



Article

Comparison of ML/DL Approaches for Detecting DDoS Attacks in SDN

Tariq Emad Ali [†], Yung-Wey Chong ^{*,†} and Selvakumar Manickam [†]

National Advanced IPv6 Centre, Universiti Sains Malaysia, Gelugor 11800, Malaysia

* Correspondence: chong@usm.my

† These authors contributed equally to this work.

Abstract: Software-defined networking (SDN) presents novel security and privacy risks, including distributed denial-of-service (DDoS) attacks. In response to these threats, machine learning (ML) and deep learning (DL) have emerged as effective approaches for quickly identifying and mitigating anomalies. To this end, this research employs various classification methods, including support vector machines (SVMs), K-nearest neighbors (KNNs), decision trees (DTs), multiple layer perceptron (MLP), and convolutional neural networks (CNNs), and compares their performance. CNN exhibits the highest train accuracy at 97.808%, yet the lowest prediction accuracy at 90.08%. In contrast, SVM demonstrates the highest prediction accuracy of 95.5%. As such, an SVM-based DDoS detection model shows superior performance. This comparative analysis offers a valuable insight into the development of efficient and accurate techniques for detecting DDoS attacks in SDN environments with less complexity and time.

Keywords: SDN; support vector machine; K-nearest neighbors; decision trees; multiple layer perceptron; convolutional neural network



Citation: Ali, T.E.; Chong, Y.-W.; Manickam, S. Comparison of ML/DL Approaches for Detecting DDoS Attacks in SDN. *Appl. Sci.* **2023**, *13*, 3033. <https://doi.org/10.3390/app13053033>

Academic Editor: Arcangelo Castiglione

Received: 19 January 2023

Revised: 17 February 2023

Accepted: 21 February 2023

Published: 27 February 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The innovative architecture of software-defined networking (SDN) has generated significant interest, as it provides a possible solution for managing rapidly expanding network topologies. By separating the data and control planes, the administrator's flexibility and elasticity are optimized. The application layer, which is the upper layer, prepares all the rules and regulations that are established by the administrator, and which can be dynamically altered by the SDN controller. SDN eliminates the need for license restrictions, and it allows network managers to create customized applications to manage network equipment. The control layer of SDN acts as the topology's intelligence by issuing commands to the infrastructure layer, which then implements these commands. In contrast, the infrastructure layer has no intelligence and only executes commands that it receives from the controller (CO) [1].

SDN provides ease in building, deploying, and maintaining a network, while maintaining network security by providing access without revealing the underlying layer details. It also offers flexibility in upgrading network functionalities by upgrading SDN apps. Hence, network hardware devices require only basic features [2,3]. DDoS attacks pose a significant challenge due to their ease of execution and difficulty in tracking, mainly because of IP spoofing [4]. These attacks can be classified into application layer attacks, protocol attacks, or volumetric attacks. Attackers typically aim to exhaust the server's bandwidth and CPU resources before executing an application layer assault to disrupt services. In the case of a protocol attack, the attacker depletes the target's available state using the protocol's processes, such as the TCP three-way handshake process. To launch a volumetric assault, the attacker floods the target with malicious traffic, resulting in resource depletion, including bandwidth and CPU usage [5].

In line with traditional networks, DDoS attacks have the potential to impact a switch's processing power or the connections between switches in the SDN data layer. This article focuses on volumetric DDoS attacks, which are the most common and long-term threat due to the rapid growth of network devices connected to the internet. To achieve early detection, ML/DL is employed to analyze the behavior, which is a new research area in the SDN, as machine response is faster than human response. Using ML/DL techniques, computers can learn from experience, analyze data attributes, or utilize trial-and-error procedures to make judgments. They serve as a supplementary method to assist network operators in understanding the situation. While a single threshold may theoretically identify flooding DDoS attacks, it may be difficult to differentiate normal burst traffic from attack traffic. One way to increase accuracy is to establish a DDoS attack definition after the threshold has been reached several times in a row. This approach allows for burst traffic while also delaying detection. Prior to defining a flooding DDoS attack, it may be possible to observe or analyze network traffic behavior to improve accuracy and hasten detection [6].

DoS/DDoS attacks pose a serious threat to SDNs, as they can cause congestion in the control plane, data plane, or control plane bandwidth, and may even bring down the entire network by targeting the network's brain—the controller. DoS/DDoS attacks on the SDN data plane can exhaust the OpenFlow switch's flow table RAM, resulting in discarded packets, and create new flows that do not match the switch's flow table entries, leading to a buffer overflow. These issues can cause delays and increase the amount of communication bandwidth required. To address these challenges, it is crucial to develop a solution that can protect against DoS/DDoS attacks while still enabling the forwarding of legitimate traffic, ensuring the proper handling of packet-in messages, and securing the communication channel between the OpenFlow switches and the SDN controller.

This article contributes to the field of intrusion detection by estimating and investigating the projected model using ICIDS2017 and CICDDoS2019 datasets. By training and testing the model with various types of datasets and comparing the results with past research, the impact of these data on the model detection behavior can be understood, and the kinds of features that would improve model performance can be identified. As a result of these analyses, it will be possible to create unique features and feed the model for training. Additionally, this article provides a summary of the output, analyzing and comparing each ML/DL approach's accuracy performance in detail. The deployment of the ML/DL overhead is also illustrated by examining the ML/DL classifier's time process for both learning and prediction. This compiled summary enables the proposal of an adaptive mechanism composed of a combination of ML/DL algorithms to enhance the model detection process with high performance, low-time detection, and low complexity.

The results of the tests indicate that (i) the data-preprocessing (DP) and ML/DL system can be applied in a single SDN-CO for estimating DDoS attacks, (ii) an ML/DL system is capable of detecting DDoS attacks with a selected input, (iii) the presence of odd instances in the learning set has little effect on the estimate's accuracy for regular valid data, and (iv) incorporating more information and characteristics in the learning process can improve results, but overfitting may occur. Furthermore, the testing of the ML/DL recognition system in a physical SDN network is feasible.

2. Related Work

Previous publications have explored the use of machine learning (ML) and deep learning (DL) algorithms to identify distributed denial of service (DDoS) attacks. These methods typically rely on labeled data to train a predictor, which can guide decision making when deployed in a network. Before implementation, network managers can validate the effectiveness of the ML/DL algorithm using known test samples to obtain confidence in the predictor's performance. However, despite the existing literature on the topic, there are still gaps in the research on ML/DL applications for DDoS attack detection. Table 1 presents a comparison of the suggested model with the existing literature.

Table 1. Existing ML/DL methods to defend SDN from DDoS assaults.

