Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

RESEARCH ARTICLE

Voting based Intrusion Detection Framework for Securing Software-Defined Networks

Rochak Swami* | Mayank Dave | Virender Ranga

Department of Computer Engineering, National Institute of Technology Kurukshetra, Haryana, India

Correspondence
*Rochak Swami, Department of Computer Engineering, NIT Kurukshetra, Haryana, India. Email: rochakswami123@gmail.com

Present Address
Department of Computer Engineering, NIT Kurukshetra, Haryana, India.

Summary

Software-defined networking (SDN) is an emerging paradigm in enterprise networks because of its flexible and cost-effective nature. By decoupling control and data plane, SDN can provide various defense solutions for securing futuristic networks. However, the architectural design and characteristics of SDN attract several severe attacks. Distributed Denial of Service (DDoS) is considered as a major destructive cyber attack that makes the services of controller unavailable for its legitimate users. In this research paper, an intrusion detection framework is proposed to detect DDoS attacks against SDN. The proposed framework relies on voting based ensemble model for the attack detection. Ensemble model is a combination of multiple machine learning classifiers for prediction of final results. In this research paper, we propose and analyze three ensemble models named as Voting-CMN, Voting-RKM, and Voting-CKM particularly to benchmarking datasets like UNSW-NB15, CICIDS2017, and NSL-KDD, respectively. For validation of the proposed models, a cross validation technique is used with the prediction algorithms. The effectiveness of proposed models is evaluated in terms of prominent metrics (accuracy, precision, recall, and F-measure). Experimental results indicate that the proposed models achieve better performance in terms of accuracy as compared to other existing models.

KEYWORDS:
Software-defined networking, DDoS, IDS, Voting

1 | INTRODUCTION

Software-defined networking (SDN) is a promising networking paradigm that provides a new way of network management to the customers as per today's business requirements. SDN has provided a new vision to the networking world by separating its control logic from the data plane devices[1]. The intention behind this separation is to enable the network configuration with more flexibility and programmability. In traditional networks, data plane devices and control plane are tightly coupled with each other. In that case, if any new rule or policy is required to be incorporated/updated in the existing mechanism, it has to be implemented in all the devices. Even a simple change can take weeks or months to be completed. This shortcoming of traditional networks makes this process time-consuming and costly. But in the time of digital world, companies and their networking facilities need to be flexible and fast. SDN fulfills this requirement by reducing the shortcomings of traditional networks. In SDN, if any update is required to be implemented as per the user's requirement, these changes are made only in the control plane. SDN completes

[0]**Abbreviations:** ANA, anti-nuclear antibodies; APC, antigen-presenting cells; IRF, interferon regulatory factor

this process in less time as compared to other traditional networks. Therefore, we can say that SDN improves the performance of the network and provides cost effective network services. In SDN, forwarding devices and control logic are governed in two separate planes namely data and control plane[2]. Data plane is also known as data forwarding plane because of its forwarding functionality. Networking devices such as routers, switches are placed in the data plane while the control logic of the network is deployed on a centralized controller. SDN controller acts as the brain of the whole network and provides global visibility to all the network functionalities[3]. This unique feature of the controller offers several applications and mechanisms for security purposes. At the same time, architecture design issues such as decoupling of control and data planes, and centralized nature of controller lead to various security challenges[4]. One such challenge is to secure the controller from a rapidly growing attack, i.e., Distributed Denial of Service (DDoS)[5] attack. It has become one of the most devastating attacks nowadays[6–9]. DDoS seeks to exhaust the system's resources by the flood of traffic at targeted server or service. This flood of traffic enforces the degradation of the system's performance, which makes the services unavailable for it's legitimate users. DDoS attack targets the controller's resources and communication channel between controller and switches. Threats like DDoS on SDN controller can paralyze the whole network. Therefore, it is necessary to find out an efficient and accurate mechanism for detection of DDoS attacks in SDN.

For detection of these attacks, various intrusion detection systems (IDS) have been proposed by researchers and networking communities[10–16]. An IDS is basically a device or a security application that is implemented to monitor and analyze the network traffic for any malicious activities, and to alert the management system when an attack is discovered. The primary function of an IDS is to detect the malicious activities and to report about them. For DDoS attack detection, machine learning (ML) based algorithms are being used extensively nowadays. In the literature, many effective defense solutions have been proposed by the researchers[17]. These defense solutions can be assessed with different evaluation measures. Some important measures have been discussed in a work presented by Praseed *et al.*[18]There are many IDSs that have utilized ML algorithms for various attack detection by improving the accuracy of the classification system[19–23]. Sultana *et al.*[24] discussed a study of SDN based IDS, which was implemented using ML for attack detection. This study presented various possible challenges in developing IDS with ML classifiers. In SDN, DDoS attacks are usually identified using a single individual classifier. Commonly used algorithms are Artificial Neural Network (ANN)[25], Support Vector Machine (SVM), *k*-Nearest Neighbor (*k*-NN), etc. In a study, a heterogeneous multi-classifier ensemble model was introduced for DDoS detection, which gives a promising way to provide stronger predictive results[26]. Ensemble learning (EL) is used in the proposed work that is based on voting scheme. It aims to improve the classification results in terms of accuracy. The key objective is to propose a more accurate and stable classification model.

In this research work, we suggest an intrusion detection framework to detect the DDoS attacks in SDN that is based on EL. The proposed framework is implemented on the SDN controller as an attack detection application. To develop and analyze the IDS framework for defending DDoS attacks against SDN is the aim of this research work. EL is used to achieve better prediction results. EL is a process of combining multiple classification models to provide accurate results. Different ML classifiers are used in detection to act as base classifiers. In this work, soft voting scheme is used in ensemble model in order to make predictions. Soft voting based classifier makes the predictions by averaging the output probabilities of individual classifiers. We propose and analyze three different voting based ensemble models specific to datasets, i.e., UNSW-NB15[27], CICIDS2017[28], and NSL-KDD[29]. The performance evaluation of the proposed ensemble models is done in terms of accuracy, precision, recall, and F-measure. Cross-validation is used to validate the performance of the classifier in terms of prominent classification metrics. In this work, the intrusion detection problem is considered as a binary classification problem, i.e., data is classified into attack or normal.

The key contributions of this study are highlighted below:

- Presents a summary of the normal and attack scenario based on SDN architecture.
- Proposes an IDS framework that is based on voting based ensemble model for attack detection in SDN.
- Proposes three voting based ensemble models and evaluate their performance on benchmarking datasets, i.e., UNSW-NB15, CICIDS2017, and NSL-KDD.

The remaining structure of the paper is organized in different sections for facilitating the understanding of the proposed work. Section 2 discusses the background information about SDN architecture and its different layers, major DDoS attacks in SDN and different ML algorithms used in this work. Section 3 shows related work for the SDN based on ML algorithms. Section 4 explains the proposed work that includes IDS framework and proposed voting based ensemble models. In Section 5, the experimental setup used for ensemble models is discussed. Next, we illustrate the performance results of proposed ensemble models in Section 6. Lastly, the conclusion is drawn in Section 7.

2 | BACKGROUND

This section is commenced by discussing SDN architecture in the normal and attack scenarios. ML algorithms proposed in the present work are explained. Further, a majority voting based ensemble classifier is discussed.

2.1 | SDN Architecture

SDN architecture is divided into three planes:

1) Data Plane : This plane consists of a number of OpenFlow switches. These switches are responsible for forwarding of incoming packets to the destination host. They make the forwarding decisions based on flow table entries. SDN switches are not capable to take the decisions on their own. All the decisions are made on behalf of the predefined rules installed in flow tables. These rules are predefined by the controller and modified periodically. Therefore, SDN switches are just simple forwarding devices and called as dumb switches[3,30].

2) Control Plane : This plane has a controller, which is responsible for the proper functioning of the network. The controller keeps a global eye on all the devices and their working in the network. In the case of an unknown or new incoming packet, a message named *packet_in* is transferred to the controller, which takes the proper decision for that packet as shown in Fig. 1. SDN switches and controller communicate via OpenFlow protocol[3,31]. OpenFlow is the first standard communication protocol that has been developed for SDN.

3) Application Plane : This plane maintains a pool of network based applications such as routing, firewalls, security, load balancing, traffic monitoring, and so forth. These applications are implemented on the controller in order to define the control logic according to the requirement of the functionalities.

