Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.026952

ARTICLE

Detecting and Mitigating DDOS Attacks in SDNs Using Deep Neural Network

Gul Nawaz[1], Muhammad Junaid[1], Adnan Akhunzada[2], Abdullah Gani[2,*], Shamyla Nawazish[3], Asim Yaqub[3], Adeel Ahmed[1] and Huma Ajab[4]

[1]Department of Information Technology, The University of Haripur, Haripur, 22060, Pakistan

[2]Faculty of Computing and Informatics, University Malaysia Sabah, Sabah, 88400, Malaysia

[3]Department of Environmental Sciences, COMSATS University Abbottabad Campus, Abbottabad, 22010, Pakistan

[4]Department of Chemistry, COMSATS University Abbottabad Campus, Abbottabad, 22010, Pakistan

*Corresponding Author: Abdullah Gani. Email: abdullahgani@ums.edu.my

Received: 08 January 2022 Accepted: 22 April 2022 Published: 29 November 2023

ABSTRACT

Distributed denial of service (DDoS) attack is the most common attack that obstructs a network and makes it unavailable for a legitimate user. We proposed a deep neural network (DNN) model for the detection of DDoS attacks in the Software-Defined Networking (SDN) paradigm. SDN centralizes the control plane and separates it from the data plane. It simplifies a network and eliminates vendor specification of a device. Because of this open nature and centralized control, SDN can easily become a victim of DDoS attacks. We proposed a supervised Developed Deep Neural Network (DDNN) model that can classify the DDoS attack traffic and legitimate traffic. Our Developed Deep Neural Network (DDNN) model takes a large number of feature values as compared to previously proposed Machine Learning (ML) models. The proposed DNN model scans the data to find the correlated features and delivers high-quality results. The model enhances the security of SDN and has better accuracy as compared to previously proposed models. We choose the latest state-of-the-art dataset which consists of many novel attacks and overcomes all the shortcomings and limitations of the existing datasets. Our model results in a high accuracy rate of 99.76% with a low false-positive rate and 0.065% low loss rate. The accuracy increases to 99.80% as we increase the number of epochs to 100 rounds. Our proposed model classifies anomalous and normal traffic more accurately as compared to the previously proposed models. It can handle a huge amount of structured and unstructured data and can easily solve complex problems.

KEYWORDS

Distributed denial of service (DDoS) attacks; software-defined networking (SDN); classification; deep neural network (DNN)

1 Introduction

A network consists of a group of linked devices to share resources and information. In traditional networks, different servers at a remote distance are connected with the help of routers. A server conveys a message through these routers. A router is an intelligent device that calculated with the help of a routing table. These routing tables are prepared by different routing protocols and decide how to treat a packet within a network. A router can be divided into two major parts, i.e., the Control plane and the Data plane. A Control plane maintains the routing table and instructs the data forwarding plane. A control plane decides how to treat an incoming packet. A Data forwarding plane follows the instructions of its control plane. In a traditional network, every router makes its calculation on its own that increases the processing time. Deployment of new rules and security services in a network is very hard and time-consuming. To meet the business needs, networks are growing very fast that also increasing network traffic. A traditional network may face difficulty handling this huge amount of data. Software-Defined Networking (SDN) overcomes these problems. SDN separates the control plane and data plane of a switch or a router as shown in Fig. 1. It centralizes the control plane and switches become simple forwarding devices. It simplifies the management of a network and provides superior performance in large-scale networks [1]. In SDN, a controller receives packets from the data plane (switches) [2].

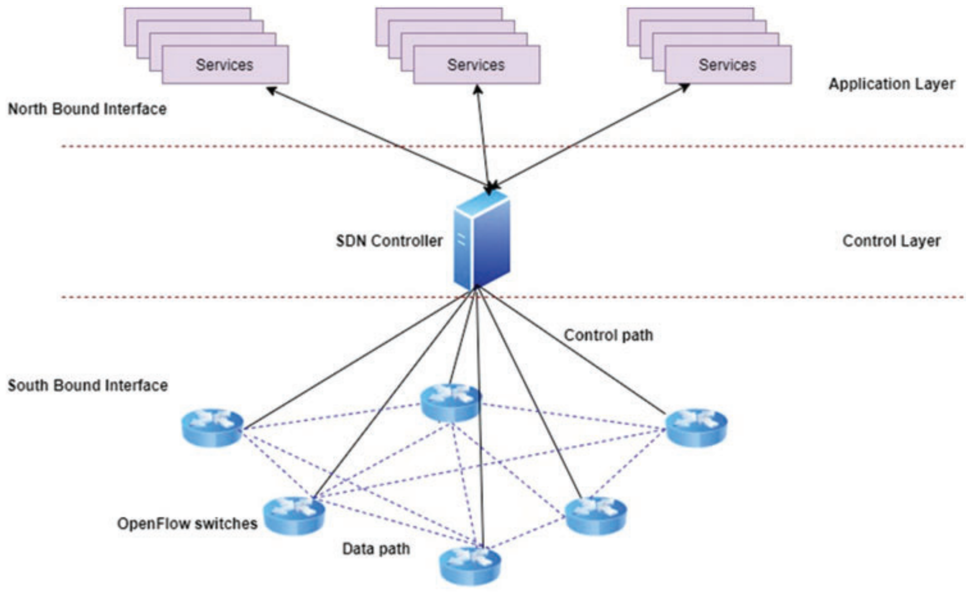

Figure 1: SDN architecture [3]

The controller sends back a flow rule to the switch and the switch will store a new entry for that particular flow. OpenFlow Protocol (OFP) [4] is used for communication between switches and controllers. Software-Defined Networking (SDN) helps to reduce traffic congestion. It is agile and easy to manage. Due to centralized and programmable controllers implementing new rules and security services become easy and cost-efficient. It simplifies the management of a network and provides superior performance in large-scale networks. Despite these benefits, there are also a few drawbacks to an SDN network. The open nature of SDN makes it a suitable victim for different kinds of attacks [5]. Conventional DDoS attacks are also viable in SDN. Distributed denial of service (DDoS) attacks is increasing rapidly over time. The DDoS attack is one of the most extensive attacks which are faced by different financial and public sector organizations nowadays. The size and complexity of the attack are increasing day by day because of different easily available tools. The main purpose of a DDoS attack is to attack the system and consume its resources by sending a huge amount of traffic so, a system can't respond to legitimate requests. These attacks are carried out by different compromised devices. Different compromised computers and devices send a huge amount of traffic to the victim machine at the same time. A victim machine will become obstructed and will not be able to entertain a legitimate user. In Dec 2016, a botnet attack (Mirai IoT botnet) was launched on a security blogger such that it brings down the whole system. This attack was comprised of different Internet of Things (IoT) devices [6]. A DDoS attack consumes all the resources of a system and exhausts the victim network by sending a huge amount of malicious traffic that results in the bottleneck of a network [7]. DDoS attacks are divided into three classes:

Volume-Based Attacks: Volume-based attacks consist of a huge amount of ICMP and UDP spoofed traffic to bring down the system. These kinds of attacks are measured in bits per second (bps).