Ref.	ML/DL	Exper.	Dataset	Features	Remarks
[7]	NB,SVM,NN	Mininet	Real-time	Host connections per second	In the SDN, SVM may be used as an IDS.
[8]	DT	VM	Self-generated	flag, TTL, and src/dst IP	A detection time of 0.6 s and accuracy of 98% for DDoS attacks.
[9]	KNN,DT, NN	A real testbed	CAIDA 2007	Ports and ICMP packets per IP	A minimally intrusive defense with a flexible monitor window period. The precision may exceed 98%.
[10]	SVM,KNN, RF	Mininet	NSL-KDD	Unmentioned specifics	SVM model improvement. Nearly 99% of the time, it is accurate.
[11]	DA,SVM, KNN,NB,DT	VM	ISCX data	Number of bytes and packets, flowduration (FD), NumberOfBytes over NumberOfPackets	Although ML/DL may be used in the SDN as a classifier, choosing the right characteristics for improved detection can be challenging.
[12]	SVM, KNN, NN, NB	VM	Self-generated	12 features	The detection accuracy can be increased by filtering important features via feature selection prior to training.
[13]	SKNN, NB, SVM	Mininet	Self-generated	Length, duration, size, and speed of the flow rate	The basic KNN model may be enhanced by adding a weight value to the neighbors. From simulation, the efficiency is astounding.
[14]	NB	Real testbed	NSL-KDD	25 features selected	The proposed strategy for DDoS mitigation has yielded promising results among ISPs.
[15]	Soft-max, NN, Autoencoder	Real testbed	Self-generated	34 features selected	Individual DDoS attack detection accuracy can exceed 95%, and attack categorization accuracy can exceed 99%.
[16]	SVM	Real testbed	KDD 1999, KDD-CUP 1999	30 features selected	The amount of characteristics used to categorize traffic affects accuracy.

Meti et al. [7] proposed a dataset comprised solely of TCP stream-traffic generated by the actual network and employed dual features to train their models, including naive Bayes (NB), support vector machine (SVM), and neural networks (NNs), with regular and irregular tags. Their comparisons indicate that NN achieved the highest accuracy (A) and precision (P) with 79.9% and 99.95%, respectively, while SVM showed a better recall value (R) of 79.99% compared to NB and NN.

Zekri et al. [8] suggested using a decision tree (DT) method to identify DDoS attacks in a cloud network. The DT selection criteria identified qualities with the highest improvement ratio, and the dataset was split accordingly. The process was repeated until no more branches were possible, and the DT was prepared to classify incoming data after learning from the training set traffic. The traffic was classified into four groups, and the model achieved over 98% accuracy in test results with a processing time of only 0.58% of a second.

Tuan et al. [9] employed entropy and logarithm values to identify TCP-SYN/ICMP flood attacks in software-defined networks (SDNs). They utilized the K-nearest neighbors (KNN) algorithm to locate the K closest entropy metrics, and the existing distance points were examined to determine whether DDoS attacks were being launched against the network. The CAIDA2007 dataset was used in this study, and the accuracy of the Bonesi test was more than 99% when K was set to 9.

Sahoo et al. [10] proposed a better support vector machine (SVM) model using genetic algorithms (GAs) and kernel principal component analysis (KPCA) to detect DDoS attacks.

This model regularly obtained flow information from switches using the SDN controller and extracted key features using kernel analysis. The SVM parameters were then modified using genetic algorithms to make the best possible predictions. The accuracy of the proposed model was evaluated using simulations and public datasets, and it achieved close to 99% accuracy.

Bakker et al. [11] compared seven classifiers in terms of initialization times and accuracy and found that SVM had the highest accuracy (93%) with the lowest detection time.

Polat et al. [12] focused on feature selection techniques and compared four ML algorithms. They found that KNN, a wrapper-based algorithm, performed the best with 98.3% accuracy when using six essential factors for selection. Both studies provide useful insights into the effectiveness of different ML/DL methods for DDoS attack recognition.

In a network system, it is possible to locate the ML/DL component on a remote network instead of an application over the controller [13]. Mohammed et al. [14] connected four ISPs in a setup scenario and considered the use of an ML server that employs NB classification outside the network. Upon receiving a fresh incoming packet, the controller extracts the features, and then the ML server examines it. If the CO receives an anomalous estimate outcome from the ML model, it will notify the CO to deny these packets to prevent network attacks. The ML server selects 25 traffic characteristics to learn from, but this is decreased to 18 due to poor real traffic processing. However, the accuracy in the normal condition is less than 70%, and 92% in the aberrant condition, which is not very encouraging. To analyze network traffic, Niyaz et al. [15] proposed running three modules, namely TCFI, FE, and TC, over the controller. TCFI stores packet headers for FE in addition to replies to fresh flow requests. The FE calculates the (mean, entropy) values from the packet characteristics to categorize traffic into one of eight classes, including seven types of DDoS attacks, and recalls upon the TC. According to test results, the model's accuracy is over 95% for each type of DDoS attack and over 98.5% for a binary class that only includes attack and normal states. Wang et al. [16] used SVM in combination with the sFlow toolset to detect threats to SDN. The model pulls behavioral variables for SVM learning from traffic information gathered from switches via the controller. The purpose of the examination is to discover the behavioral traits of recognized assaults. SVM can identify an attack by contrasting normal and malicious behavioral profiles. The outcomes revealed that the accuracy (A) is 96.8%.

In recent years, there have been several studies focused on comparison techniques for detecting distributed denial-of-service (DDoS) attacks, as evidenced by research papers [7,9–13,15]. However, these studies have certain limitations that require further research. Specifically, many comparison methods evaluate only various machine learning (ML) or deep learning (DL) algorithms using simulation offline analysis, which poses a challenge for creating and assessing the performance of one's own ML–DL model combination. Moreover, some approaches concentrate on assessing the performance of the training model without taking into account the performance of the prediction model, which is crucial for assessing how the algorithm behaves with new data. Another limitation is that most studies evaluate the model's performance based on accuracy, F1-score, and other metrics without considering the time taken for detection during training or prediction, which is an important factor for real-time detection. Lastly, the majority of research on this topic evaluated the model's performance using default parameter values and without considering the type of kernel, thereby missing opportunities to explore how these parameters and kernel types impact the algorithm's behavior on both linear and non-linear datasets.

3. Method

Five ML/DL methods were evaluated using the (CICIDS2017 and CICDDoS2019) datasets in this study. The actions taken are depicted in Figure 1. Model was created using ML/DL techniques, as shown in the diagram. For the comparison, significant performance metrics (train accuracy, prediction accuracy, F1-score, Matthews correlation coefficient (MCC), training time, and predict time) were employed.

