: The tree's maximum depth and max-features: the number of characteristics (features) to consider while looking for the ideal split). As shown in Figure 2, after performing the preprocessing of the data set, the process of optimizing the hyperparameters was implemented using the genetic algorithm to obtain the appropriate values for the hyperparameters of the machine learning model and then used them in the DT model to get the best possible accuracy.

4. RESULTS AND DISCUSSION

This section discusses the results of experiments and the findings of the proposed evolutionary machine learning model, as well as comparing the findings of the suggested model with other studies. Binary classification was used on the public "DDoS attack SDN dataset", and it was done using the sklearn python machine learning framework.

4.1. Performance evaluation using performance metrics

The experimental results in terms of performance investigations done to define the legitimate and malicious network records generated with SDN were tested using a confusion matrix. This matrix contains

both estimated and actual values. Table 2 shows the confusion matrix. True-negative (TN), true-positive (TP) values indicate correctly expected network motion, whereas false-negative (FN), false-positive (FP) indicate incorrectly expected network motion [37]. Furthermore, the receiver operating curve (ROC) and areas under the curves (AUC) were used to assess model performance. The false_positive rate and true positive rate are represented by the x (horizontal) and y (vertical) axes, respectively, in the ROC curve [38].

Table 2. Confusion matrix

Predicted class	True class	
	TP	FP
	TN	FN

The accuracy (ACC), precision (Pr), specificity (Sp), sensitivity (Se), F1-score and performance measures obtained from confusion matrix were used to evaluate the suggested model. These metrics' formulas are as:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$Sp = \frac{TN}{TN+FP} \quad (4)$$

$$Se = \frac{TP}{TP+FN} \quad (5)$$

$$F1 - score = \frac{2 \times se \times pr}{se + pr} \quad (6)$$

$$Pr = \frac{TP}{TP+FP} \quad (7)$$

4.2. Performance evaluation based on proposed evolutionary decision tree algorithm

In this stage after the preprocessing step, the dataset was partitioned into two parts: testing and training at a rate of 0.3 and 0.7, respectively, and the GA was used to optimize hyperparameters of the DT algorithm. The genetic algorithm was constructed using the following hyperparameters (population size=10, mutation probability=0.10, crossover probability=0.5, tournament size=3 and generations number=15). Hyperparameter-optimization technique is used to get suitable hyperparameters and improve the accuracy and efficiency of the DT model. After performing the hyperparameter optimization process using the genetic algorithm, then a decision tree model was built using the obtained hyperparameter values and the model accuracy was 99.46%. Table 3 demonstrates the classification results obtained and the hyperparameter values shown in the Table 4 were obtained. The accuracy (acc) of the traffic is measured by how well the classifier predicts both benign and anomalous classes. Precision (Pr) expects the percentage of traffic that is normal or malware, based on the count in the dataset. The measure of the negative class prediction in the dataset is known as specificity (Sp). The sensitivity (Se) metric assesses a model's facility to estimate true positives in each category. The F1-score represent a metric for determining how accurate a test is, it is calculated using the test's precision and recall. Figure 3 show ROC curve of evolutionary decision tree algorithm (EDT), ROC curve is a binary classification problem evaluation metric. It's a likelihood curve that compares true positive rate (TPR) to false positive rate (FPR) at various thresholds. AUC is a summary of ROC curve that provides the possibility of classifier to identify between classes.

Table 3. Classification results of evolutionary decision tree algorithm

Acc (%)	Pr (%)	Sp (%)	Se (%)	F1-score (%)
99.46	99.19	99.07	99.80	99.42

Table 4. The values of hyper-parameters obtained after implementing genetic algorithm

Hyper-parameter	Type	Value
criterion	Categorical	['entropy']
splitter	Categorical	['best']
max-depth	Discrete	49
max-features	Discrete	18