Network Layer Attacks: These kinds of attacks occur while flooding a network infrastructure by sending a huge amount of SYN floods or smurf attacks. These kinds of attacks are measured in packets per second (pps).

Application Layer Attacks: These attacks occur on different applications of the network. These attacks are measured in requests per second (rps).

The architecture of the Software-Defined Networking (SDN) exposes it for DDoS attacks. At the control plane level, a single or many switches receive a huge amount of spoofed traffic by different IP addresses. The switches have no flow rules for these new packets so they send these packets to the controller for generating flow rules. So, the link between the controller and switches is obstructed and it will not be able to entertain a legitimate request. A switch memory can be targeted by the attacker by sending a large number of packets from different IP addresses to the switch. So, it will not be able to handle the traffic. In another scenario, a controller receives a large number of packets from different switches to a specific destination (victim). The controller sends a new flow rule for every packet and sends it to the victim. This will generate a huge amount of traffic towards the victim. A switch has limited memory and needs to store a new entry for each flow. So, this will cause the bottleneck of a switch memory.

For detecting DDoS attacks many Machine Learning (ML) approaches are proposed but Deep Learning (DL) outperformed existing ML techniques [8]. A Deep Learning (DL) technique is used in different fields of computer science nowadays. For example, for speech recognition and Natural Language Processing (NLP) LSTM is used widely. A Convolutional Neural Network (CNN) is used for image recognition and computer vision. Machine Learning (ML) models take a small number of packet features and when we increase the packet features the outcome result varies and accuracy decreases. A DL algorithm can take a large number of feature values and deliver high-quality results. DL model is also capable of creating new features by it and scans the data to find the correlated feature. Moreover, the DDoS attacks are considered one of the most dangerous attacks which slow down the SDN performance. Attackers hold on control plan of the routers which become difficult for the SDN to control the packets and control is handed over to attacks. The detection and mitigation of the attacks are very important to keep the SDN working with its packets management. The detection is not entirely enough so, we mitigate the DDoS attacks from the SDN network.

In this paper, we proposed a deep neural network (DNN) model for SDN to detect DDoS attacks. The proposed Developed Deep Neural Network (DDNN) model consists of an input layer, two hidden layers, and finally an output layer. We train our model with 12 kinds of different DDoS attacks. We choose the latest state-of-the-art dataset which consists of many novel attacks. More details of the dataset are described briefly in Section 5. Our model achieves 99.77% detection accuracy which is quite high as compared to other proposed models.

1.1 Contributions

The main contributions of the proposed DDNN approach are to classify the DDoS attack traffic and legitimate traffic with handling larger numbers of features from DDoS attacks to enhance the further detection of attacks using DNN approach in SDN. The contributions are.

- DDNN is a method which allows the system to detect DDoS attack based on detection and prevention for co-related features from the dataset.
- The research considers multiple attributes with different features associated with them, but we considered to enhance the Quality of Detection (QoD) requirements for every individual SDN packet. We selected the dataset from real SDN network which fulfills the research demands. Based on the dataset a DDNN Model for Classification Algorithm is proposed to select and classify the features for DDoS attacks for service requirements.
- A huge number of packets are considered to detect the DDoS attacks and find the DDNN model Accuracy, Loss, Precision, and F1-score.
- Finally, the ANOVA test is performed i.e., a parametric test involving two variables which finds that DDNN is an effective approach to classification, detection, and mitigation of the DDoS attacks.

The rest of the paper is organized as follows. Section 2 describes the literature review. The theoretical concept of SDN is discussed in Section 3. Section 4 presents the material required for our proposed Deep Neural Network (DNN) model. In Section 5 the experimental setup and in Section 6 the results are discussed and compared with different parameters. CICDDoS2019 dataset [9] is also discussed in this section that is used to train our model.

2 Literature Review

In [10], Stacked Auto-encoder (SAE) is an unsupervised Deep Neural Network (DNN) model that consists of multiple layers of sparse auto-encoders. In the DNN model output of the previously hidden layer is connected with the input of the next hidden layer. So, the model is trained with the help of these hidden layers. The Backpropagation algorithm is combined with SAE to minimize the weights and cost function of the model. As result, we can achieve good results and fine-tuned training model. The Sigmoid activation function is used at every layer. The Sigmoid function is one of the most used functions in Deep Learning (DL) models that can map the values in the 0, 1 range. The technique accurately detects different kinds of TCP, UDP, and ICMP attacks. A home wireless network with twelve (12) wireless devices (interconnected with each other) is used for normal and malicious traffic collection. The network collects 74 h of traffic data out of which the last 24 h' data is containing normal and Malicious flows. DDoS attack detection application is installed on a Python-based POX controller is used in the SDN. An interval time of 60 s is set to trigger the Feature Extractor (FE) module. FE module extracted the features of the incoming packets from Traffic Collector (TC). After feature extraction SAE model classifies the incoming traffic into legitimate or malicious traffic. Feature extraction is carried out on every individual host in the network therefore after an attack from a particular host; the controller will send a flow rule to the switch to block that host. In [11], three Machine Learning (ML) techniques i.e., Naive Bayes, Support Vector Machine (SVM), and Neural Network (NN) are used for the detection of DDoS attacks in SDN networks. Naive Bayes is a supervised probabilistic technique. It calculates the probability of an incoming packet and compares it with the given classes and classifies the incoming traffic accordingly. Support Vector Machine (SVM) a supervised classification ML technique learns the pattern of the given data and performs a linear separation. A simple Neural Network (NN) algorithm is also used for classification. The proposed NN model has an input layer, a single hidden layer, and finally an output layer. Weights are given to each input of the model. These weights play a vital role in classification. For training, the model undergoes 10,000 epochs. The Sigmoid function is used to classify the traffic into two classes, i.e., malicious and legitimate traffic.