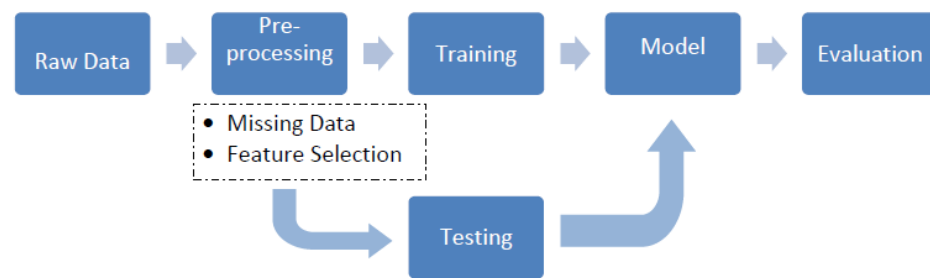


Figure 1. Process flow diagram.

3.1. Dataset (DS)

For network security and intrusion detection, we utilized the CICIDS2017 dataset, which was developed by the Canadian Institute for Cybersecurity (CIC). This dataset comprises 79 features and covers 6 distinct attack methods. However, in this study, we only used data for one day (Tuesday) that included both normal activity and attacks, totaling 11 G of traffic. This attack can be classified as a DDoS assault, which renders the network unusable for legitimate purposes due to system overload [17]. Moreover, we employed the CIC-DDoS2019 public dataset, which contains the latest DDoS attacks and reflects actual real-world statistics. This dataset includes 84 features and encompasses 12 different attack methods, along with the outcomes of network traffic analysis. The training process was conducted using Jupyter Notebook 6.4.12 and Python 8.4.0 on a Windows 10 laptop equipped with an Intel(R) Core(TM) i5-4200U CPU @ 1.60 GHz 2.30 GHz, 12 GB RAM, and Intel HD Graphics 3000.

3.2. Pre-Processing

Pre-processing is a crucial step in data analysis as raw data often contain flows that are incomplete, have missing or duplicated values, and exhibit irregularities. Pre-processing aims to provide a cleaner dataset, which can facilitate the construction and training of an ML/DL model with reduced errors and enhanced performance [18].

3.2.1. Missing Data

Missing data is a common issue in raw data, and it can take different forms, such as null values that are not compatible with the data format. For instance, a numerical feature should not contain symbols or alphabetic letters. The most straightforward way to handle such data is to discard them completely. However, in some cases, missing data points can be valuable, and therefore, we use the maximum likelihood estimation approach to estimate the missing values [19]. In this study, we used linear interpolation to fill in the gaps in the data. During this process, we identified missing data for two features in four observations, which were Flow Bytes/s and Flow Packets/s.

3.2.2. Feature Selection

For the purpose of our study, all features except for the mean value were removed from the dataset. The eliminated features include Fwd Packet Length (FPL), Bwd Packet Length (BPL), Flow IAT (FIAT), Packet Length (PL), Fwd IAT Total, Bwd IAT Total, and Packet Length Variance (PLV). Additionally, to improve the algorithm's performance, we also removed features such as FlowID (FID), SourceIP (SrcIP), SourcePort (SrcP), DestinationIP (DesIP), DestinationPort (DesP), Protocol, and Timestamp (TS). This resulted in a final dataset of 50 features, as opposed to the original 84 features. The label or tag datum was the feature "Label". However, during the pre-processing stage, issues were identified with three methods, including KNN, MLP, and SVM. Further investigation revealed that dual characteristics (FlowBytes/s (FB) and FlowPackets/s (FP)) contained values of 'Intf'. Therefore, when these three algorithms were performed, these two features were replaced by zero.

3.3. ML/DL Classifiers

Five ML/DL algorithms are used to train/test organized DS.

3.3.1. Support Vector Machine (SVM)

SVM locates the ideal hyper-plane that divides binary categories by greatest separation among their margin line. The support vector is made up of these boundary points. As a result, it is memory efficient and effective for high-dimensional space issues. However, this strategy will perform only at a mediocre level if the number of features exceeds the amount of samples [20]. A number of patterns that belong to the +1 and −1 classes are displayed in Figure 2. Red (squares) are used to depict joined patterns in class 1, whereas yellow is used to represent joined patterns in class +1 (circles).

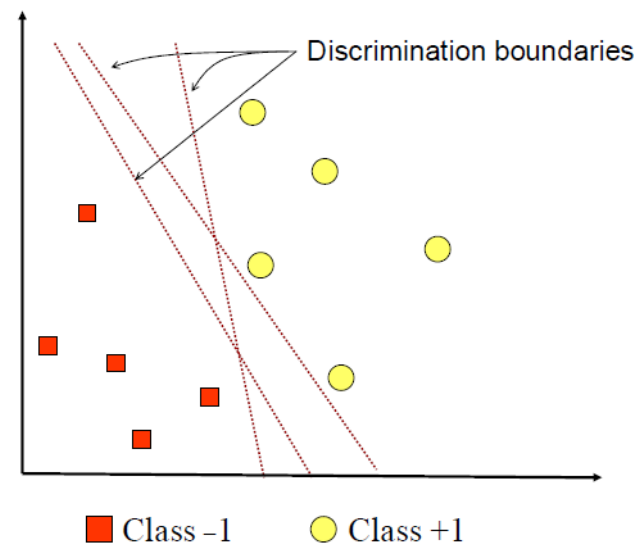


Figure 2. Hyperplane SVM.

Lagrange multiplier (LM) is one of the computational methods that may be used to address this problem:

$$L(w, s, n) = \frac{1}{2} \times \|w\|^2 + \sum_i n_i \left(B_i \left(w^T y_i + s \right) - 1 \right) \quad (1)$$

n_i is the LM and can be zero value or positive value ($n_i > 0$). The best rate of the calculation can be considered by minimizing (L) compared to (w,s) and top (L) against n_i .

3.3.2. K-Nearest Neighbor (KNN)

A technique that makes use of the supervised algorithm is the KNN algorithm [21]. Instance-based learning groups are included in KNN. Its method is straightforward and determines the KNN by comparing how similar the test sample is to the training sample [22]. Finding groupings of (k) items in the learning data that are the nearest to the item in the fresh data is known as KNN [23]. KNN is a straightforward classification method, yet it performs well [24]. Typically, the formula of distance is applied as shown in Equation (2) to determine the separation between two x and y objects:

$$d_{xy} = \sqrt{\sum_{i=1}^n (X_i - Y_i)^2} \quad (2)$$

KNN provides an advantage, such as the capability to train with noisy data and effectiveness with huge training sets [25].

3.3.3. Decision Tree (DT)

The decision tree algorithm is a tree structure that follows a set of rules to determine a class or value, where the root node represents the strongest predictor, and subsequent predictors are reached through stem nodes. The algorithm ultimately arrives at leaf nodes, which typically represent a decision or classification [26]. Despite their usefulness, decision trees have several drawbacks, including instability resulting from even small changes in the data, which can lead to a significant change in the structure of the tree. Additionally, when compared to other algorithms, decision tree calculations can occasionally become significantly more complicated.