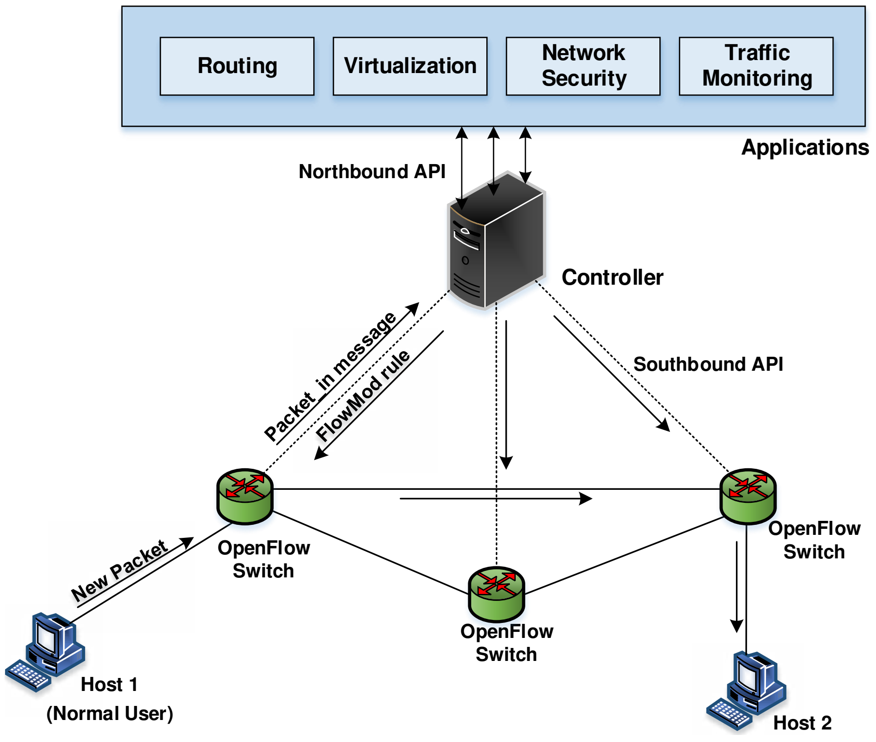

FIGURE 1 SDN Architecture

2.2 | Threat Model

SDN is a centralized network in which a controller manages all the forwarding devices and their working in the network. Controller manages the complete functionality of the network. OpenFlow switches are responsible for forwarding the packets to the destination host. They work on the basis of flow entries placed in the flow table. Entries of the flow table are described in Fig. 2. It consists of three entries, i.e., rule, action, and counters. Rules contain various header fields that are used as matching criteria. These flow rules for the packets are predefined by the controller. Counter field is used to maintain the statistics related to packets, i.e., how many packets have been forwarded or dropped for a particular flow. Action field specifies what actions should be taken by switch. Therefore, switch sends a message known as *packet_in* to the controller, if the header fields do not match with rules of the flow table. Controller takes an appropriate decision for that packet whether it should be dropped or forwarded. SDN controller sends back a flow modification (*FlowMod*) rule to the switch in order to update the flow table.

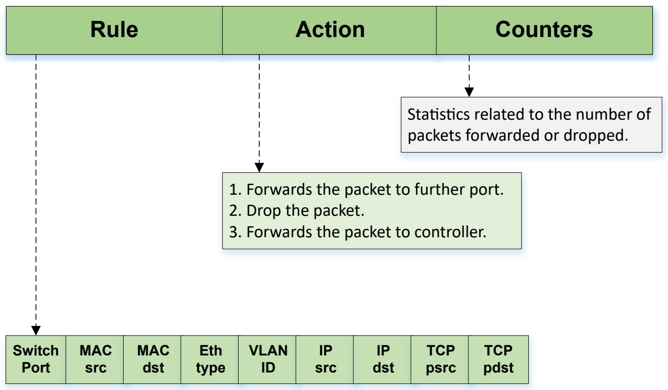
Rule
Action
Counters
Statistics related to the number of packets forwarded or dropped.
1. Forwards the packet to further port.
2. Drop the packet.
3. Forwards the packet to controller.
Switch Port
MAC src
MAC dst
Eth type
VLAN ID
IP src
IP dst
TCP psrc
TCP pdst

FIGURE 2 Structure of Flow Table

DDoS attempts to make a malicious attempt for exhausting the services of server for a certain amount of time. In SDN, DDoS attack generates a flood of new packets and sends them to the SDN switches. As shown in Fig. 3, host 1 acts as an attacker and sends a number of packets to the SDN switch. These flooding packets do not match with the rules in the flow table therefore, many *packet_in* messages are forwarded to the controller. As a result, controller sends a number of *FlowMod* messages containing rules in order to update the rules in the switch's flow table. These *FlowMod* rules are sent by the controller upto a limit until controller gets collapsed. Thus, this enormous flood of packets prompts the network to get affected by bandwidth congestion and exhaustion of controller resources. In Fig. 3, a scenario of DDoS attack is illustrated on SDN architecture. In the presented scenario, two DDoS attacks, i.e., controller exhaustion and controller-switch channel congestion have been indicated. These attacks affect the processing of the SDN network.

1. Controller Exhaustion : SDN controller is the main intellect entity of the network. After receiving a *packet_in* message, the controller activates processing for incoming message. This processing requires controller's resources - (CPU, memory, and bandwidth, etc) to be utilized. In case of a large volume of *packet_in* requests received, controller needs to use all its resources completely and may run out of resources. Exhaustion of controller makes the benign users unserved the services of controller, which can degrade the performance of the complete network.

2. Controller-Switch Channel Congestion : DDoS attack enforces the data plane switch to forward *packet_in* messages to controller. For such flooding, OpenFlow protocol is used for facilitating communication between switch and controller. The channel gets congested by a huge volume of data messages as described in Fig. 3. All SDN switches and controllers share the same physical link for connectivity. Congestion between a switch and controller causes all the links to get affected, which further causes the legitimate packets to be dropped out.

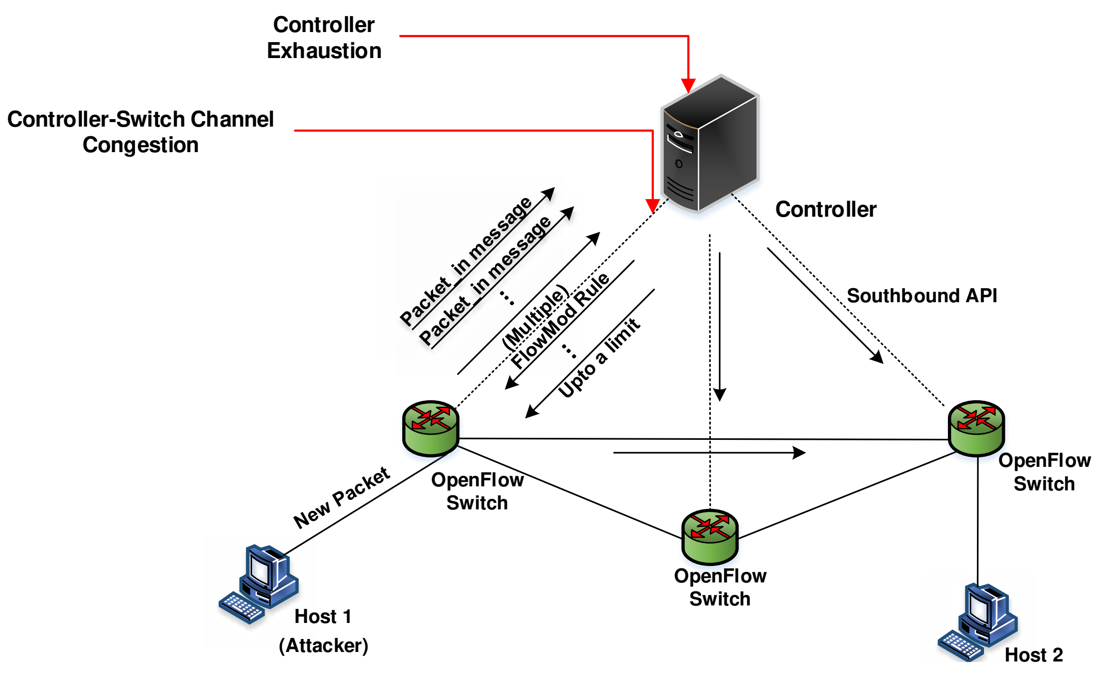

FIGURE 3 A Scenario of DDoS Attacks in SDN

2.3 | Machine Learning Algorithms

ML provides a set of algorithms, which work on the automatic learning ability and prediction of the final results. These algorithms can be used for various applications. One such application is security from various attacks. ML algorithms are being utilized in the development of security solutions for the networks. These algorithms are adopted to classify normal and attack traffic.

Wolpert et al.[32] stated a theorem popularly known as “No Free Lunch" theorem for making a selection of different ML classifiers to solve a classification problem. The “No Free Lunch" theorem states “there is no single learning algorithm that universally performs best across all domains"[33]. It can be said that there is no such a single classification model that gives the best performance results for each and every problem. The assumptions of one model with best performance for one problem may not work for another problem. Therefore, different ML classifiers should be tried to solve a classification problem. We use ML models for intrusion detection and classify the network traffic. In this paper, we consider different ML models belonging to different families, i.e., tree (CART), ensemble tree (RF), neural network (MLP), distance-based (k-NN), and gaussian model (NB) to propose an accurate ensemble model.