Four different types of models are used in [12] on a self-generated dataset. Machine Learning (ML) approach is used widely to detection of attacks in a network. Because of continuous model improvement and pattern identification, the ML algorithm is widely used for classification. Four different ML algorithms are used and compared with each other. TCP and ICMP flood attack data set is collected using "hping3" program. For capturing the incoming packets Tshark is used and saved into a pcap file. The pcap file is converted into Comma-separated values (CSV) format for packet features extraction. After data preprocessing the proposed four ML algorithm is applied to the data set and results are compared. Support Vector Machine (SVM) algorithm results with better accuracy as compared to other models but, it also takes a longer time at the training phase than the other proposed model. If the model detects a malicious flow, it invokes the mitigation process of the model. The mitigation script sends a REST message to the controller. The controller will block the attack ports on the OpenFlow (OF) switch for thirty (30) seconds. In [13], with the help of self-generated dataset different kinds of TCP, UDP, and ICMP attacks are collected for training the model. Three different types of feature selection methods are applied to overcome the overfitting problem. Different algorithms are tested and finally K-Nearest Neighbors (KNN) machine learning algorithm chose as a better classifier than others. The model's accuracy depends on features selection as it takes only six features to train the algorithm. The accuracy of the model decreases as we increase the feature size from eight to ten. An advanced support vector machine (ASVM) algorithm is used in [14]. A new dataset is used for training the model. However, the model covers only volume-based attacks (i.e., UDP and SYN flood attacks). The accuracy of the model varies as we change the test and train split rate.

In [15], due to high-quality results and low memory consumption Support Vector Machine (SVM) is used widely for attack detection and classification. DARPA dataset is used for training the model and results are compared with existing DDoS detection models. Mostly, SVM is used for binary classification, but we can extend its functionality to predict multiclass problems. A multiclass SVM classifier was developed that gives a low false-positive rate and results in high detection accuracy. The proposed model is compared with other widely used Machine Learning (ML) algorithms as well. The model also classified different DDoS attacks correctly, but it also takes more time to train our model to detect a DDoS attack. The concept of classification using SVM has been deeply discussed in [16–22] wherein, various formats of data and security issues are addressed in detail. In [23], SYN flood attacks occurred during TCP handshake. TCP connection establishment involves in three-way handshake. In an SYN flood attack, an attacker sends an SYN request with a spoofed IP address to the network server. The server sends back an SYN/ACK packet to that particular IP address. Now the server is in a half-open state, and it will be waiting for an acknowledgment from the spoofed IP address. This consumes the memory and resources of the server. The ISDSDN method is based on the intentional dropping concept. This method also eliminates the TCAM switch overloading as it does not install any flow rule to the switch before filtering an incoming packet. The proposed ISDSDN model is configured to detect and mitigate an attack after receiving 10 malicious packets. It might vary the detection time. The proposed model works efficiently in Mac address spoofing as well.

In [24], a technique is proposed which is a combination of two models, i.e., Claude Shannon's entropy and key observations using data flow entropy (extracting clues from the entropy analysis). The model used only a single victim destination node because of entropy based solution limitation. Simulation of the proposed model is done with synthetic traffic which is less randomized as compared to realistic traffic and the false positive rate increased as we increased the traffic. In [25], time-series analysis deals with the data that is in a series of intervals. It is a static analysis technique. The proposed algorithm monitors incoming traffic on every single switch of the SDN network and extracts pre-defined features. USIP algorithm is used for anomaly detection and UNDIP algorithm is used for DDoS attack detection. Both algorithms are used for labeling the incoming packets. For the dataset, real network traffic was collected with the help of MAVILab. After detecting a DDoS attack, an attack alarm is raised for that specific OpenFlow (OF) switch. The mitigation module adds a flow rule and sends it to that specific switch. The proposed model detects anomalies in the switch of an SDN network efficiently with high accuracy and low false-positive rate. In [26], WEKA is a simulator used for different machine learning algorithms. WEKA can perform many tasks including classification, clustering, visualization, and regression for a given data. A virtual SDN network is created with the help of the Mininet emulator. Nine most relevant features are extracted from the packets and seven Machine Learning (ML) algorithms are applied in the WEKA simulator. Among these seven ML algorithms, the AdaBoost model efficiently detects and classifies different kinds of ICMP, ARP, UDP, TCP, and SSH DDoS attacks. The model detects and classifies these attacks in different DDoS attack classes that provide extra security to the controller and provide efficient firewall rules.

In [27], authors presented a Software Defined Networking Resilience Management for Bandwidth (SDN-RMbw) framework, to find the fault and resilient protection against the profound bandwidth. Two events are detection which are run time changes and link failure. These events are bound to be performed for the final analysis. Another approach in [28], worked on Controller Area Network (CAN) for detection of In-Vehicle Intrusion attacks using CNN on CAN bus. The proposed approach detects the single intrusion attacks on network and mixed intrusion attacks on CAN bus. The results shows that the proposed technique effectively enhance the results by 10.79% of the proposed approach. Another approach [29], authors proposed a intrusion detection system for system analysis. The training and efficiency of the network performance is proposed and provide the ML based model. The deadlocks are originated for the provision of the results and enhance the scenario for the proposed approach with efficiency of the network parameters. The approach uses a dynamic expectation function that computes the number of client models expected in each round and a weighted averaging algorithm for continuous modification of the global model. The detection system improves the performance of the system with results enhancements of around 99.5% with accuracy.

The existing literature identifies gaps to fulfills the requirements of the proposed methodology and research adoption. The existing research does not correct handle TCP, UDP, ICMP, IMAP, Miscellaneous Apps, HTTP, Random IP, SQL Injection, SNP, PORT, TFTP, and SYN. The proposed DDNN approach handle these parameters effectively and identify the current research gap which fulfills the research contributions in the proposed scenario. The previous proposed techniques does not take into account for larger number of feature but proposed DDNN approach work on these features to justify the research gap.