3.3.4. Multilayer Perceptron (MLP)

A multilayer artificial neuron network is a crucial part of DL. Multi-layer perception is referred to as “MLP” [27]. It is composed of substantial, intricately interconnected layers, which might transform any incoming length into the shape desired. An MLP is a NN that has several layers. The MLP connect neurons in such a way that its outcomes are also their input in order to construct a neural network. The MLP is depicted in Figure 3.

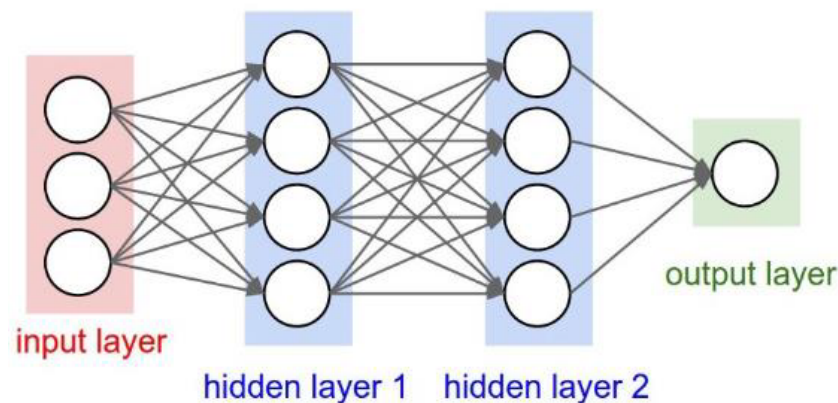


Figure 3. Multi-layer perceptron (MLP) diagram.

3.3.5. Convolutional Neural Network (CNN)

CNN is a DL system that can recognize different items and attributes in an image based on trainable biases and weights. Compared to other classification algorithms, a ConvNet requires significantly less pre-processing [28]. Contrary to fundamental approaches, in which restrictions must be custom, ConvNets can explore all such restrictions and attributes. The organization of the visual cortex and the connectivity network of neurons in the human brain both have an impact on the design of a ConvNet. Artificial neurons only react to signals in this restricted region of the visual field, known as the observe and analyze. There are several overlapping areas like this that make up the whole visual field. Figure 4 shows the CNN process.

The non-image converted to an image data step is important in applicable CNN architectures. The concept is very simple: Instead of operating the extraction of features and choice for samples taken, we choose to figure out how to group connected or correlated features into neighboring areas of a two-dimensional (2d) feature space (d features \times S samples) to make learning more about intricate interactions and relationships among different features extremely easy (S samples \times d features). As a friendly representation of samples to CNNs, we could theoretically convert any type of non-image data into feature map images using this general approach. CNNs offer several unique advantages over other neural network architectures, such as automated feature extraction from raw features and memory-footprint reduction through effective weight sharing. The essential stages are depicted in Figure 5. The quality of the output feature image, which is an import and export among the degree of image compression and the required hardware resources (such as RAM, GPU, HD, etc.) affects the proportion of feature overlapping.

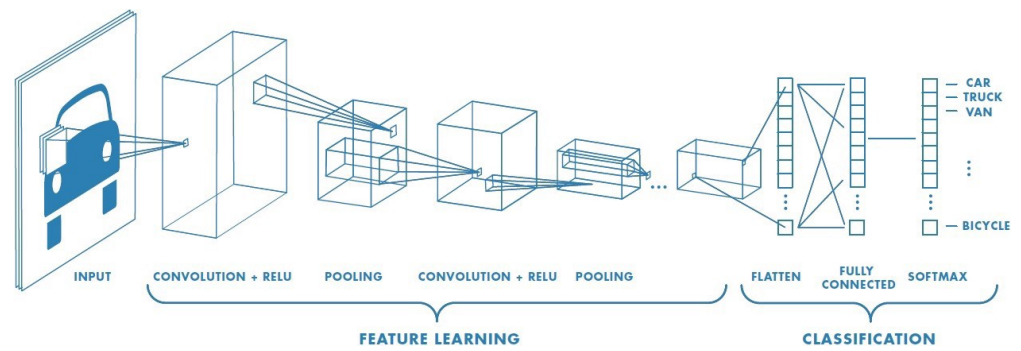


Figure 4. CNN classification sequence <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, accessed on 1 December 2022.

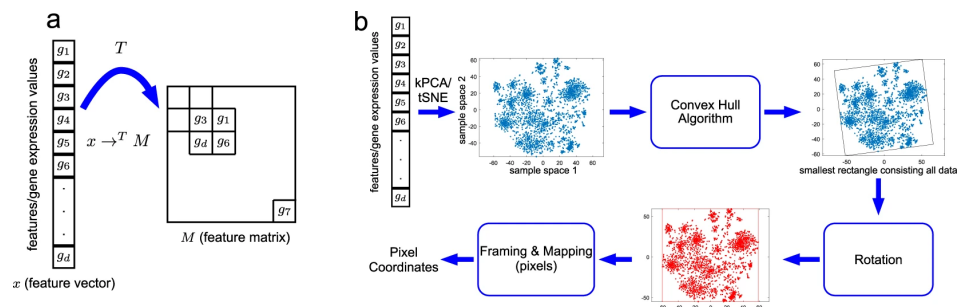


Figure 5. Non-image data to images <https://www.nature.com/articles/s41598-019-47765-6>, accessed on 1 December 2022. (a) An example of the conversion of a feature vector to a feature matrix (b) An example of the DeepInsight process for converting a feature vector to picture pixels.

4. Measures of Performance

The four major metrics for measuring the effectiveness of an ML/DL system are TruePositive (TP), TrueNegative (TN), FalsePositive (FP), and FalseNegative (FN). Positive in our test relates to attack states, whereas negative denotes normal states. The performance of each ML/DL approach is evaluated using these measures to derive the following indicators [29]:

- Accuracy (A) measures how well it predicts the future, encompassing both positive and negative outcomes:

$$\text{Accuracy (A)} = \left(\frac{TP + TN}{TP + TN + FP + FN} \right) \times 100\% \quad (3)$$

- Precision (P) is the proportion of all normal state predictions that are right for a given network state:

$$\text{Precision (P)} = \left(\frac{TP}{TP + FN} \right) \times 100\% \quad (4)$$

- The accuracy and recall levels are measured using the F-Measure (F):

$$\text{F-Measure (F)} = \frac{2 \times (\text{Precision} \times \text{Recall})}{(\text{Precision} + \text{Recall})} \quad (5)$$