2.3.1 | Random Forest

Random Forest (RF)[34] is a supervised learning algorithm that generates a collection of decision trees (DT) from randomly selected subsets of the training dataset. RF aggregates the votes from different DTs to make predictions. In other words, it is an ensemble method using tree based classifiers, i.e., trees $t(x_{in}, \theta_n)$, n = 1, ... where θ_n is the random set of variables that are sampled independently with the same distribution. x_{in} is an input pattern on which predictions are made by determining the majority votes for the most popular class. The key idea behind this ensemble method is that it tries to achieve high accuracy with a number of decision trees. It is a versatile algorithm, which is capable of solving both classification and regression problems. RF has the power to handle large datasets with higher dimensionalities. For the performance evaluation, 100 trees are used in the forest.

2.3.2 | Classification and Regression Trees

One of the best implementations of DTs is Classification and Regression Trees (CART)[35]. CART is one of the most powerful classification algorithms to be used for classification problems. CART works on tree based structure in which nodes represent features, links represent decision rule, and leaf nodes represent predicted output. It involves the partitioning of the input space recursively and fitting a simple prediction model within each partition. This partitioning can be represented as a decision tree graphically. CART utilizes an exhaustive search technique on each node in order to identify splitting variables such that the total impurity of node's children is minimized. It is capable to handle missing values of the features. CART is able to handle numeric and nominal data. It utilizes Gini index as an impurity function and provides computationally efficient prediction results. The maximum depth of tree construction is set to 10 in this algorithm.

2.3.3 | Multi-Layer Perceptron

Multi-layer perceptron (MLP)[36] is derived from the family of feed-forward ANNs. It contains a layer based structure of connected artificial neurons. The key idea behind MLP is that the working of artificial neurons is analogous to biological neurons. It is a perceptron having an input, hidden, and output layer. Each neuron has its own activation function. Input layer consists of a number of neurons, i.e., $\{x_1, x_2, ..., x_n\}$ with corresponding input features $\{X = x_1, x_2, ..., x_n\}$. Middle layer, i.e., hidden layer conducts the transformation for converting inputs features to the final output. Inputs from the previous layer are summed up with the assigned weights using summation, which is represented in eq. (1). After performing the summation function, a non linear activation function is applied to obtain output, which is transferred to the next layer as a predicted output as shown in eq. (2) where $f(\cdot)$ is an activation function. The output layer takes the values from the hidden layer as input and gives the final prediction results as an output. For MLP the optimal parameter values are: hidden layer size of 100, learning rate of 0.001, and 200 maximum iterations.

$$a = \sum_{j=1}^{n} w_j x_j \tag{1}$$

$$y = f(a) \tag{2}$$

2.3.4 | *k*-Nearest Neighbor

k-Nearest Neighbor (*k*-NN)[37] is a supervised learning algorithm that is used in the IDSs. The key idea behind this algorithm is that it classifies the dataset based on their similarity measure with its neighbors. Its working involves storing all the cases and classifying the new case by using a similarity measure, i.e., distance measure. Most commonly, euclidean distance is used to carry out the similarity between two different instances. To classify a new instance, a majority voting is performed by its closest k number of neighbors using the distance measure and it is assigned to the most common class among k number of nearest neighbors. The number of neighbors of 5 and leaf_size of 30 are set as optimal parameters for *k*-NN.

2.3.5 | Naive Bayes

Naive Bayes (NB)[38] is a probabilistic model based on Bayes theorem. In this algorithm, presence of one feature does not affect another feature's presence. NB is a conditional probability model where an input instance $X = (x_1, x_2, ..., x_n)$ is represented with n number of features. The probability function is defined for the target value or output (y) as in eq. (3).

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)} \tag{3}$$

The above expression can also be expressed as in eq. (4).

$$P(y|x_1, x_2, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)} \tag{4}$$

In eq. (4), the denominator part does not change, it remains the same for all the input values. Therefore, it can be removed and the predicted class value for the set of inputs is defined with maximum probability as in eq. (5)

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y), \tag{5}$$

where $P(y)$ is known as class probability and $P(x_i|y)$ is known as conditional probability. The optimal parameter values for NB are set to default values.

2.3.6 | Voting Classifier

Voting is a way to improve accuracy and achieve more stable prediction results. It is used in EL models to predict the performance results[39]. It involves a combination of multiple individual ML classifiers to achieve better predictive results. This combination of multiple classifiers provides more accurate results as compared to an individual classifier. The major steps of combining the N classifiers using voting are illustrated in Fig. 4. Therefore, the main goal of voting is to achieve more accurate and stable prediction models.

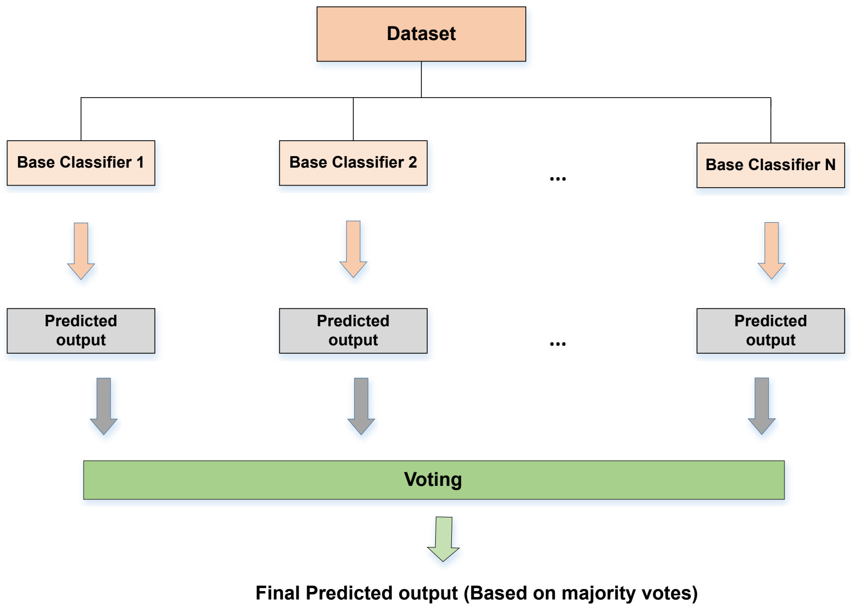

FIGURE 4 Voting based ensemble classifier

Voting classifier involves the aggregation of the predictions resulted from base classifiers and then predicting the class by voting[40]. For selecting different base classifiers, parameters such as diversity and accuracy should be considered. Voting classifier supports two types of voting schemes,

- Hard Voting: In hard voting, the class label is predicted as final class label, which obtains most of the votes from individual prediction classifiers. It is also known as majority voting. In eq. (6), y is the predicted class label voted by majority of N classifiers.

$$y = mode\{C_1(x), C_2(x), ..., C_N(x)\} \tag{6}$$

- Soft Voting: In soft voting, the final class label is defined based on the predicted probabilities for each prediction classifier. Soft voting is suggested only in case of the well-calibrated classifier. In eq. (7), class label is computed using weight w_j and probability p_{ij} assigned to the classifier C_j for class labels $i \in \{0, 1\}$.