2.1 Background

2.1.1 Software-Defined Networking (SDN)

In traditional network different servers at a remote distance are connected with the help of routers and switches. These routers and switches have two different levels of abstraction, i.e.,

1) Control Plane

The Control plane implements the routing functionalities and the construction of the routing table and its management.

2) Data Plane

Data plane forward a packet by looking into the destination IP field in the IP header and making a match with the routing table inside the interface that is the forwarding information base, and then forwards the packet to the outgoing interface. Traditionally these control and data forwarding functionalities are implemented in every single router in a network. Implementation of these functionalities in a single router or switch makes its complexity higher because we have multiple routers with their control planes in a network and these control planes need to coordinate with each other to generate the global routing table or to manage the global routing table [30]. Because of the limitations of traditional routing architecture these control planes need to be performed in a distributed way [31].

With this distributed controlled architecture, our routing protocols get problematic as we have seen both distance vector and the link state routings have significant limitations in terms of their scalability, distance vector routing table cannot get scalable because of this count to infinity problem, whereas the link-state routing protocol that cannot get scalable because of its size of the link-state packets or the size of the link-state information that we need to implement it over a large network. Because of this limitation, we have restricted this link-state routing and the distance vector routing within a local internet or a subnet, and for a network to network, we have this border gateway protocol that implements the policy. This may be difficult for the Network managers because, if there is a policy change then he needs to update every individual router and all the routing protocols in all the router's control planes are needed to get coordinated with each other to make a policy update at the individual router. Obviously, in a distributed architecture it will take time. Because of these time requirements, there can be inconsistencies across the routers and these inconsistencies can get signed in a large network. So, that is why managing a subnet with huge numbers of routers becomes a very difficult task. This also makes difficult the compatibility among the different vendors. Routers may have different models their configuration options may be different (for example CISCO IOS manual comes with 5000 pages' document) so, the management functionalities become very complex. With this distributed architecture, maintaining consistencies at the control plane of routers of a network becomes difficult and that is why we gradually try to move from distributed control plane architecture to centralized control plane architecture. That is the basic motivation behind the design of a Software-Defined Networking (SDN) concept.

Software-Defined Networking (SDN) centralized control plane. The control plane is like the brain of a router. It makes the decisions and the TCAM hardware just makes a forwarding processing. So, the centralized control plane takes all decisions and acts as an instructor. The control plane makes a routing algorithm for the network traffic. A data plane is a module in SDN that carries out the tasks given by the controller and simply forwards a packet. So, in a traditional network, the network devices are proprietary in which the vendor decides software (control plane) and Hardware (data plane) and there is no standardization where every vendor applied their optimization so managing interoperability among different vendor routers becomes difficult in a large network. Further, in the Software-Defined Networking (SDN) paradigm, the vendor provides only hardware (data plane) where we decide the control plane by writing custom logic- the software and the control plane will be decided by the application designer or the network manager. In SDN, the switch just has the TCAM hardware along with the forwarding engine, the control logic is not there. And we will implement our control logic ourselves [32]. There are certain advantages exhibited by the SDN such as:

- Features are no longer limited to what a vendor provides.
- In an open-source movement, people can come together and design new protocols and implement it on a control plane itself.
- Longer life of a product.

2.1.2 SDN Architecture

Open networking foundation (ONF) presents a high-level SDN architecture that is comprised of the following three layers.

1) Application Layer

The application layer (also known as a management plan) is consists of end-user applications. These applications use APIs which are provided by the controller to program the switches for different purposes. Applications communicate with the controller through the northbound API [33].

2) Control Layer

The control layer consists of different controllers that maintain a centralized view of an SDN network. A controller is the brain of an SDN network. It translates the data from the application plane to switches. OpenDaylight, Project Floodlight, POX, and RYU are some examples of SDN controllers. The most prominent protocol used for southbound API is the OpenFlow (OF) Protocol. The OpenFlow (OF) Protocol acts as a bridge between a controller and a switch [34].

3) Infrastructural Layer

The infrastructure layer (control plane) consists of switches. These switches receive flow rules from the control plane and simply forward the packets according to the instructions. A switch interacts with the controller through southbound API [35].

3 Research Methodology

Our DDoS detection technique requires to gather certain parameters from the network in normal scenario and in DDoS attack scenario. The main features and assumptions that we have considered in this research are:

- The normal operations exist in the network are consistent. Every node which exchanges data has profile to show complete working. This enables our technique to detect and transform our anomaly detection and prevention mechanism as shown in Fig. 2.
- The training of the control layer is done through on-line mode. The model is exported and used by the controller to get extra benefits for detection of DDoS attacks.

3.1 Mininet

Mininet is a network emulator [36] that allows creating a network virtually. We can create Traditional/OpenFlow (OF) switches, hosts, and controllers in Mininet. Usually, we create an SDN virtual network with the help of Mininet. It helps us to learn an SDN network and its applications. By using the Minenet emulator we can create a range of networks from a simple network to a complex network consisting of thousands of hosts, hundreds of switches, and controllers. Different applications can be developed and tested on it. These applications can be moved to real-life network infrastructure. Mininet is an inexpensive and easily configurable tool as compared to other hardware testbeds.