- The Matthews correlation coefficient (MCC) is a contingency matrix technique of generating the Pearson product-moment correlation coefficient between actual and expected values. This alternative measure is unaffected by the problem of imbalanced datasets:

$$\text{MCC} = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FN) \times (TN + FP) \times (TN + FN) \times (TP + FP)}} \quad (6)$$

The confusion matrix (CM) is a highly well-liked metric for classifying issues. Both binary classification and multiclass classification issues may be solved with it [30]. The CM estimates from actual and expected values. The “TN” finding counts the set of accurately detected negatively classified instances. Related to it, “TP” signifies the rate of accurately detected positive instances. The letter “P” stand for the ratio of genuine negative instances which have been misclassified as positive, while “FN” stands for the ratio of genuine positive samples which have been misclassified as negative. An important metric most frequently employed in categorization is accuracy. Alternative measures focused on CM are also relevant for evaluating because accuracy may be deceiving once implemented to imbalanced ds. To obtain the CM, use the “confusion matrix ()” function from the Py “sklearn” package (<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusionmatrix.html>, accessed on 5 Novemebr 2022). Import this function into Python by typing “from sklearn.metrics import confusion matrix”. To produce the CM, it must run the function with both the real and predictable values [31].

In this study, the CM was used to assess how well the procedures were employed after the categorization was performed. Figure 6 depicts the confusion matrix’s layout for binary classification.

		Predicted Class	
		Negative Normal	Positive Attack
Actual Class	Negative Normal	<i>True Negative (TN)</i>	<i>False Positive (FP)</i>
	Positive Attack	<i>False Negative (FN)</i>	<i>True Positive (TP)</i>

Figure 6. CM for binary classification.

5. Discussion and Result

It appears that using the different datasets (CICIDS2017 and CICDDoS2019) gives approximately the same values, and it can be noted that there is a light improvement in the performance values by using the CICDDoS2019 dataset. Additionally, we can note that SVM and MLP have better accuracy performance in both training and prediction steps. In contrast, CNN has a big difference in the performance of accuracy between training and prediction, and this is because CNN needs a large dataset to feed the model to increase the prediction accuracy rate. According to DT, it has an acceptable accuracy rate, but instability can result from a bit of change in the data, leading to a substantial change in the DT structure. When compared to other algorithms, DT calculations can occasionally become significantly more complicated. Additionally, we can note that SVM has better performance in terms of F1-score, which means the SVM classifier works best because it has higher precision and recall compared with other algorithms. Finally, in terms of the training and prediction time, we can note that CNN has the highest time in (reprocessing, training, and prediction) phases and this goes back to CNN working with image data, so we need to transform the non-image data to image data. This takes a long time during our test in both training and prediction also because it uses auto-select features, which also take time during the process, as well as going back to using multiple convolution layers and then inserting the data to flatten the input layer, multiple hidden layers, and then the output layer to build the model. Figures 7 and 8 show the accuracy and time for our test result for both the training and prediction model.

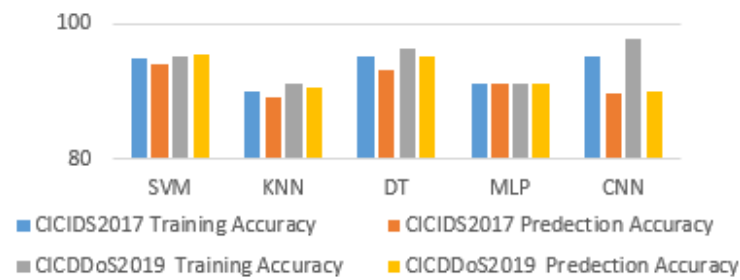


Figure 7. Models' accuracy result.

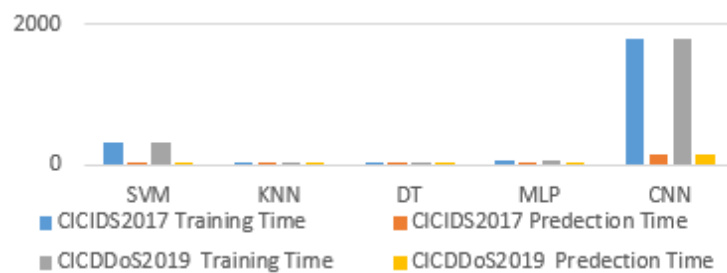


Figure 8. Models' time result.

Also, Figures 9 and 10 show the result of MCC and F1-Score by using both CICIDS2017 and CICDDoS2019 datasets.

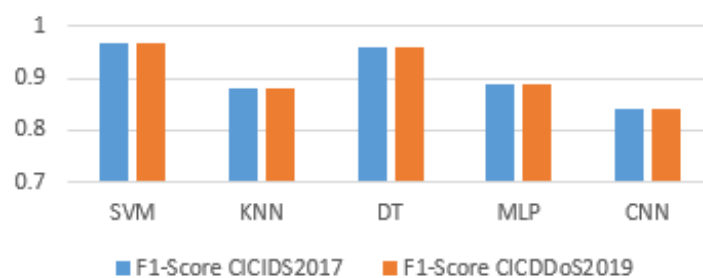


Figure 9. Models' F1-score result.

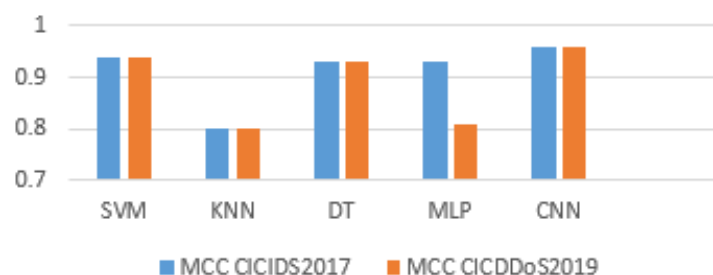


Figure 10. Models' MCC result.

The outcomes of our employment of several ML/DL algorithms are displayed in Tables 2 and 3 for both the CICIDS2017 and CICDDoS2019 datasets.

In both tables, there is not a big difference in the results when using different datasets. Additionally, we can note that CNN has the highest training accuracy (97.81%), while the performance of prediction accuracy is the lowest (90.09%). Additionally, the SVM has a very similar accuracy rate in both training and prediction (95.08% for training and 95.57% for prediction), as well as the MLP (91.22% for training and 91.13% for prediction). In addition, in both tables, we can note that KNN has the lowest accuracy in both training and prediction

(90.41% for training and 89.16% for prediction). According to the training and prediction times, we can note that CNN takes a long time in both training and prediction steps, while DT is a lower value. According to the F1-score, we can note that SVM has a higher value (0.97), while CNN has the lowest value (0.84).

Table 2. Findings from the five classification algorithms applied to the CICIDS2017 dataset.