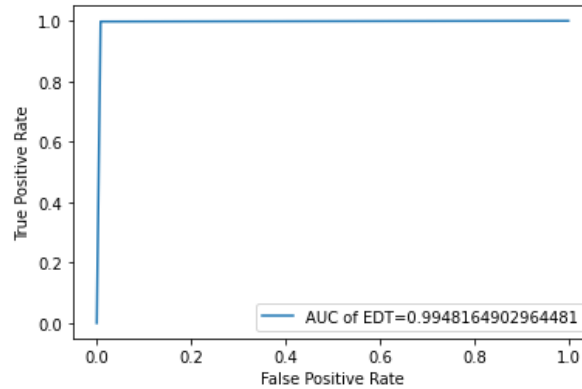


Figure 3. ROC curve of EDT algorithm

Table 5 compares the results of research on DDoS attack traffic detection using machine learning techniques with the model we propose. When looking at Table 4, it's clear that several datasets were utilized to detect attack traffic. Many of the authors employed public datasets including network traffic statistics from classical network architectures, such as NSL-KDD, UNB-ISCX, KDD Cup'99, CICIDS2017, and CAIDA2016 [16], [17], [22]. These datasets are useful for assessing the performance of machine learning techniques used in attack traffic detection. However, because the SDN design differs from traditional network architecture, it has its own set of attack vectors in addition to the present ones. Furthermore, the growing volume and variety of attack traffic necessitates the usage of up-to-date data-sets. As a result, researchers, [18], [19], [25] employ datasets collected through the SDN architecture in their studies. The Study Group (Bennett-University study group) for deep learning and machine learning studies created the SDN dataset that was used in this research. The most critical condition for choosing this dataset is that it was developed utilizing SDN architecture and includes modern SDN DDoS traffic data. We note from the information mentioned in Table 5 that the model proposed by us achieved high results compared to other works.

Table 5. A comparison of relevant work

Datasets and Related studies	ML techniques	Accuracy (%)
CIC DoS dataset [17]	REP Tree, MLP, Random Tree, J48, SVM, and Random Forest	95.00
Their dataset [18]	Linear SVM-Polynomial SVM	95.00
Their dataset [19]	KNN and K-Means	98.85
CAIDA 2016 [16]	SVM, Naive Bayes, SOM, and KNN	98.12
KDD cup 99 [22]	DNN and SVM	92.30
DDOS attack SDN Dataset [25]	LR, ANN, KNN, SVC, RF, Ensemble Classifier, SVC-RF	98.80
DDOS attack SDN Dataset	Our proposed evolutionary DT	99.46

Machine learning models are highly good in detecting attack traffic, according to the results. Our work aims to contribute to the research being conducted in this field (assaults detection in SDN utilizing machine-learning and optimization techniques). The use of a GA for hyperparameters optimization improved the accuracy of machine learning approaches in identifying attack traffics, according to our results. Experimental studies were conducted by selecting the hyperparameters automatically by using the GA to choose the appropriate values for the hyperparameters that make the model accuracy as best as possible. It can be said that the model's classification performance contributes positively to the attack classification when used in conjunction with hyperparameters optimization algorithms.

5. CONCLUSION

In this study, an evolutionary machine learning-algorithm was used to classify attack and normal traffic in a dataset generated from an SDN environment. The dataset contains 1,04,345 records and 23 features and consist of (UDP, TCP, and ICMP) protocols as attack and normal traffics. The data includes numerical (statistical) features such as packet rate, byte count, packet per flow and duration-sec, in addition to features that indicate source and destination devices. The GA was used to perform efficient classification and select the most appropriate hyperparameters for the decision tree model. After conducting several experiments, the most

hyperparameters that affect the efficiency of the machine learning model were extracted, the optimal values for these hyperparameters were determined using the genetic algorithm. After preprocessing and hyperparameter optimization, the suggested method was able to classify over 100,000 network records. According to the outcomes of the tests, the proposed model has a 99.46% accuracy rate. In the future, it is planned to raise the diversity of assaults and discuss the classification efficiency and performance of ML models with hyperparameter optimization algorithms.