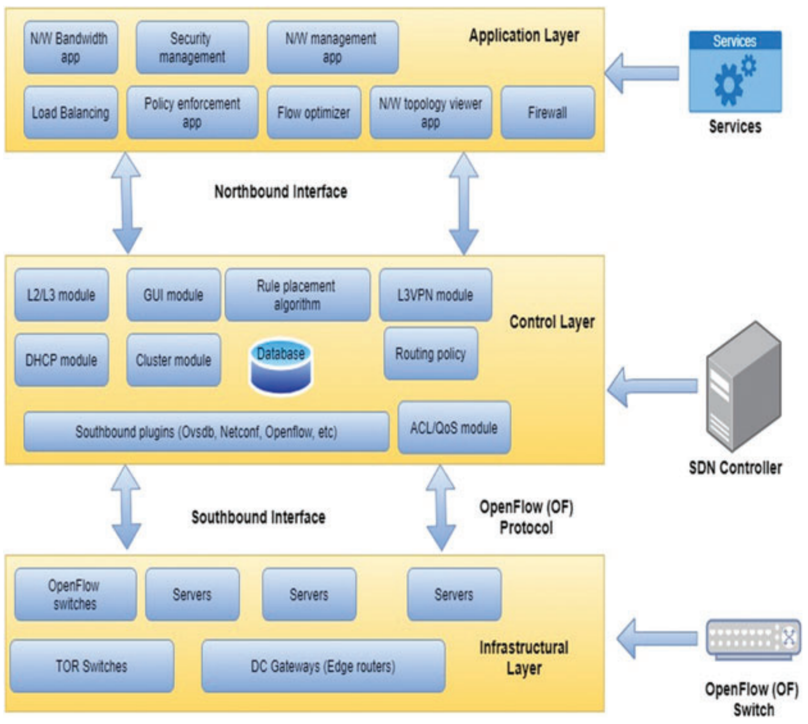

Figure 2: Three-layer SDN architecture

In Mininet, we use Miniedit GUI [37] for creating a virtual SDN network. It has a simple user-friendly interface. A controller is like a brain of an SDN network. A controller communicates with all the components of an SDN network. It communicates with the switches with the help of Southbound API and with the application layer with the help of Northbound API. We use the Ryu controller in our proposed topology. Ryu is an open-source Python-based framework, used as a centralized controller in an SDN network. The Ryu controller helps us to create and control new applications easily. A network administrator can easily manage the whole network and manage it with the help of the Ryu controller. The Ryu Controller is maintained by the open Ryu community. The Ryu controller uses different protocols to communicate with the data plane, including the OpenFlow (OF) Protocol [38].

3.2 *Deep Neural Network (DNN)*

Neural Networks (NN) form the base of deep learning a subfield of Machine learning. In the Neural Networks algorithms are stimulated by the anatomy of the human brain. Neural Networks (NN) instruct the algorithms on certain data. So, it can perceive the patterns and then anticipate the outputs for a new set of kindred data. It is one of the most preferable topics in the research area. From speech recognition and face recognition to healthcare and marketing neural networks have been used in a varied set of domains. An Artificial Neural Network (ANN) is the functional unit of Deep Learning (DL) [39]. Deep Learning (DL) uses Artificial Neural Networks (ANN) that mimic the behavior of the human brain to solve composite problems. Now Deep Learning (DL) itself is incorporated of Machine Learning (ML) that lay under the larger umbrella of Artificial Intelligence (AI). Artificial Intelligence

(AI), Machine Learning (ML), and Deep Learning (DL) are interconnected fields where ML and DL aids AI by providing a set of algorithms and Neural Networks (NN) to solve data-driven problems.

Deep Learning (DL) uses Artificial Neural Networks (ANN) that behave similarly to the neurons in a brain. A Neural Network (NN) functions when we give it some data. Then this data is refined and processed via a layer of perceptron to induce the desired output. The input data is in the form of matrices. Then this data is forwarded to the first layer of the NN called input layer. Our brain has neurons that assist us in connecting thoughts. In the same way, an ANN network has perceptron that get inputs from these matrices and process them by passing them on from the input layer to the next hidden layers. After many backpropagations and calculation finally, the data is sent to the output layer. At the early stage, when the input passes from the input layer to the hidden layer, some initial random weights are assigned to each input data. The multiplex of the incoming data and their corresponding weights are summed up. Their sum is further processed through the network [40,41]. So, we assigned numerical values called bias to each perceptron. Furthermore, every single perceptron is passed through a function known as the activation function. The activation function decides that whether that specific perceptron might get activated. A triggered/activated perceptron is utilized further to send out the data to the next upcoming layer. In this manner, our desired data is passed on forward through the network till the perceptron reaches the last layer called output layer. At the output layer, a possibility is derived that determines the data classification. In a condition that the predicted output is wrong, then in this situation, the NN will be trained by utilizing the back-propagation technique. In the first instance, while scheming a neural network, we modify the weights to every individual input with some arbitrary values. These weights represent the significance of each input variable.

Consequently, if we propagate rearward in a neural network and differentiate the true output to the forecasted output, we can re-adjust the weights of each incoming data. This will help us to lessen the errors. As a consequence, we get more faultless output. With the help of Deep Learning (DL) techniques, Google can translate between many different human languages' straightaway. One of the interesting applications of DL is visual translation. Perhaps it can be used to identify letters within images. After identification letters in an image, a DL algorithm can translate the same within an image and show the same image having translated text. Automated self-driven cars are yet another example of DL. DL has played an enormous initiative in the domain of self-driven cars. A neural network makes these cars perfect and error prone.

3.3 Proposed Developed Deep Neural Network (DDNN) Model

Our proposed methodology is comprised of modules as shown in Fig. 3. 1) Traffic Collector (TC) 2) Feature Extractor (FE) and Traffic Classifier (TC) and 3) Mitigation Rule. When an SDN controller receives a packet, Traffic Collector (TC) module collects the traffic from the controller. Feature Extractor (FE) module extracted the desired features from the packet and saved them in a CSV file. We selected twenty (20) packet features for different kinds of DDoS attacks. Traffic Classifier (TC) classifies the incoming traffic into malicious or legitimate packets. We used a supervised DNN model which provides discriminating power for classification problems. Our model is comprised of an input layer, two hidden layers, and finally an output layer. Each layer has a different number of neurons. The output result of the previous layer becomes the input of the next layer and so on. Every input has an adjustable parameter known as weights. Two operations are performed at every neuron, i.e.

3.3.1 Pre-Activation

At this stage, we do aggregation of the parameters (i.e., Inputs, weights, and bias).

$$ai\,(x) = (Wi\,(hi - 1)\,(x)) + bi \tag{1}$$

where $hi - 1\,(x) \in R^{n*n}$ is the i^{th} input or hidden layer where $n * n$ are the input dimensions. $W_i \in R^{n*n}$ is the dimensions of the weights, and $b_i \in R^n$ is the bias dimension. The Bias is an additional parameter used to adjust the output along with the weighted sum of input neurons. The first input layer can be called the 0^{th} input.