Algorithm	Train Accuracy %	Prediction Accuracy %	F1-Score	MCC	Training Time (s)	Predict Time (s)	Tuning Parameter
SVM	94.86	94.01	0.97	0.94	316.36	54.83	kernel = 'liner', C = 1, gamma = 1, random_state = 0
KNN	90.14	89.16	0.88	0.802	0.033	9.46	n_neighbors = 161, p = 2, metric = 'euclidean'
DT	95.33	93.14	0.96	0.93	0.45	0.006	max_depth = 5, random_state = 0
MLP	91.21	91.11	0.89	0.81	59.26	0.026	random_state = 0, max_iter = 200, alpha = 1
CNN	95.18	89.61	0.84	0.96	1800.59	157.22	Conv2D(64,(3,3), strides = (2,2), padding = 'same', input_shape(2,3,3), activation = 'relu'

Table 3. Findings from the five classification algorithms applied to the CICDDoS2019 dataset.

Algorithm	Train Accuracy %	Prediction Accuracy %	F1-Score	MCC	Training Time (s)	Predict Time (s)	Tuning Parameter
SVM	95.08	95.57	0.97	0.94	316.36	54.83	kernel = 'liner', C = 1, gamma = 1, random_state = 0
KNN	91.23	90.61	0.88	0.802	0.033	9.46	n_neighbors = 161, p = 2, metric = 'euclidean'
DT	96.43	95.31	0.96	0.93	0.45	0.006	max_depth = 5, random_state = 0
MLP	91.22	91.13	0.89	0.81	59.26	0.026	random_state = 0, max_iter = 200, alpha = 1
CNN	97.81	90.09	0.84	0.96	1800.59	157.22	Conv2D(64,(3,3), strides = (2,2), padding = 'same', input_shape(2,3,3), activation = 'relu'

Tables 4–7 lists a comparison between our study result and the previous literature study using various datasets. It is important to note that each work's performance is based on a unique dataset with distinct features, so the goal of the comparison between the various prior reviews of the literature and our result study is to show which output, when using particular training, a prediction strategy, a unique dataset and testbed hardware, produces the best results. As we can see, in the training stage, the SVM from the literature study has the greatest accuracy (94.59%) in [16]; however, in our test, it has 95.08%. Additionally, only the prediction stage in [11] was computed with 93.4%, although in our test, it had 95.57%. The best result for the F1-score is 0.9142 in [13]; however, it is 0.97 in our test. Additionally, only [10] discovered it with 1562.75 s of training time, but in our simulated test, it took 316.36 s.

According to a survey of the literature, the greatest KNN accuracy and training time are 89.21% and 0.848 s, respectively, in [9], while the F1-score has 0.848 in [12]. The statistics for training accuracy (91.23%), prediction accuracy (90.61%), F1-score (0.88), and training time (0.033 s) are higher in our test. The greatest F1-score and accuracy for DT were (94.8%) and (0.9051) concurrently in [8], and training time (0.505 s) in [9]. The numbers for training

accuracy (96.43%), prediction accuracy (95.31%), F1-Score (0.96), and training time (0.45 s) are superior in our test.

Table 4. Training accuracy comparison between other literature studies and our study result.

Algo.	Ref. [8]	Ref. [9]	Ref. [10]	Ref. [11]	Ref. [12]	Ref. [13]	Ref. [14]	Ref. [15]	Ref. [16]	Ref. [17]	Our Test
SVM	80.12	X	X	94.417	X	92.46	X	X	X	94.59	95.08
KNN	X	X	89.21	88.3	X	89.1	X	X	X	X	91.23
DT	X	94.8	94.15	X	X	X	X	X	X	X	96.43

Table 5. Prediction accuracy comparison between other literature studies and our study result.

Algo.	Ref. [8]	Ref. [9]	Ref. [10]	Ref. [11]	Ref. [12]	Ref. [13]	Ref. [14]	Ref. [15]	Ref. [16]	Ref. [17]	Our Test
SVM	X	X	X	X	93.468	X	X	X	X	X	95.57
KNN	X	X	X	X	88.575	X	X	X	X	X	90.61

Table 6. F1-score comparison between other literature studies and our study result.

Algo.	Ref. [8]	Ref. [9]	Ref. [10]	Ref. [11]	Ref. [12]	Ref. [13]	Ref. [14]	Ref. [15]	Ref. [16]	Ref. [17]	Our Test
SVM	0.8	X	X	0.833	X	0.906	0.914	X	X	X	0.97
KNN	X	X	0.807	X	X	0.848	0.814	X	X	X	0.88
DT	X	0.905	0.902	X	X	X	X	X	X	X	0.96

Table 7. Training time comparison between other literature studies and our study result.

Algo.	Ref. [8]	Ref. [9]	Ref. [10]	Ref. [11]	Ref. [12]	Ref. [13]	Ref. [14]	Ref. [15]	Ref. [16]	Ref. [17]	Our Test
SVM	X	X	X	1563	X	X	X	X	X	X	316.36
KNN	X	X	0.411	9.091	X	X	X	X	X	X	0.033
DT	X	X	0.505	X	X	X	X	X	X	X	0.45

As a result of this comparison between our testbed and the testbed used in the literature review, even though the dataset and test environment are different, our test in this study still achieves a high level of accuracy, F1-score, and training time when compared to the results from the prior literature. This is due to synchronization, algorithmic tuning parameters, and the type of dataset that we used when training the model, as well as other factors. Additionally, we take into account in our simulation the prediction side because [11] is the only literature study that mentions this computation side. In order to evaluate the model's effectiveness in the speed detection process, we computed the accuracy as well as the timing for both training and prediction—calculations that were not included in the previous literature reviews. As a result, we included time calculations in our analysis. Additionally, we took into account the Matthews correlation coefficient (MCC), which generates a score that is more enlightening and accurate when analyzing binary classifications than accuracy and F1-score and that is also not found in any other literature study. Finally, we included DL scenarios, such as MLP and CNN in our test study, to examine how they interact with ML scenarios. Figure 11 shows the result comparison for accuracy and Figure 12 shows the result comparison for the F1-score. Finally, Figure 13 shows the result comparison for timing.

By comparing the results, we can see how the dataset type affects the training model's performance. We can also see how tuning parameters affect model performance, as well as how using different kernel functions in shallow architectures, such as SVMs, or deep architectures, such as multilayer kernel machines, allowed us to demonstrate their benefits and examine their impact on model performance. Finally, we can determine which approach had better performance with less complexity and time use from this as well. Thus, this study will serve as our first step in examining the effectiveness of each algorithm. From

the data gathered, we can then create our own ML-DL model that combines various ML techniques to achieve better performance in terms of accuracy, F1-score, MCC, and timing for both the training and prediction of models.