REFERENCES

- [1] P. E. Numan *et al.*, "On the latency and jitter evaluation of software defined networks," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 4, pp. 1507–1516, Dec. 2019, doi: 10.11591/EEI.V8I4.1578.
- [2] A. C. Jaramillo, R. Alcivar, J. Pesantez, and R. Ponguillo, "Cost Effective test-bed for Comparison of SDN Network and Traditional Network," in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, 2018, pp. 1–2, doi: 10.1109/PCCC.2018.8711223.
- [3] E. Amiri, E. Alizadeh, and M. H. Rezvani, "Controller selection in software defined networks using best-worst multi-criteria decision-making," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 4, pp. 1506–1517, Aug. 2020, doi: 10.11591/eei.v9i4.2393.
- [4] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, Jun. 2020, doi: 10.1016/j.jnca.2020.102595.
- [5] B. Mladenov and G. Iliev, "Optimal software-defined network topology for distributed denial of service attack mitigation," *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 6, pp. 2588–2594, Dec. 2020, doi: 10.11591/eei.v9i6.2581.
- [6] M. A. Naagas, E. L. Mique, T. D. Palaoag, and J. S. D. Cruz, "Defense-through-deception network security model: Securing university campus network from DOS/DDoS attack," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 4, pp. 593–600, Dec. 2018, doi: 10.11591/eei.v7i4.1349.
- [7] P. Sharma, D. Saini, and A. Saxena, "Fault Detection and Classification in Transmission Line Using Wavelet Transform and ANN," *Bulletin of Electrical Engineering and Informatics*, vol. 5, no. 3, pp. 284–295, Sep. 2016, doi: 10.11591/eei.v5i3.537.
- [8] W. Nazih, W. S. Elkilani, H. Dhahri, and T. Abdelkader, "Survey of Countering DoS/DDoS Attacks on SIP Based VoIP Networks," *Electronics*, vol. 9, no. 11, p. 1827, 2020, doi: 10.3390/electronics9111827.
- [9] T. Horak, P. Strelec, L. Huraj, P. Tanuska, A. Vaclavova, and M. Kebisek, "The Vulnerability of the Production Line Using Industrial IoT Systems under DDoS Attack," *Electronics*, vol. 10, no. 4, p. 381, 2021, doi: 10.3390/electronics10040381.
- [10] C. Hu, L. Han, and S. M. Yiu, "Efficient and secure multi-functional searchable symmetric encryption schemes," *Secur. Commun. Networks*, vol. 9, no. 1, pp. 34–42, Jan. 2016, doi: 10.1002/sec.1376.
- [11] A. Praseed and P. S. Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 661–685, 2019, doi: 10.1109/COMST.2018.2870658.
- [12] T. Mahjabin, Y. Xiao, G. Sun, and W. Jiang, "A survey of distributed denial-of-service attack, prevention, and mitigation techniques," *International Journal of Distributed Sensor Networks*, vol. 13, no. 12, p. 1550147717741463, Dec. 2017, doi: 10.1177/1550147717741463.
- [13] M. A. M. Yusuf, F. H. M. Ali, and M. Y. Darus, "Detection and Defense Algorithms of Different Types of DDoS Attacks," *International Journal of Engineering and Technology*, vol. 9, no. 5, pp. 410–444, 2018, doi: 10.7763/ijet.2017.v9.1008.
- [14] J. Narantuya *et al.*, "SDN-Based IP Shuffling Moving Target Defense with Multiple SDN Controllers," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Supplemental Volume (DSN-S)*, 2019, pp. 15–16, doi: 10.1109/DSN-S.2019.00013.
- [15] C.-Y. J. Chiang *et al.*, "On Defensive Cyber Deception: A Case Study Using SDN," in *MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM)*, 2018, pp. 110–115, doi: 10.1109/MILCOM.2018.8599755.
- [16] V. Deepa, K. M. Sudar, and P. Deepalakshmi, "Design of Ensemble Learning Methods for DDoS Detection in SDN Environment," in *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, 2019, pp. 1–6, doi: 10.1109/ViTECoN.2019.8899682.
- [17] J. A. Pérez-Díaz, I. A. Valdovinos, K.-K. R. Choo, and D. Zhu, "A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning," *IEEE Access*, vol. 8, pp. 155859–155872, 2020, doi: 10.1109/ACCESS.2020.3019330.
- [18] A. T. Kyaw, M. Z. Oo, and C. S. Khin, "Machine-Learning Based DDOS Attack Classifier in Software Defined Network," in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2020, pp. 431–434, doi: 10.1109/ECTI-CON49241.2020.9158230.
- [19] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, "A New Framework for DDoS Attack Detection and Defense in SDN Environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020, doi: 10.1109/ACCESS.2020.3021435.
- [20] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of global optimization* vol. 39, no. 3, pp. 459–471, 2007, doi: 10.1007/s10898-007-9149-x.
- [21] K. S. Bhuvaneshwari, K. Venkatachalam, S. Hubálovský, P. Trojovský, and P. Prabu, "Improved Dragonfly Optimizer for Intrusion Detection Using Deep Clustering," *CMC-Computers Materials & Continua*, vol. 70, no. 3, pp. 5949–5965, 2022, doi: 10.32604/cmc.2022.020769.
- [22] K. B. V., N. D. G., and P. S. Hiremath, "Detection of DDoS Attacks in Software Defined Networks," in *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, 2018, pp. 265–270, doi: 10.1109/CSITSS.2018.8768551.
- [23] T. M. Nam *et al.*, "Self-organizing map-based approaches in DDoS flooding detection using SDN," in *2018 International Conference on Information Networking (ICOIN)*, 2018, pp. 249–254, doi: 10.1109/ICOIN.2018.8343119.
- [24] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurchut, "InSDN: A Novel SDN Intrusion Dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020, doi: 10.1109/ACCESS.2020.3022633.
- [25] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *Journal of Network and Computer Applications*, vol. 187, p. 103108, Aug. 2021, doi: 10.1016/j.jnca.2021.103108.
- [26] N. Ahuja, G. Singal, and D. Mukhopadhyay, "DDOS attack SDN Dataset," vol. 1, p. 17632, Sep. 2020, doi: 10.17632/JXPFC64KR.1.
- [27] E. Shao, "Encoding IP Address as a Feature for Network Intrusion Detection," *Doctoral dissertation, Purdue University Graduate*