$$y = argmax_i \sum_{j=1}^{N} w_j p_{ij} \tag{7}$$

3 | RELATED WORK

ML based techniques have become an important part of attack defense solutions. These ML based solutions are capable enough in discovering the differences between normal traffic and attack traffic with high accuracy. There are many research studies that suggest attack detection and defense solutions against DDoS attacks in SDN. Niyaz *et al.*[19] proposed an IDS that was implemented in SDN controller as an application. In this work, an stacked auto-encoder based deep learning model was used as a detection method. Detection model involves traffic flows collector, feature extractor, and then classification of the normal and malicious traffic flows. Performance evaluation was done on the basis of classification measures. However, it suffers from processing capabilities for the reason of feature extraction. Recently, Chen *et al.*[41] introduced an attack detection system using XGBoost classifier. The traffic collector and classification model were deployed in the controller by utilizing the features of the controller, i.e., global visibility. The goal was to solve the threats issues on controller. One another solution for DDoS detection based on ML approach was proposed Ye *et al.*[42]. This work is based on collecting the traffic statistics from flow table of switches by the controller. Then characteristics values are carried out from the network traffic and SVM is used to identify the attack traffic. He *et al.*[43] suggested two clustering based algorithms to detect DDoS attacks. One algorithm was feature selection based on clustering and second was density peak based clustering, which achieved better results in terms of memory efficiency and running time. A method named Weighted and Diversity (WAD) to measure the effectiveness of the ensemble classifier was introduced by Zeng *et al.*[44] This method finds a way to balance between accuracy and diversity to select a classifier for ensemble model. The harmonic mean of these two parameters is computed for performance assessment. Selection of ensemble model is performed on 15 benchmark datasets. Vimala *et al.*[45] proposed a SDN based attack detection solution in cloud environment. This defense solution relies on an ensemble model. This ensemble model combines k-NN, SVM, and Extreme Learning Machine (ELM). The proposed model is evaluated in terms of accuracy, recall, and precision. The ensemble model performs significantly better than individual classifiers as per results. Braga *et al.*[46] proposed a DDoS detection approach that used Self Organization Map (SOM) as a detection method. This method includes the analysis of traffic flow features that are extracted from OpenFlow switches. Jankowski *et al.*[47] presented a scheme for attack detection with the help of SDN technology and ML classifiers. The authors used SOM and Learning Vector Quantization and their enhanced version as detection methods in this study. Recently, Chen *et al.*[41] used extreme gradient boosting (XGBoost) classifier for DDoS detection in SDN. Ashraf *et al.*[48] analyzed the performance results of various ML classifiers, i.e., neural networks, SVM, decision tree, genetics algorithms, bayesian networks, and fuzzy logic in particular to SDN. Anand *et al.*[49] studied some threat vectors that represent the failure of SDN controller. The authors also analyzed the performance of attack detection schemes, which were based on ML classifiers. Diro *et al.*[50] introduced a deep learning based attack detection model that achieved excellent accuracy results. Balkanli *et al.*[51] analyzed the performances of two supervised learning ML classifiers such as CART and Naive Bayes on a CAIDA-provided darknet dataset. The performance was measured in terms of detection rate and computational time. Das *et al.*[52] proposed an IDS to detect the DDoS attacks that utilize a voting based ensemble model as a detection method. The ensemble model combines four different classifiers, i.e., MLP, SVM, K-NN, and DT. The performance of the proposed IDS is evaluated using cross-validation on NSL-KDD dataset in terms of accuracy, precision, recall, and F-measure. Singh *et al.*[53] proposed a ML based defense system against DDoS in SDN. The proposed system consists of three modules: flow statistics collection, features extraction, and training and classification of network traffic. For classifying the normal and attack traffic, different ML classifiers such as Logistic Regression, Nearest Neighbors, Decision Tree, SVM are analyzed in this defense system. The classifier with high accuracy and less false positive rate is selected for attack detection. Recently, a new DDoS mitigation system against flooding attacks in SDN was proposed by Tuan *et al.*[54]. In this system, k-NN and XGBoost are used as detection engines. The performance of the classifiers as a detection system is evaluated using CAIDA2007 dataset in terms of accuracy, precision, recall, and F-measure.

4 | PROPOSED WORK

This section includes the proposed IDS framework for the detection of DDoS attacks in SDN. Proposed ensembled models are also discussed in the section.

4.1 | Voting based IDS Framework

In this paper, we propose an ML based IDS framework to detect the attack in SDN network. Our proposed framework is implemented on the SDN controller. The controller can monitor and control all the switches. Controller can do request for all the network traffic statistics when it is required. Thus, having global visibility on the network can be beneficial for the IDS in identifying the intrusion. The IDS framework consists of ML based attack detection module that is implemented as an application.

IDS is designed with different components, which are responsible for performing different functions. Fig. 5 illustrates the components and workflow of the proposed framework. The main components of IDS framework are : 1) Packet Capturing; 2) Pcap Extraction and Feature Extraction; 3) Data Preprocessing; 4) Voting based classification.

In the proposed framework, traffic generated at a targeted switch is captured using a packet capture tool. Several tools can be used to capture all the communicated packets across the network such as Wireshark. It contains all the possible information fields of the incoming packet such as source IP, destination IP, source port, destination port, protocol, time-stamp, length, and other header details. Wireshark permits the administrator to see all the traffic being passed over the network. All the captured packets with its header fields are collected in a log file, known as pcap file. A pcap log file is converted into csv format using JSON parsing tool so that the information related to packets can be used. For parsing a pcap file, Python based tool Scapy or python-dpkt module can also be used. It coverts the log file into an accessible and readable data file. After parsing the pcap file, important features are extracted (derived) using existing features. These extracted features are utilized to train the proposed IDS.

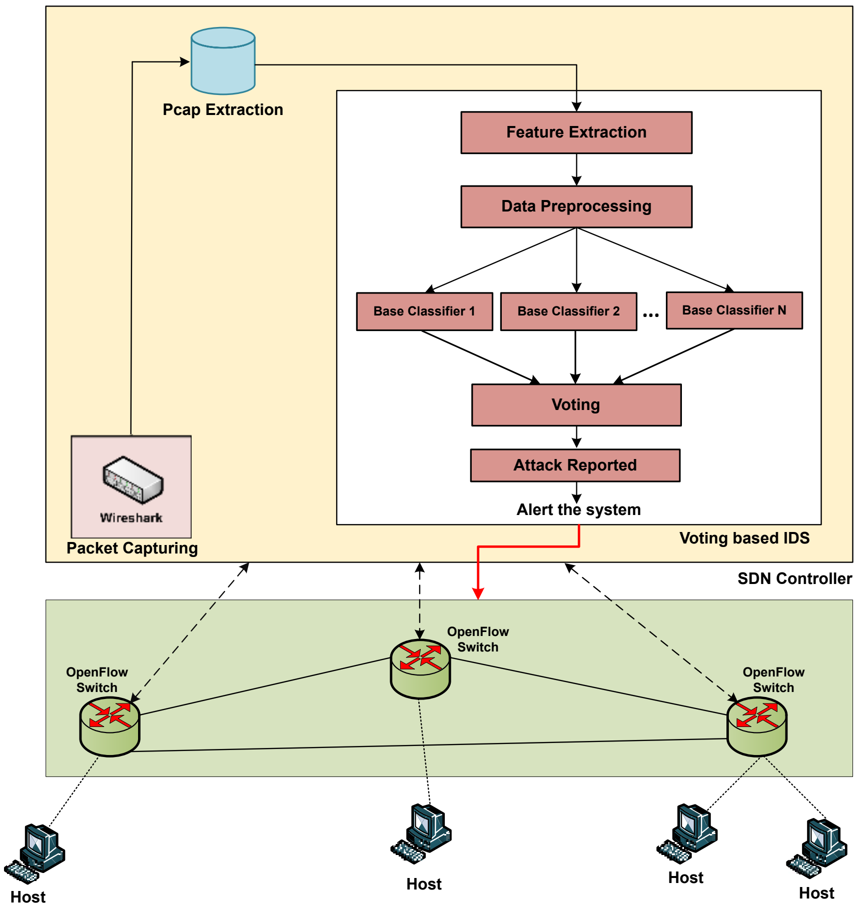

FIGURE 5 Architecture for the proposed IDS

For classification of the network flows based on ML techniques, preprocessing of dataset is an important step to be done. It is one of the necessary tasks in data mining. Mostly data contains numerous erroneous values such as missing values, infinity or NaN values. Therefore, the preprocessing of data plays an important role to resolve these issues. After preprocessing the data, ML based algorithm is applied to perform the classification. In the proposed IDS, a voting based EL model is considered as a detection mechanism to classify the traffic into benign or malicious traffic. It can take a number of resources for execution purpose but produces better results in terms of accuracy for the classification. EL is based on the combination of different ML classifiers. Three different ML classifiers are used as base classifiers for ensembling. Voting classifier is used to predict the final results on the basis of output of all the individual classifiers. Soft voting is used to predict the class in the proposed framework that gives more weight to the highly confident votes. As per the results, network traffic is categorized as either malicious or benign according to network statistics. Further, IDS alerts the controller if the classified traffic is detected as an attack. The controller informs the OpenFlow switches in data plane about the attack and to update their flow tables.

4.2 | Voting based Ensemble Models

EL[55,56] is an ML based approach that combines more than one classifier to generate better and optimal prediction results. Combination of multiple classifiers has an important role in the modeling of IDS to achieve higher accuracy and to reduce the detection error. The selected classifiers should be diverse in nature. Three different ML based classifiers are used for evaluation of the voting based classification model, which are tabulated as in Table 1. Three different voting based models are proposed for three different datasets. In this work, soft voting is used to predict the final results on the basis of output of all the individual classifiers. Soft voting is used to predict the class in the proposed framework, which gives more weight to the highly confident votes. Three different datasets have been used in this research work, i.e., UNSW-NB15, CICIDS2017, and NSL-KDD.

1. Ensemble model for UNSW-NB15 : This model relies on three different classifiers, i.e., CART, MLP, and NB and is named as Voting-CMN. The performance evaluation of the proposed model is measured using UNSW-NB15 dataset.
2. Ensemble model for CICIDS2017 : This voting based model involves three base classifiers, which are RF, *k*-NN, and MLP, named as Voting-RKM.
3. Ensemble model for NSL-KDD : This ensemble model combines CART, *k*-NN, and MLP classifiers, which improve the accuracy of the model on NSL-KDD dataset This model is named as Voting-CKM.