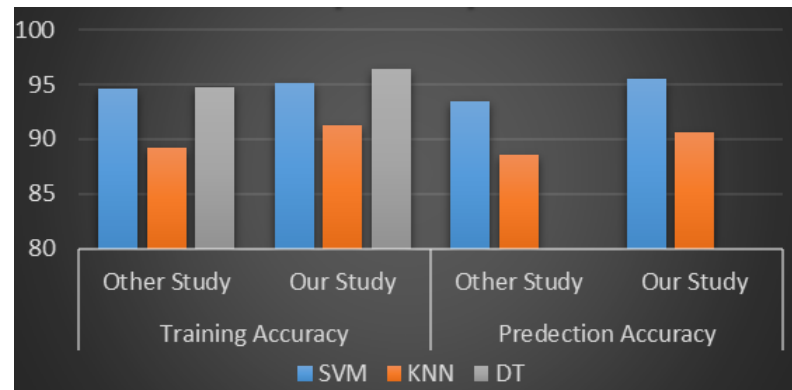


Figure 11. Models accuracy comparison result.

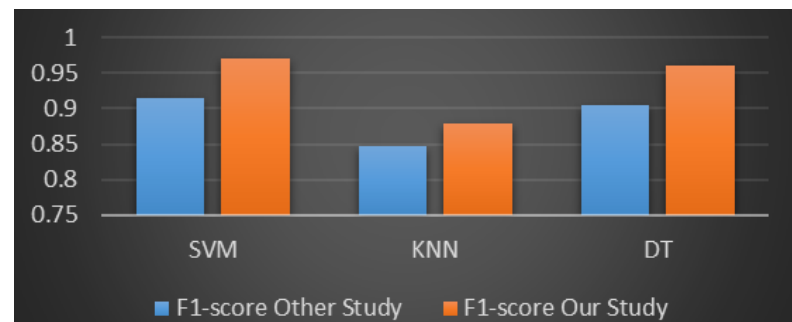


Figure 12. Models' F1-score comparison result.

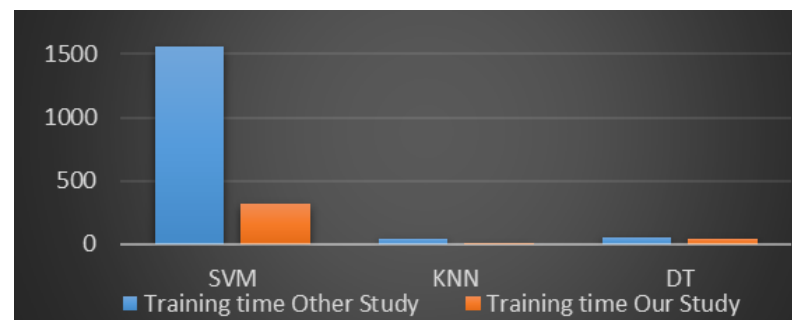


Figure 13. Models' timing comparison result.

6. Conclusions

SDN leads to network changes by redefining network administration and connectivity. Even though SDN has many advantages, it also introduces new security concerns that demand academics/industry. DDoS assaults can affect both the SDN controlplane as well as the dataplane. The identification of DDoS attacks is more difficult today than it was in the old network due to the complexity of traffic flow features. We may create a novel strategy for characterizing DDoS assaults in an elastic technique owing to a decoupled SDN architecture. To ensure the accuracy of the attack definition, the behavior of DDoS assaults in the SDN must be examined in novel contexts. ML/DL might be a creative approach that integrates with SDN as a novel development for assisting people in making choices athwart various domains. In this study, we used datasets (CICIDS2017 and CICDDoS2019) that cover several current attack types. The datasets include 79 features and 6 different types of attack. Five classification approaches (SVM, KNN, DT, MLP, and CNN) were used to train the datasets and test the results. In order to make classification easier, the processed

datasets contain varying percentages. According to the results of this study, among the different algorithms employed, CNN had the maximum train accuracy (A) (95.18%) and prediction accuracy (89.61%) using the CICIDS2017 dataset, and it had a training accuracy of 97.81% and prediction accuracy of 90.09% by using the CICDDoS2019 dataset, and it can be noted that in both datasets, there is a large difference between the training and prediction in terms of accuracy using CNN because the CNN needs a huge dataset for training to increase the accuracy of the prediction. Additionally, it can be noted that SVM achieves a good accuracy rate (i.e., accuracy rate between the training and prediction) in both datasets compared with CNN and other algorithms. Finally, it can be noted also that CNN has a high training and testing time compared with other algorithms, and this goes back to deep learning in general, and CNN specifically uses an image dataset to build the model, so converting the non-image data to image data adds an additional time process. So, this indicates that the SVM can correctly and simply detect DDoS assaults. So there is potential for improvement, and the model might be tuned to function more effectively in terms of accuracy rate and time for the training and testing model. Additionally, just one day's worth of DS from different days was used for this investigation.

As a consequence, it is clear from our test that we performed better than expected in terms of accuracy, F1-score, MCC, and timing when compared to earlier literature reviews. This improvement may be attributed to the kernel type, the device used to conduct our simulation, and the tuning parameter. Additionally, there is room for development, and the model may be adjusted to work more efficiently. Additionally, we can observe that the ML/DL model performs better when utilizing the CICDDoS2019 dataset since it has the most recent information on a wide variety of attacks. As a result, the model requires constant refeeding with UpToDate features to optimize its learning. As a result, it is possible to create one's own dataset in the future that includes various attack types and UpToDate characteristics to raise the degree of model learning. On the other hand, our study's limitation is that we were only able to analyze data from one day's worth of DS from several days. Therefore, it is crucial to use the entire dataset, which contains a total of six different types of assaults, in order to draw a clear conclusion. Additionally, simulation and offline analysis form the foundation of our work. Due to this, assessing the effectiveness of detection algorithms in real-time and on actual networks is challenging. In the future, we intend to build our own ML/DL mix model based on the test results gathered during this research and evaluate its performance for detecting DDoS attacks with actual SDN devices and actual traffic. It will be preferable to achieve real-time training so that the system is always kept up to date to improve the model detection process with high performance, low time detection, and low complexity.

Author Contributions: T.E.A., Y.-W.C. and S.M. contributed to the study's conception and design. The first draft of the manuscript was written by T.E.A., and T.E.A., Y.-W.C. and S.M. commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding: This research was funded by Universiti Sains Malaysia (USM) and National Advanced IPv6 Center (NAv6).

Institutional Review Board Statement: This article does not contain any studies with human participants or animals performed by any of the authors.

Informed Consent Statement: Informed consent was obtained from all individual participants included in the study.

Conflicts of Interest: The authors declare that they have no conflict of interest.