3.3.2 Activation

After aggregation, we applied an activation function. The activation function activates or deactivates a neuron according to weights.

$$hi\,(x) = g\,(ai\,(x)) \tag{2}$$

This activation function can be Sigmoid, ReLu, tanh, etc. The output of each layer is computed and an activation function is applied then this output becomes the input of the next layer. At the final output layer

$$f\,(x) = O\,(aL) \tag{3}$$

where $f\,(x) = h_L,\ h_L$ is the final output and O is the activation function at the output (i.e., Sigmoid, Softmax, etc.). The Sigmoid function will always clamp the output 0 to 1. The output layer is denoted to $L \in R^k$ where k is the dimension of the output. In our example k = 2. After getting the final output we calculate the loss function and backpropagate through the model to update the weights and reduce the difference between the expected result and the actual result. We predicted the model loss with the help of Cross entropy.

$$\text{Cross entropy} = (l\,(\theta)) = -\sum p_i \log\,(q_i) \tag{4}$$

where, $pi = \textit{True distribution}$ and $q_i = \textit{predicted distribution}.$

After getting the updated weights the forward propagation takes place again. This forward and backward propagation takes place a specific number of times (called epochs) to minimize the cost function of actual and predicted outputs. We used different numbers of epochs and observed the output of the model accordingly. We observe that the results improved as we increase the number of epochs gradually.

Algorithm 1: Proposed DDNN Model for Classification

```
Input: Incoming packet from the network
1.    for each active port of the OF switch in the network do
2.        OF switch ← Ryu controller (request for flow rule)
3.        Collect the flow statistics
4.        Extract features using the model
5.        for Classify the packet features with the help of DDNN classifier
          do
6.            Classification Process starts
7.            If (The DDNN predict a DDoS attack)
```

(Continued)

Algorithm 1 (continued)

```
8.              Take mitigation
9.              End If
10.           End do
11.       End for loop
12.   End for loop
```

After classification, if the incoming packet from any port is classified as an attack packet our proposed model will send a mitigation flow rule towards the Ryu controller as shown in Fig. 3. This mitigation flow rule instructs the controller to block that port for a specific time duration.

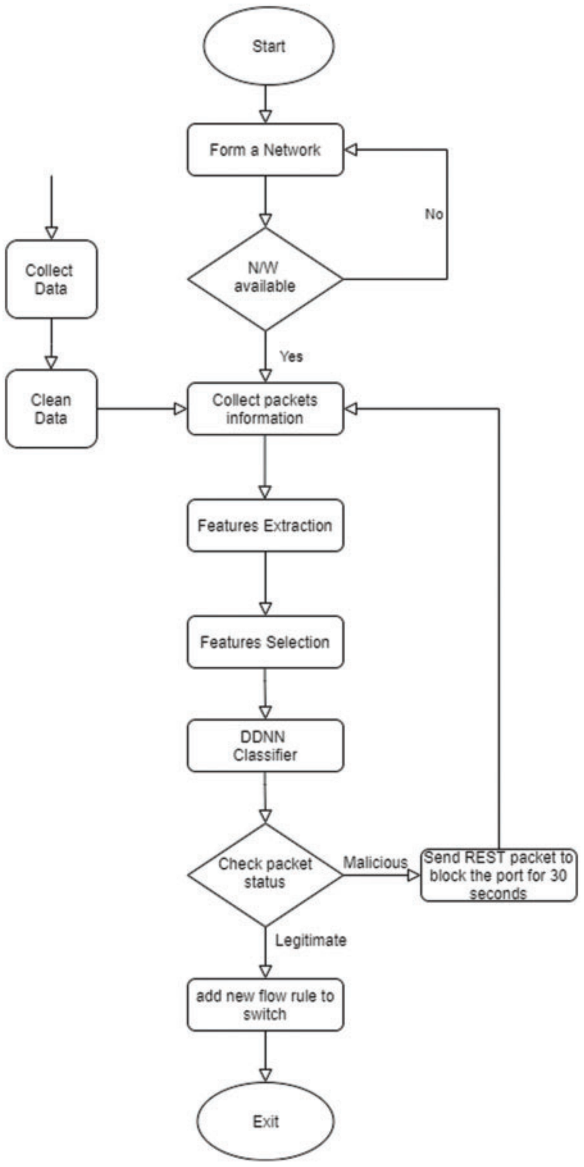

Figure 3: Flow diagram of the proposed model

3.4 Detection and Mitigation Mechanism

The model is designed and exported after evaluation. The applications which are designed to detect and mitigate DDoS attacks runs on the controller. The detection time is measured with application when DDoS is intact with SDN controller. If the model detects the DDoS, it feeds the output into its defense logs. After then a REST packet is sent to block the port for 30 s. The defense DDNN classifier built on the top of the SVM classifier, a model implementation which match the packets as accepted, altered, detect, dropped, and given a mark-based detection. The rules are designed to setup with marked output. The rules are fixed using SVM classifier and does not fit into its consideration. Based on this we provide an automatic defense mechanism to match DDoS packets and provide output based on detection algorithm. The packets matching for DDoS detection is automatically set based on features of the dataset. The mitigation mechanism is automatically set by defending and mitigating the DDoS attacks.

4 Experimental Setup

4.1 Dataset

For the evaluation of our model, we choose the latest and up-to-date data set CICDDoS2019 [42]. The data set generates realistic background traffic and overcomes all the shortcomings and limitations of the existing data sets. To our knowledge, the data set is not used in any kind of SDN DDoS attack detection technique before.

4.2 Data Preprocessing

For generating the realistic traffic two kinds of traffic captured during the traffic capturing period, i.e., a) benign traffic and b) attack traffic and saved in a pcap file. Different Machine Learning (ML) models were used to extract important packet features from the pcap file. After examining the performance of the features, we extracted eighty (80) features with the help of the CICFlowMeter software [43] for all benign and DDoS attack packets. Then we use Principle Component Analysis (PCA) algorithm [44] for packet features reduction and avoiding overfitting. With the help of PCA, we choose twenty (20) most important features and train our model to detect eleven (11) kinds of DDoS attacks. Table 1 shows the details of the CICDDoS2019 data set generated attacks and compares them with the previously proposed models. In previously proposed papers attacks are generalized in their basic categories like TCP attacks, ICMP attacks, and UDP attacks. We grouped them into their basic categories. We also make separate columns for the attacks which are only proposed in the CICDDoS2019 data set to distinguish them from the previous work. We extracted twenty (20) features from the data set and use these features for training. Table 1 shows the extracted features with details. Selected features are taken as input nodes of the model. We have used the normal initializer and the ReLu activation function for hidden layers. For the output layer, the Golorot-uniform initializer and the Softmax activation function is used. We use the Mini-batch gradient descent (batch size = 500) with the RMSPROP optimizer. Our training model undergoes several epochs. In the results section, we show the results of forty epochs. We notice that the results improved as we increase the number of epochs. We also compare our results against the number of epochs in Table 2.