TABLE 1 Proposed Voting based Ensemble Models

Proposed Model	**Classifier 1**	**Classifier 2**	**Classifier 3**
Voting-CMN	CART	MLP	NB
Voting-RKM	RF	*k*-NN	MLP
Voting-CKM	CART	*k*-NN	MLP

5 | EXPERIMENTAL DESIGN

5.1 | Experimental Setup

The experimental assessment of the proposed voting based models is performed on PARAM SHAVAK DL-GPU system with 64-bit Ubuntu 14.04 and equipped with $Intel^{®}$ $Xeon^{®}$ Gold 6132 twenty eight core CPU having 2.6GHz clock speed and 96GB main memory. This system has been exclusively designed for deploying deep learning, big data and machine learning applications to be used by researchers. All the ML classifiers are implemented in Python programming language (version 3.6.1). For implementing the classifiers, a Python based ML library *scikit-learn*[57] is utilized.

5.2 | Datasets

As mentioned earlier, three different datasets UNSW-NB15[27], CICIDS2017[28], NSL-KDD[29] have been used in this work for performance evaluation of suggested ensemble models.

- UNSW-NB15 : This dataset is used to analyze the performance of the proposed model Voting-CMN. The dataset consists of 49 features and 1 class attribute. A part of the dataset is used for experiment as train and test set, i.e., UNSW_NB15_Train and UNSW_NB15_Test. The train set consists of 1,75,341 records and the test set consists of 82,332 records. The train set involves 56,000 records of normal traffic and 1,19,341 records of attack traffic. The test set involves 37,000 records of normal traffic and 45,332 records of attack traffic.

- CICIDS2017 : For model Voting-RKM, CICIDS2017 is used for benchmarking the classification assessment results. This dataset consists of 78 features and 1 class attribute. A part of the dataset-“Wednesday-WorkingHours" is used for validating the performance of the proposed voting based model. The dataset contains 6,92,703 records, which includes 4,40,031 records of normal traffic and 2,52,672 records of attack traffic.
- NSL-KDD : This dataset is used to perform experimental analysis for a different ensemble model named Voting-CKM. The dataset contains 41 features and 1 class attribute. A subset KDDTrain+ (training) of NSL-KDD is used in this paper. This part of dataset contains a total of 25,192 records out of which 13,499 records are attack traffic and 11,743 records are of normal traffic.

5.3 | Dataset Preprocessing

The preprocessing of dataset is an important step in ML and to be performed because the real-world data generally have missing values, duplicate values, errors, and NaN/Infinity values that may affect the performance of the classification model. Thus the preprocessing improves the performance of the IDS by resolving such flaws. This step has some functions, i.e., feature reduction (a special case of feature selection), conversion of features and normalization. The actions performed during preprocessing of the datasets are explained below:

1. Drop the missing values: While extracting the features, it may also be possible that there are missing values for some features. It is one of the most challenging tasks to handle the missing values and can make significant changes to the final result. Therefore, it is necessary to check for the missing values and process them as per the dataset requirement.
2. Drop the duplicate packets: The dataset may contain duplicate packets having the same instances. To avoid this overhead, duplicate packets are dropped, which do not serve any purpose in model training.
3. Relevant format of the features: In the dataset, all the features may not have the numeric values. The feature may have a combination of numeric and string values. It is required to convert the string values into numeric values so that data can also be processed easily. In our data, the predicted class is labeled as an attack or normal that is changed to 1 or 0, respectively.
4. Features Scaling: Feature scaling is a technique to scale the features in a specific range to be processed on the same ground. There are two types of feature scaling, i.e., normalization and standardization. Normalization is used to scale the features in the range between 0 and 1 whereas standardization transforms the features having mean value as 0 and standard deviation as 1.

6 | PERFORMANCE EVALUATION

6.1 | Performance Indicators

In this work, the performance evaluation of proposed IDS framework is done using various performance indicators (metrics). An IDS is supposed to have high accuracy and high detection rate.Thus, to study the effectiveness of the proposed system we have used accuracy, precision, recall, and F-measure as performance indicators. The main reason behind the selection of these metrics for performance evaluation is that they are popularly used in the literature to evaluate the performance of classification algorithms[52,54,58,59]. Moreover, our focus is on studying the effectiveness of classifiers in terms of their capability to detect attacks. Therefore, we have chosen precision, recall, and F-measure over false alarm rate.

The various performance indicators are defined as follows:

- Accuracy : It is the number of correct predictions over all the predictions in the dataset. It is also known as classification rate.
- Precision : It is the number of correctly predicted attack instances over all the predicted attack instances.
- Recall : It represents the number of positive predictions that are correctly identified.

- F-measure : It is defined as a harmonic mean of precision and recall. It is also called as F-score. It seeks to balance between precision and recall.

The validation of the proposed IDS framework is conducted using K-fold cross validation ($K = 10$). In 10-fold cross validation, the training set is divided into 10 equal parts and each part is called a fold. Out of all the parts, one part is used for testing the model whereas the other 9 parts are used for model training. This process is performed 10 times (K-times) repeatedly. Further, 10 (K) results from all the folds are summarized with an average of all the estimated results to generate a single prediction result. Cross-validation is performed to provide more effective assessment of the proposed ML model. The estimations provided by cross-validation are more accurate with mitigating the effects of sampling, which may be a case in hold-out validation. We repeated all the experiments 30 times to avoid biased results. The proposed IDS framework is evaluated using different seed values and then the mean value is reported.

6.2 | Result Analysis and Discussion

In this section, we discuss the performance results and analysis of the proposed voting based ML models with respect to UNSW-NB15, CICIDS2017, and NSL-KDD benchmarking datasets. The performance results of CART, MLP, NB, and their ensemble voting classifiers are tabulated in Fig. 6. In this work, we focus on achieving better accuracy and a stable model.

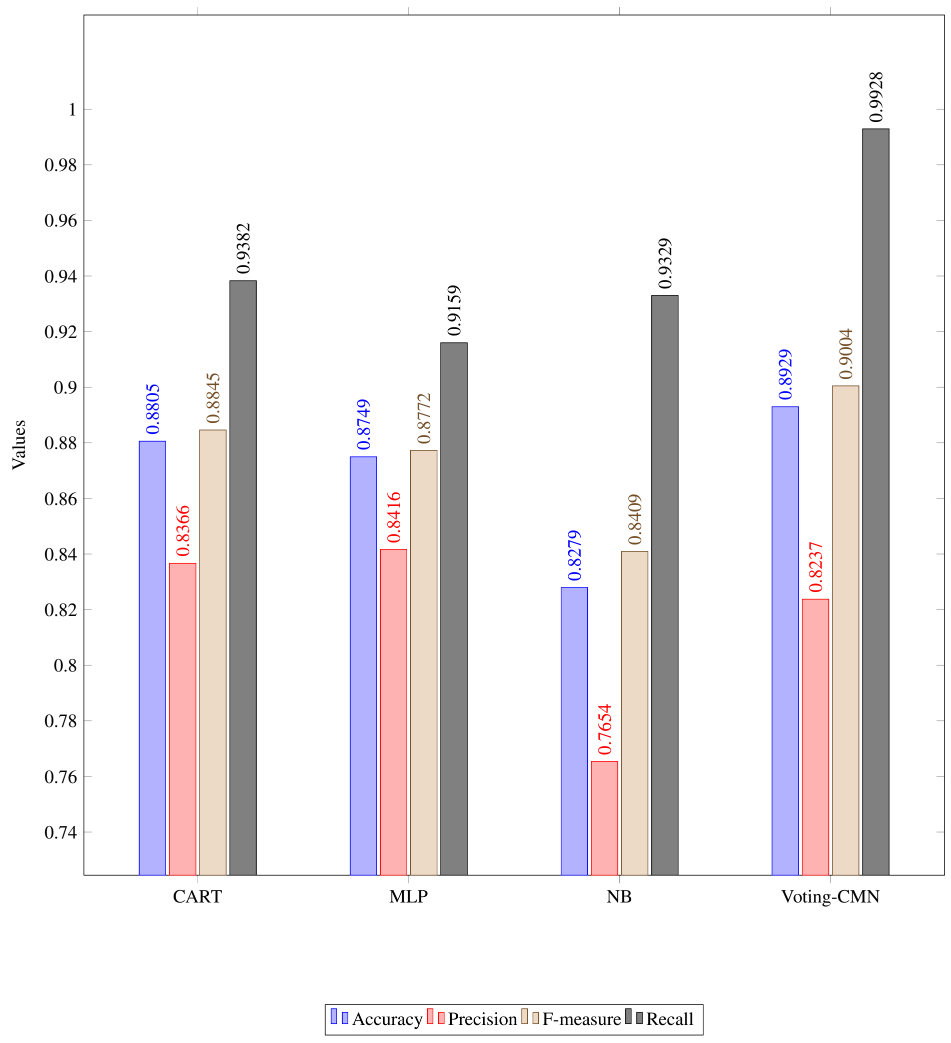

FIGURE 6 Performance of Voting classifier on UNSW-NB15 dataset

It is observed that model Voting-CMN outperforms than its base classifiers in terms of accuracy (89.29%), whereas the accuracies of CART, MLP, and NB are 88.05%, 87.49% and 82.79%, respectively. However, Voting-CMN provides improved results in terms of recall (90.04%) and F-measure (99.28%). As we can see in Fig. 6 that proposed classifier model achieves the best accuracy. In terms of precision, MLP achieves the best results as of 84.16%. Further, CART performs best among all the base classifiers by achieving 88.05% of accuracy, 88.45% of F-measure, and 93.82% value of recall. In terms of precision, MLP performs better than CART and NB. NB gives the worst result in terms of precision (76.54%), accuracy (82.79%), and F-measure (84.09%), whereas in terms of recall, MLP performs the worst with the value of 91.59%.