- School*, p. 64, 2019.
- [28] S. Harun and M. F. Ibrahim, "A genetic algorithm based task scheduling system for logistics service robots," *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 1, pp. 206–213, Mar. 2019, doi: 10.11591/EEI.V8I1.1437.
- [29] S. N. Anual *et al.*, "Ga-based optimisation of a lidar feedback autonomous mobile robot navigation system," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 3, pp. 433–441, Sep. 2018, doi: 10.11591/eei.v7i3.1275.
- [30] F. Itano, M. A. de A. de Sousa, and E. Del-Moral-Hernandez, "Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8, doi: 10.1109/IJCNN.2018.8489520.
- [31] S. Lessmann, R. Stahlbock, and S. Crone, "Optimizing Hyperparameters of Support Vector Machines by Genetic Algorithms," in *IC-AI*, Jan. 2005, pp. 74–82.
- [32] R. Elshawi, M. Maher, and S. Sakr, "Automated Machine Learning: State-of-The-Art and Open Challenges," 2019, doi: 10.48550/arXiv.1906.02287. [Online]. Available: <http://arxiv.org/abs/1906.02287>.
- [33] A. Gogna and A. Tayal, "Metaheuristics: review and application," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 25, no. 4, pp. 503–526, Dec. 2013, doi: 10.1080/0952813X.2013.782347.
- [34] F. M. De Rainville, F. A. Fortin, M. A. Gardner, M. Parizeau, and C. Gagné, "DEAP: A Python framework for Evolutionary Algorithms," *Proceedings of the 14th annual conference companion on Genetic and evolutionary computation*, pp. 85–92, 2012, doi: 10.1145/2330784.2330799.
- [35] M. M. Saad, N. Jamil, and R. Hamzah, "Evaluation of support vector machine and decision tree for emotion recognition of malay folklores," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 3, pp. 479–486, Sep. 2018, doi: 10.11591/eei.v7i3.1279.
- [36] C. Zafer, "Fusing fine-tuned deep features for recognizing different tympanic membranes," *Biocybernetics and Biomedical Engineering*, vol. 40, no. 1, pp. 40–51, 2020, doi: <https://doi.org/10.1016/j.bbe.2019.11.001>.
- [37] M. Ramasamy and P. V. Eric, "An improved deep bagging convolutional neural network classifier for efficient intrusion detection system," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 1, pp. 405–413, Feb. 2022, doi: 10.11591/eei.v11i1.3252.
- [38] A. Fadlil, I. Riadi, and S. Aji, "Review of detection DDOS attack detection using naive bayes classifier for network forensics," *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 2, pp. 140–148, Jun. 01, 2017, doi: 10.11591/eei.v6i2.605.

BIOGRAPHIES OF AUTHORS



Hasan Kamel    received his Bachelor degree in Computer Engineering, from Al Mustansiriyah University, Iraq in 2018. His interests involve computer networks and artificial intelligence. He can be contacted at email: egma003@uomustansiriyah.edu.iq.



Mahmood Zaki Abdullah    is one of an associate professor doctor at Mustansiriya University Baghdad-Iraq. He holds a Ph.D degree in computer engineering and has many books in computer engineering, networks, smart systems and information technology. He can be contacted at email: drmzaali@uomustansiriyah.edu.iq.