Table 1: Survey of various classification-based approaches

Reference	TCP	UDP	ICMP	IMAP	Misc. apps	HTTP/ Secure web	Random IP	SQL injec- tion	SNP	PORT MAP	TFTP	SYN
DDNN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
[8]	✓	✓	✓	✓	✓	✓		–	✓	–	–	✓
[12]	✓	✓	✓	–	–	–	✓	–	–	–	–	–
[13]	✓	✓	✓	–	–	–	–	–	–	–	–	–
[14]	–	✓	–	–	–	–	–	–	–	–	–	–
[15]	✓	✓	✓	–	✓	✓	✓	✓	✓	–	–	✓
[16]	–	–	–	–	–	–	–	–	–	–	–	✓
[24]	–	–	–	–	–	–	–	–	–	–	–	✓
[25]	–	–	✓	–	–	–	–	–	–	–	–	–
[26]	✓	✓	–	–	–	✓	–	✓	–	–	–	✓
[27]	✓	✓	✓	–	–	✓	–	–	✓	–	–	✓

Table 2: Accuracy per epoch

	Number of epochs	Accuracy
1	10	99.70%
2	20	99.73%
3	40	99.77%
4	80	99.78%
5	100	99.80%

To build our model we use the Keras library with the Tensorflow Backend [45]. Linux system installed on Virtual box on-Intel Core i5-2450 M CPU, 2.5 GHz processor, 4 GB memory with OS windows 10. We use the Mininet emulator for creating a virtual SDN network. Ryu (Python-based) open-source controller is used as a centralized controller of the network. For features extraction, we used twenty (20) most important packet features and added twenty (20) columns shown in Table 3. In the Mininet emulator, we created a topology with the help of Miniedit Graphical User Interface (GUI) containing three different switches and connected it with the Ryu controller. A total of nine (9) hosts were created and every switch is connected with three (3) hosts. First of all, we open the h7 interface in xterm and generate normal traffic from h1 to record the normal traffic and our model behavior towards the normal traffic. Now we flood the data traffic with the help of the "hping3" command from h2 towards h1 (victim). The detection script in the Ryu controller uses our proposed Developed Deep Neural Network (DDNN) model to classify the incoming packets into normal and malicious flow. Our proposed model will capture the attack packets and will add mitigation flow rules. After detecting malicious packets, at the mitigation stage, our model will send a Rest message towards the controller that will instruct the controller to block that specific port of the OpenFlow (OF) switch.

Now if we ping h1 from another host then it will reply to the host without any delay in traffic. It means that detection and mitigation are done successfully and h1 can respond to any other legitimate port.

Table 3: Extracted features of the packet

Sr.	Feature	Feature description
1	Destination port	Destination port number
2	Protocol	Protocol identification
3	Flow duration	Total duration of a flow
4	Length of Fwd packets	Length of forward packets
5	Fwd packet length max	Max length of forwarded packet
6	Fwd packet length min	Min length of forwarded packet
7	Flow IAT mean	Mean flow inter arrival time of a packet
8	Flow IAT min	Min packet flow inter arrival time
9	Fwd IAT total	Total time between two packets sent forward
10	Fwd IAT mean	Mean time between two packets sent forward
11	Fwd IAT max	Max packet flow inter arrival time
12	Fwd header length	Forward packet's header length
13	Fwd packets/s	Numbers of forwarded packets
14	Min packet length	Minimum size of a packet
15	Max packet length	Max size of a packet
16	ACK flag count	Acknowledge flag count
17	Average packet size	Average size of a packet
18	Fwd header length.1	Forward packet's header length
19	Subflow Fwd bytes	Numbers of forward bytes in a sub flow
20	Min seg size forward	Maximum forward segment size

After classification, if the incoming packet from any port is classified as an attack packet our proposed model represented in Fig. 4 will send a mitigation flow rule towards the Ryu controller. This mitigation flow rule instructs the controller to block that port for a specific time duration.

5 Results and Discussion

Due to the fast arrival nature of a DDoS attack, we have taken 537,723 packets of different DDoS attacks and 23,385 normal packets. We split the data set and take 90% packets for training and the remaining 10% is used to test the model. Selected features are taken as input nodes of the model. We have used a he-normal initializer and the ReLu activation function for hidden layers. For the output layer, the Golorot-uniform initializer and the Softmax activation function are used. We used the Mini-batch gradient descent (batch size = 500) with the RMSPROP optimizer. Our training model (Fig. 3) undergoes several epochs. In the results section, we have shown the results of forty epochs. We noticed that the results improved as we increase the number of epochs shown in Fig. 5. We also compare our results against the number of epochs in Table 2. The performance is evaluated based on accuracy, precision, recall, and F1-score. We test our model with the following three parameters.