As shown in Fig. 7, our second voting model (Voting-RKM) achieves the accuracy of 97.77% on dataset CICIDS2017. Base classifiers (RF, *k*-NN, and MLP) achieve accuracy results of 97.41%, 97.57%, and 97.57%, respectively, which are lower than ensembled voting classifier. Experiment results show that the proposed model - Voting-RKM also performs better in terms of F-measure (96.36%). RF achieves the highest value of precision as of 99.89% and MLP performs best in terms of recall (93.25%) over all other classifiers.

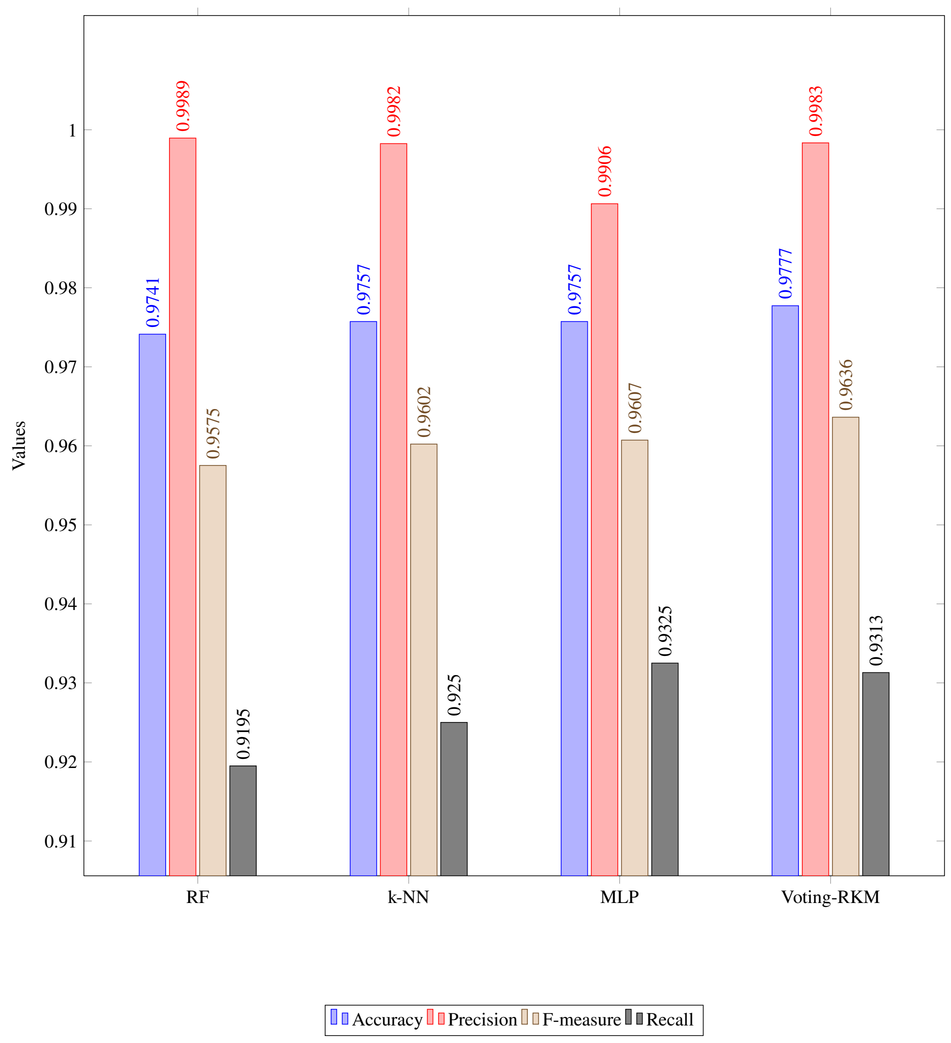

FIGURE 7 Performance of Voting classifier on CICIDS2017 dataset

Among all the base classifiers that are used for the ensemble model, MLP and *k*-NN achieve higher results by acquiring the same accuracy of 97.57%. Again MLP also performs best in terms of F-measure (96.07%) which is higher than RF (95.75%) and *k*-NN (96.02%). RF achieves the worst results in terms of accuracy (97.41%), recall (91.95%), and F-measure (95.75%) and

MLP performs worst in terms of precision (99.06%). Therefore, among all the classifiers voting classifier gives an improved and stable model by achieving the highest accuracy.

In one ensemble model we performed experiment on voting classifiers including CART, *k*-NN, and MLP on NSL-KDD. This model (Voting-CKM) achieves improved classification results over its ensembled base classifiers. As shown in Fig. 8 that voting classifier increases the accuracy upto 99.68%, which is higher than base classifiers. CART achieves accuracy of 98.49%, *k*-NN achieves 99.59% accuracy, and MLP upto 99.65%. Further, voting classifier performs better also in terms of precision (99.74%) and F-measure (99.66%). However, it achieves 99.57% average value of recall, which has a very less difference from the best recall value (99.58%) in all the base classifiers. Therefore, Voting-CKM performs as the best classifier, which results in a more stable and accurate model. among all the base classifiers, MLP achieves better results in terms of classification indicators, i.e., accuracy of 99.65%, precision of 99.68%, recall of 99.58%, and 99.63% f-measure.

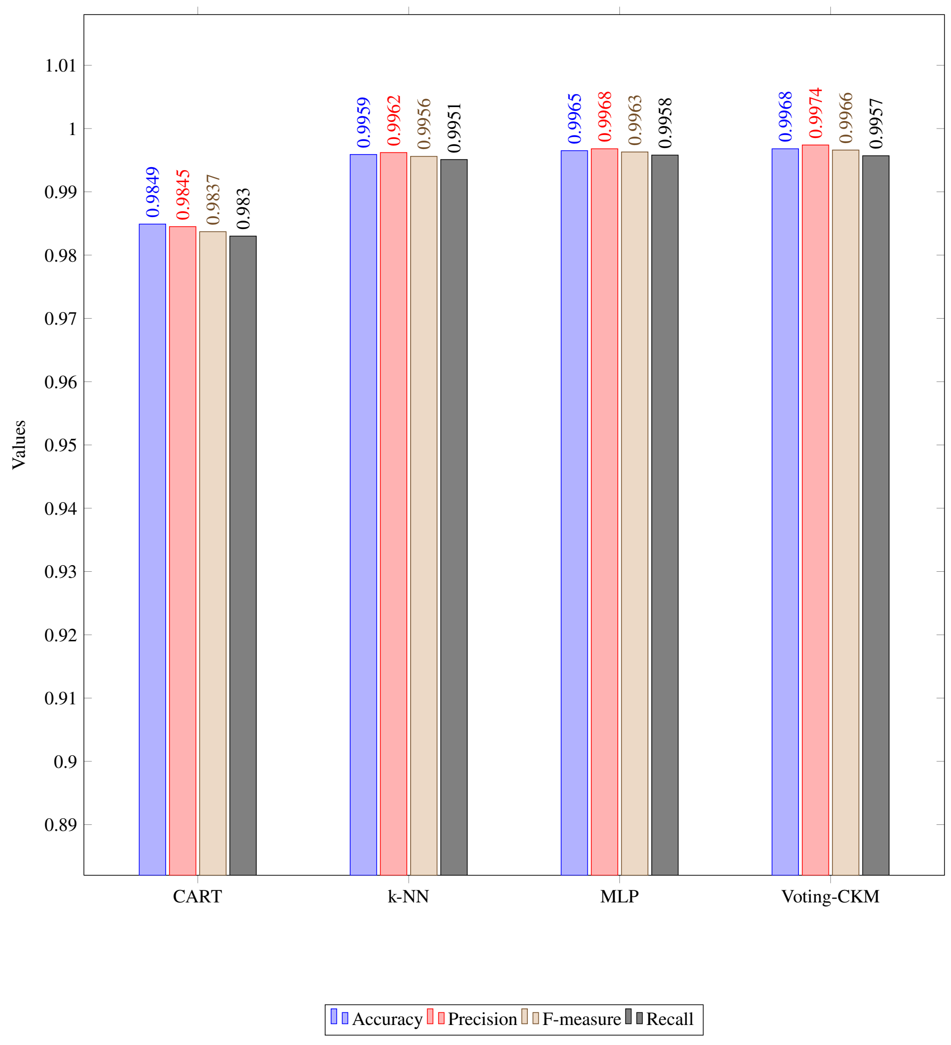

FIGURE 8 Performance of Voting classifier on NSL-KDD dataset

It is observed from the experimental results shown in Fig. 8 that MLP performs the best over CART and *k*-NN, whereas CART performs the worst in terms of all the classification measures, i.e., 98.49% accuracy, 98.45% precision, 98.37% F-measure, and 98.30% of recall. Classification results of all three ensemble models are tabulated in Table 2.