References

1. Ali, T.E.; Morad, A.H.; Abdala, M.A. Load balance in data center sdn networks. *Int. J. Electr. Comput. Eng.* **2018**, *8*, 3086–3092.
2. Ali, T.E.; Abdala, M.A.; Morad, A.H. SDN implementation in data center network. *J. Commun.* **2019**, *14*, 223–228. [[CrossRef](#)]
3. Ali, T.E.; Morad, A.H.; Abdala, M.A. Traffic management inside software-defined data center networking. *Bull. Electr. Eng. Inform.* **2020**, *9*, 2045–2054. [[CrossRef](#)]

4. Yin, D.; Zhang, L.; Yang, K. A DDoS attack detection and mitigation with software-defined Internet of Things framework. *IEEE Access* **2018**, *6*, 24694–24705. [\[CrossRef\]](#)
5. Zargar, S.T.; Joshi, J.; Tipper, D. A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Commun. Surveys Tutor.* **2013**, *15*, 2046–2069. [\[CrossRef\]](#)
6. B. Karan, D. Narayan, P. Hiremath, Detection of ddos attacks in software defined networks. In Proceedings of the 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 20–22 December 2018; pp. 265–270.
7. Meti, N.; Narayan, D.G.; Baligar, V.P. Detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In Proceedings of the IEEE Conference on Advances in Computing, Communications and Informatics, Udipi, India, 13–16 September 2017; pp. 1366–1371.
8. Zekri, M.; El Kafhali, S.; Aboutabit, N.; Saadi, Y. DDoS attack detection using machine learning techniques in cloud computing environments. In Proceedings of the IEEE Conference of Cloud Computing Technologies and Applications, Rabat, Morocco, 24–26 October 2017; pp. 1–7.
9. Tuan, N.N.; Hung, P.H.; Nghia, N.D.; Tho, N.V.; Phan, T.V.; Thanh, N.H. A ddos attack mitigation scheme in isp networks using machine learning based on sdn. *Electronics* **2020**, *9*, 413. [\[CrossRef\]](#)
10. Sahoo, K.S.; Tripathy, B.K.; Naik, K.; Ramasubbareddy, S.; Balusamy, B.; Khari, M.; Burgos, D. An evolutionary svm model for ddos attack detection in software defined networks. *IEEE Access* **2020**, *8*, 132502–132513. [\[CrossRef\]](#)
11. Bakker, J.N.; Ng, B.; Seah, W.K. Can machine learning techniques be effectively used in real networks against DDoS attacks? In Proceedings of the IEEE Conference on Computer Communication and Networks, Hangzhou, China, 30 July–2 August 2018.
12. Polat, H.; Polat, O.; Cetin, A. Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability* **2020**, *12*, 1035. [\[CrossRef\]](#)
13. Dong, S.; Sarem, M. Ddos attack detection method based on improved knn with the degree of ddos attack in software-defined networks. *IEEE Access* **2019**, *8*, 5039–5048. [\[CrossRef\]](#)
14. Mohammed, S.S.; Hussain, R.; Senko, O.; Bimaganbetov, B.; Lee, J.; Hussain, F.; Bhuiyan, M.Z.A. A new machine learning-based collaborative DDoS mitigation mechanism in software-defined network. In Proceedings of the IEEE Conference on Wireless and Mobile Computing, Networking and Communications, Limassol, Cyprus, 15–17 October 2018.
15. Niyaz, Q.; Sun, W.; Javaid, A.Y. A deep learning based DDoS detection system in software-defined networking (SDN). *arXiv* **2016**, arXiv:1611.07400.
16. Wang, P.; Chao, K.M.; Lin, H.C.; Lin, W.H.; Lo, C.C. An efficient flow control approach for SDN-based network threat detection and migration using support vector machine. In Proceedings of the IEEE Conference on e-Business Engineering, Macau, China, 4–6 November 2016; pp. 56–63.
17. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, Singapore, 8–10 August 2018; pp. 108–116. Available online: <http://www.scitepress.org/DigitalLibrary/Link.aspx?> (accessed on 20 November 2022).
18. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* **2017**, *239*, 39–57. [\[CrossRef\]](#)
19. García, S.; Luengo, J.; Herrera, F. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowl.-Based Syst.* **2016**, *98*, 1–29. [\[CrossRef\]](#)
20. Roy, S.S.; Mittal, D.; Biba, M.; Abraham, A. Random forest, support vector machine and nearest centroid methods for classifying network intrusion. *Comput. Sci. Ser.* **2016**, *14*, 9–17.
21. WID Mining. *Data mining Concept and Techniques*; WID Mining: New York, NY, USA, 2006.
22. Uddin, S.; Haque, I.; Lu, H.; Moni, M.A.; Gide, E. Comparative performance analysis of K-nearest neighbor (KNN) algorithm and its different variants for disease prediction. *Sci. Rep.* **2022**, *12*, 6256. [\[CrossRef\]](#)
23. Imandoust, S.B.; Bolandraftar, M. Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *Int. J. Eng. Res. Appl.* **2013**, *3*, 605–610.
24. Ihsan, M.A. Reduksi Atribut Pada Algoritma K-Nearest Neighbor (KNN) Dengan Menggunakan Algoritma Genetika. Doctoral Dissertation, Universitas Sumatera Utara, Kota Medan, Indonesia, 2018.
25. Universitas Sumatera Utara. *Botnet Detection Using the K-Nearest Neighbor Algorithm*; Universitas Sumatera Utara: Sumatera Utara, Indonesia, 2018.
26. Balogun, A.O.; Balogun, A.M.; Sadiku, P.O.; Amusa, L. An ensemble approach based on decision tree and bayesian network for intrusion detection. *Comput. Sci. Ser.* **2017**, *15*, 82–91.
27. Rezaeiapanah, A.; Ahmadi, G. Breast cancer diagnosis using multi-stage weight adjustment in the MLP neural network. *Comput. J.* **2022**, *65*, 788–804. [\[CrossRef\]](#)
28. Xie, Y.; Zaccagna, F.; Rundo, L.; Testa, C.; Agati, R.; Lodi, R.; Manners, D.N.; Tonon, C. Convolutional neural network techniques for brain tumor classification (from 2015 to 2022): Review, challenges, and future perspectives. *Diagnostics* **2022**, *12*, 1850. [\[CrossRef\]](#)
29. Liu, X.Y.; Wu, J.; Zhou, Z.H. Exploratory undersampling for class-imbalance learning. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2009**, *39*, 539–550.

30. Wang, Y.; Jia, Y.; Tian, Y.; Xiao, J. Deep reinforcement learning with the confusion-matrix-based dynamic reward function for customer credit scoring. *Expert Syst. Appl.* **2022**, *200*, 117013. [[CrossRef](#)]
31. Heydarian, M.; Doyle, T.E.; Samavi, R. MLCM: Multi-label confusion matrix. *IEEE Access* **2022**, *10*, 19083–19095. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.