TABLE 2 Classification results of Voting based Ensemble models

Ensemble Model	**Accuracy**	**Precision**	**Recall**	**F-measure**
Voting-CMN	89.29%	82.37%	99.28%	90.04%
Voting-RKM	97.77%	99.83%	93.13%	96.36%
Voting-CKM	99.68%	99.74%	99.57%	99.66%

6.3 | Comparative Study

It is observed that the proposed models perform better than individual classifier. Voting ensemble model combines multiple classification models to predict the final result, whereas in an individual classification model final result is predicted on the basis of learning of a single classifier. It always provides a strong decision by taking opinions from multiple classifiers into consideration. Therefore, it ensures improvement in stability and accuracy of the machine learning models. We compare the performance results of Voting-CMN with the performances of ML techniques presented in[60]. As shown in Table 3, the proposed voting classifier achieves the best accuracy (89.29%) among other listed algorithms. However, in terms of FAR, Voting-CMN performs worse than DT, LR, and NB, whereas it performs better than EM clustering and ANN by achieving upto 20.0%. As discussed previously that our aim is to focus on the accuracy, it is observed that Voting-CMN performs better than existing techniques, i.e., DT, LR, NB, ANN, and EM.

TABLE 3 Comparing Voting-CMN with other models on UNSW-NB15

Algorithms	Accuracy	FAR
DT	85.56	15.78
LR	83.15	18.48
NB	82.07	18.56
ANN	81.34	21.13
EM clustering	78.47	23.79
Voting-CMN	**89.29**	20.2

In Table 4, Voting-RKM is compared with XGBoost based IDS[61] using CICIDS2017 dataset. XGBoost classifier achieves an accuracy of 91.36%, which is lower than proposed model (97.77%). In terms of recall, XGBoost performs better than Voting-RKM. It can be observed from Table 4 that Voting-RKM achieves higher accuracy than XGBoost.

TABLE 4 Comparing Voting-RKM with other model on CICIDS2017

Algorithms	Accuracy	Recall
XGBoost-IDS[61]	91.36	97.4
Voting-RKM	**97.77**	93.13

TABLE 5 Comparing Voting-CKM with other model on NSL-KDD

Algorithms	Accuracy	Detection Rate	Precision	F-measure
OS-ELM (Alpha-full features)	98.69	98.93	98.55	98.74
Voting-CKM	**99.68**	**99.57**	**99.74**	**99.66**

Further, the performance of Voting-CKM is compared with an existing algorithm[62] on NSL-KDD dataset. In the compared paper, classification measures have been analyzed with different number of hidden neuron. We compared Voting-CKM with the OS-ELM (Alpha full features) having the highest accuracy (98.69%), which is lower than proposed model (99.68%). As shown in Table 5, Voting-CKM also performs better in terms of detection rate (99.57%), precision (99.74%), and F-measure (99.66%).

7 | CONCLUSION AND FUTURE WORK

Software-Defined Networking provides a promising way to the networking customers by dynamically handling its infrastructure and services. However, SDN security is considered to be one of the major concerns for the research in the present scenario. In this research paper, we proposed an intrusion detection system framework that relies on majority voting based ensemble learning for attack detection. We performed various experiments on three voting models using UNSW-NB15, CICIDS2017, and NSL-KDD dataset and observed the performance of all the classifiers. Our observation was evaluated in terms of accuracy, precision, recall, and F-measure. The proposed ensemble model Voting-CMN achieved the highest accuracy measure of 89.29% experimented on UNSW-NB15 dataset. The performance of Voting-RKM and Voting-CKM classifiers was evaluated on CICIDS2017 and NSL-KDD, respectively. Voting-RKM achieved the accuracy of 97.77%, which is higher than all its base classifiers. Similarly, Voting-CKM improved the performance of the classifier model by ensembling base classifiers, which provided stability in the performance results. The proposed ensemble models performed better compared to other models. All the experiments were evaluated using 10-fold cross validation. It can be concluded that evaluated ensemble models are suitable for SDN based network intrusion detection systems. Our future goal is to implement an intrusion detection system to detect and mitigate DDoS in real-time.

References

1. Hakiri A, Gokhale A, Berthou P, Schmidt DC, Gayraud T. Software-Defined Networking: Challenges and research opportunities for Future Internet. *Computer Networks* 2014; 75: 453–471.

2. Kirkpatrick K. Software-defined Networking. *Commun. ACM* 2013; 56: 16–19.

3. Kreutz D, Ramos FM, Verissimo P, Rothenberg CE, Azodolmolky S, Uhlig S. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE* 2015; 103: 14–76.

4. Swami R, Dave M, Ranga V. Software-defined Networking-based DDoS Defense Mechanisms. *ACM Computing Surveys (CSUR)* 2019; 52(2): 28.

5. Swami R, Dave M, Ranga V. DDoS Attacks and Defense Mechanisms Using Machine Learning Techniques for SDN. In: IGI Global. 2020 (pp. 193–214).

6. Douligeris C, Mitrokotsa A. DDoS attacks and defense mechanisms: classification and state-of-the-art. *Computer Networks* 2004; 44: 643–666.

7. Specht SM, Lee RB. Distributed Denial of Service: Taxonomies of Attacks, Tools and Countermeasures, Princeton architecture laboratory for multimedia and security. tech. rep., technical report; 2003.

8. Tripathi N, Hubballi N. Slow rate denial of service attacks against HTTP/2 and detection. *Computers & security* 2018; 72: 255–272.

9. Singh K, Singh P, Kumar K. Application layer HTTP-GET flood DDoS attacks: Research landscape and challenges. *Computers & security* 2017; 65: 344–372.

10. Sattar D, Matrawy A, Adeojo O. Adaptive bubble burst (ABB): Mitigating DDoS attacks in software-defined networks. In: IEEE. ; 2016: 50–55.

11. Tripathi N, Hubballi N, Singh Y. How secure are web servers? An empirical study of slow HTTP DoS attacks and detection. In: IEEE. ; 2016: 454–463.

12. Tripathi N, Hubballi N. Detecting stealth DHCP starvation attack using machine learning approach. *Journal of Computer Virology and Hacking Techniques* 2018; 14(3): 233–244.

13. Tripathi N, Hubballi N. Exploiting dhcp server-side ip address conflict detection: A dhcp starvation attack. In: IEEE. ; 2015: 1–3.

14. Yan Q, Gong Q, Yu FR. Effective software-defined networking controller scheduling method to mitigate DDoS attacks. *Electronics Letters* 2017; 53(7): 469–471.

15. Zheng J, Li Q, Gu G, Cao J, Yau DK, Wu J. Realtime DDoS defense using COTS SDN switches via adaptive correlation analysis. *IEEE Transactions on Information Forensics and Security* 2018; 13(7): 1838–1853.

16. Hubballi N, Tripathi N. A closer look into DHCP starvation attack in wireless networks. *Computers & Security* 2017; 65: 387–404.

17. Michie D, Spiegelhalter DJ, Taylor C, others . Machine learning. *Neural and Statistical Classification* 1994; 13.

18. Praseed A, Thilagam PS. DDoS attacks at the application layer: Challenges and research perspectives for safeguarding Web applications. *IEEE Communications Surveys & Tutorials* 2018; 21(1): 661–685.

19. Niyaz Q, Sun W, Javaid AY. A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN). *CoRR* 2016; abs/1611.07400.

20. Ahmed ME, Kim H, Park M. Mitigating DNS query-based DDoS attacks with machine learning on software-defined networking. In: IEEE. ; 2017: 11–16.

21. Hu D, Hong P, Chen Y. FADM: DDoS flooding attack detection and mitigation system in software-defined networking. In: IEEE. ; 2017: 1–7.

22. Cui Y, Yan L, Li S, et al. SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks. *Journal of Network and Computer Applications* 2016; 68: 65–79.

23. Silva dAS, Wickboldt JA, Granville LZ, Schaeffer-Filho A. ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN. In: IEEE. ; 2016: 27–35.

24. Sultana N, Chilamkurti N, Peng W, Alhadad R. Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications* 2019; 12: 493-501.

25. Adeli H, Hung SL. *Machine learning: neural networks, genetic algorithms, and fuzzy systems*. John Wiley & Sons, Inc. . 1994.

26. Jia B, Huang X, Liu R, Ma Y. A DDoS Attack Detection Method Based on Hybrid Heterogeneous Multiclassifier Ensemble Learning. *Journal of Electrical and Computer Engineering* 2017; 2: 1–9.

27. UNSW-NB15 dataset. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/; 2017. [Online; accessed 19-October-2018].

28. CICIDS2017 dataset. https://www.unb.ca/cic/datasets/ids-2017.html; 2018. [Online; accessed 2-January-2019].

29. NSL-KDD dataset. https://www.unb.ca/cic/datasets/nsl.html/; 2017. [Online; accessed 10-September-2018].

30. Feamster N, Rexford J, Zegura E. The Road to SDN. *Queue* 2013; 11: 20:20–20:40.

31. Tourrilhes J, Sharma P, Banerjee S, Pettit J. Sdn and openflow evolution: A standards perspective. *Computer* 2014; 47: 22–29.

32. Wolpert DH, Macready WG. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1997; 1(1): 67–82.

33. Douglas PK, Harris S, Yuille A, Cohen MS. Performance comparison of machine learning algorithms and number of independent components used in fMRI decoding of belief vs. disbelief. *Neuroimage* 2011; 56(2): 544–553.

34. Breiman L. Random forests. *Machine learning* 2001; 45: 5–32.

35. Breiman L. *Classification and regression trees*. Routledge . 2017.

36. Kubat M. Neural Networks: A Comprehensive Foundation by Simon Haykin, Macmillan. *The Knowledge Engineering Review* 1999; 13: 409–412.

37. Liao Y, Vemuri VR. Use of K-Nearest Neighbor classifier for intrusion detection. *Computers & security* 2002; 21: 439–448.

38. Murphy KP. Naive Bayes classifiers. *University of British Columbia* 2006; 18: 60.

39. Rokach L. Ensemble-based classifiers. *Artificial Intelligence Review* 2010; 33: 1–39.

40. Cao J, Lin Z, Huang GB, Liu N. Voting based extreme learning machine. *Information Sciences* 2012; 185: 66–77.

41. Chen Z, Jiang F, Cheng Y, Gu X, Liu W, Peng J. XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-Based Cloud. In: 2018 IEEE International Conference on Big Data and Smart Computing (BigComp). IEEE. ; 2018: 251–256.

42. Ye J, Cheng X, Zhu J, Feng L, Song L. A DDoS attack detection method based on SVM in Software Defined Network. *Security and Communication Networks* 2018; 2018: 1–8.

43. He D, Chan S, Ni X, Guizani M. Software-Defined-Networking-Enabled Traffic Anomaly Detection and Mitigation. *IEEE Internet of Things Journal* 2017; 4: 1890-1898.

44. Zeng X, Wong DF, Chao LS. Constructing better classifier ensemble based on weighted accuracy and diversity measure. *The Scientific World Journal* 2014; 2014: 1–12.

45. Vimala S, Dhas J. SDN Based DDoS Attack Detection System by Exploiting Ensemble Classification for Cloud Computing. *International Journal of Intelligent Engineering and Systems* 2018; 11: 282–291.

46. Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: IEEE Local Computer Network Conference. IEEE. ; 2010: 408–415.

47. Jankowski D, Amanowicz M. On efficiency of selected machine learning algorithms for intrusion detection in software defined networks. *International Journal of Electronics and Telecommunications* 2016; 62: 247–252.

48. Ashraf J, Latif S. Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques. In: National Software Engineering Conference. ; 2014: 55–60.

49. Anand N, Babu S, Manoj B. On detecting compromised controller in software defined networks. *Computer Networks* 2018; 137: 107 - 118.

50. Diro AA, Chilamkurti N. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems* 2018; 82: 761–768.

51. Balkanli E, Alves J, Zincir-Heywood AN. Supervised learning to detect DDoS attacks. In: IEEE. ; 2014: 1–8.

52. Das S, Mahfouz AM, Venugopal D, Shiva S. DDoS Intrusion Detection Through Machine Learning Ensemble. In: IEEE. ; 2019: 471–477.

53. Singh PK, Jha SK, Nandi SK, Nandi S. ML-Based Approach to Detect DDoS Attack in V2I Communication Under SDN Architecture. In: IEEE. ; 2018: 0144–0149.

54. Tuan NN, Hung PH, Nghia ND, Tho NV, Phan TV, Thanh NH. A DDoS Attack Mitigation Scheme in ISP Networks Using Machine Learning Based on SDN. *Electronics* 2020; 9(3): 413.

55. Dietterich TG, others . Ensemble learning. *The handbook of brain theory and neural networks* 2002; 2: 110–125.

56. Dietterich TG. Ensemble methods in machine learning. In: International workshop on multiple classifier systems. Springer. ; 2000: 1–15.

57. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 2011; 12: 2825–2830.

58. Verma A, Ranga V. On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques.. *Pertanika Journal of Science & Technology* 2018; 26(3).

59. Verma A, Ranga V. Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. *Procedia Computer Science* 2018; 125: 709–716.

60. Moustafa N, Slay J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective* 2016; 25(1-3): 18–31.

61. Bansal A, Kaur S. Extreme Gradient Boosting Based Tuning for Classification in Intrusion Detection Systems. In: International Conference on Advances in Computing and Data Sciences. Springer. ; 2018: 372–380.

62. Singh R, Kumar H, Singla R. An intrusion detection system using network traffic profiling and online sequential extreme learning machine. *Expert Systems with Applications* 2015; 42: 8609–8624.

AUTHOR BIOGRAPHY

Rochak Swami. Rochak Swami is a PhD research scholar at the Department of Computer Engineering in National Institute of Technology, Kurukshetra, Haryana, India. She received her B.Tech degree (2012) in Computer Science and Engineering from Rajasthan Technical University, Kota, India and M.Tech degree (2015) in Computer Science from Birla Institute of Technology Mesra, Ranchi, Jharkhand, India. She is an ACM student member. Her current areas of interest include cyber security, security issues in Software defined networking, and DDoS attacks.

Mayank Dave. Prof. Mayank Dave received BTech degree from Aligarh Muslim University, Aligarh, India in 1989, MTech degree in Computer Science and Technology and the PhD degree from IIT Roorkee, India in 1991 and 2002, respectively. He is a Professor in the Department of Computer Engineering at National Institute of Technology Kurukshetra (NIT Kurukshetra), India with more than 27 years experience of academic and administrative affairs in the institute. He has published approximately 170 research papers in various international/national journals and conferences. He has coordinated several research and development projects in the department and institute. He has written a text book titled "Computer Networks" published by Cengage Learning, India. Prof Mayank Dave has guided fifteen PhDs and several M.Tech and BTech thesis and projects. He is currently guiding few more PhDs. His research interests include mobile networks, cyber security, cloud computing, web technologies etc. He is a Senior Member of the IEEE and the IEEE Computer Society, and member of the ACM, IETE, Computer Society of India, and Institution of Engineers (India).

Virender Ranga is Assistant Professor in the Computer Engineering Department at National Institute of Technology Kurukshetra, India. He received his PhD degree in 2016 from National Institute of Technology Kurukshetra, Haryana, India. He has published more than 80 research papers in various International SCI Journals as well as reputed International Conferences in the area of Computer Communications. He has been conferred by Young Faculty Award in 2016 for his excellent contributions in the field of Computer Communications. He has been acted as a member of TPC in various International conferences of repute. He is a member of editorial board various reputed journals like Journal of Applied Computer Science & Artificial Intelligence, International Journal of Advances in Computer Science and Information Technology(IJACSIT), Circulation in Computer Science (CCS), International Journal of Bio Based and Modern Engineering (IJBBME) and International Journal of Wireless Networks and Broadband Technologies. Currently, he has been selected Guest Editor for a special issue to be published in the International Journal of Sensors, Wireless Communications and Control (Bentham Science Publication). He is an active reviewer of many reputed journals of IEEE, Springer, Elsevier, Talyor & Francis, Wiley, and InderScience. His current area of interest include Wireless Sensor & Adhoc Networks, Internet of Things, Flying ad hoc networks, and Software Defined Networking.