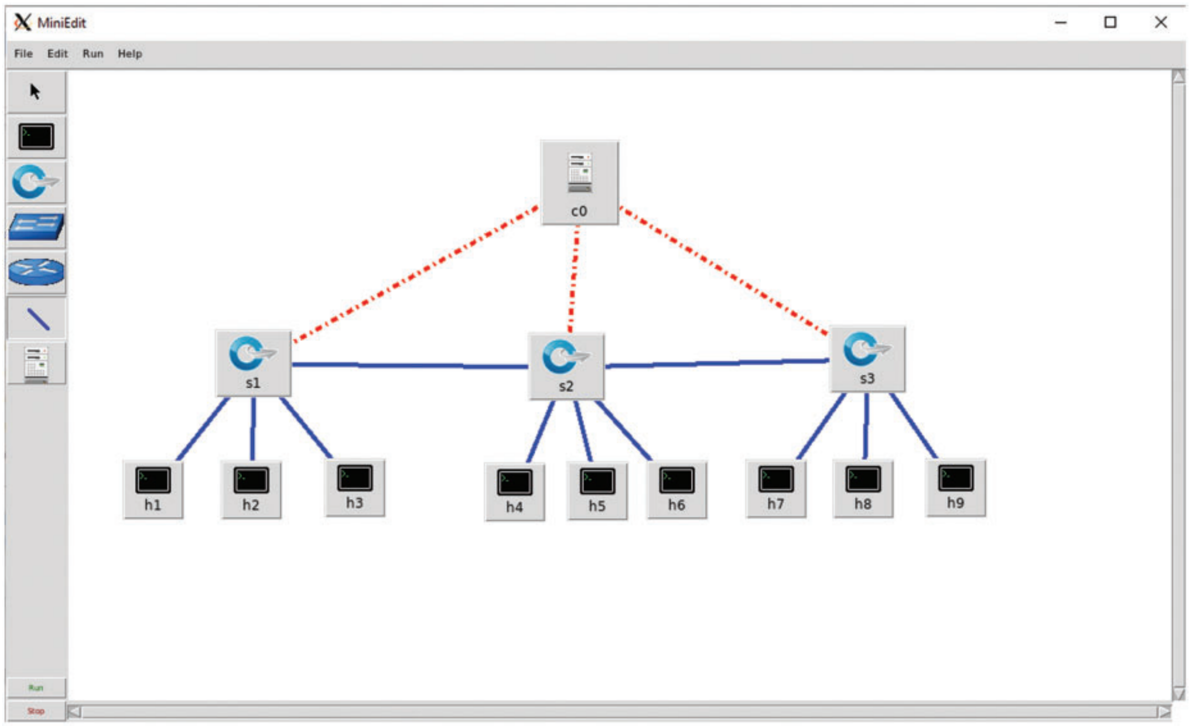

Figure 4: Proposed topology

5.1 Accuracy

The accuracy of the model results in 99.76% that is quite high as compared to previously proposed models shown in Fig. 6.

$$Accuracy\ (A) = \frac{Accurately\ classified\ record}{Total\ sample} \times 100 \tag{5}$$

5.2 Loss

After ten (10) epochs the loss of the model becomes as low as 0.0074% and after forty (40) epochs the loss reduces to 0.0065% shown in Fig. 7.

5.3 Precision

Precision is the estimation of accurately predicted results out of the entire predicted outcome shown in Fig. 8.

$$Precision\ (P) = \frac{True\ Positive}{True\ Positive + False\ Positive} \times 100 \tag{6}$$

5.4 Recall

Recall estimates the accurately predicted results over the entire outcome of a class.

$$Recall\ (R) = \frac{True\ Positive}{True\ Positive + False\ Negitive} \times 100 \tag{7}$$

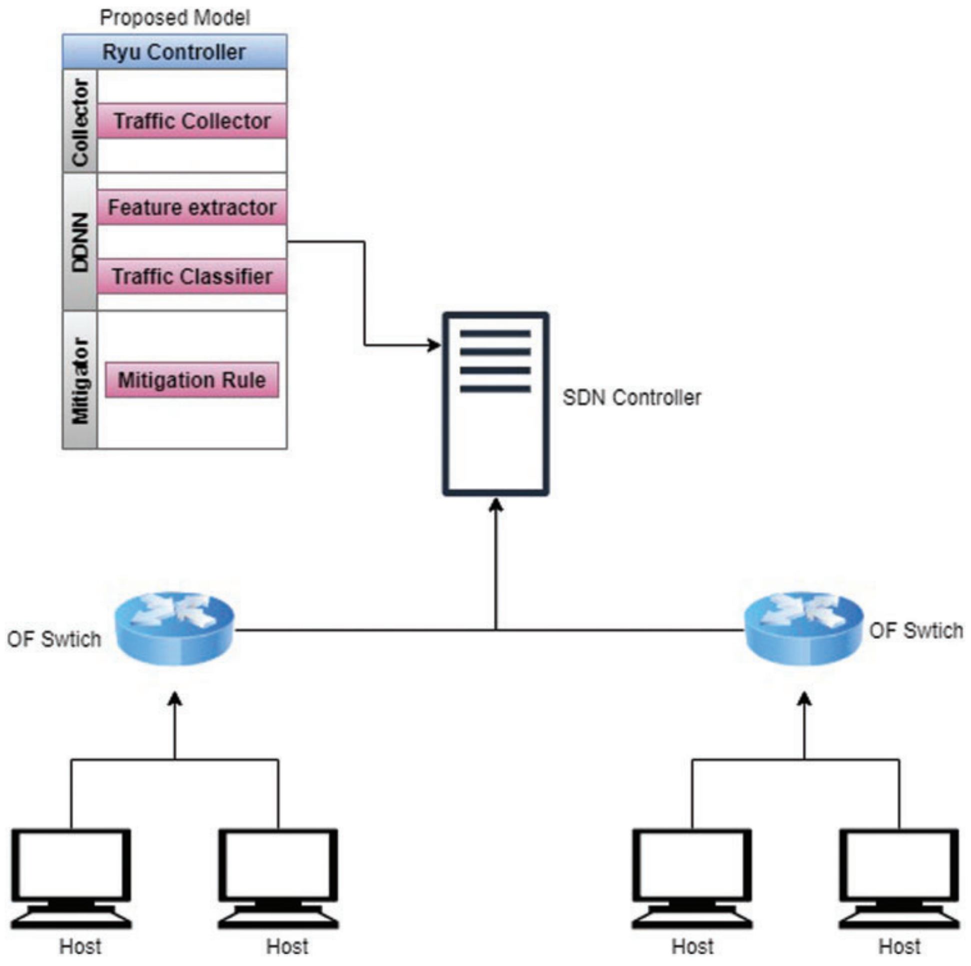

Figure 5: Proposed DDNN model

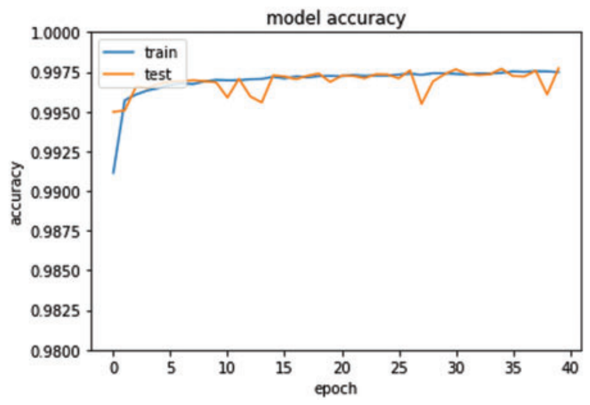

Figure 6: Model accuracy

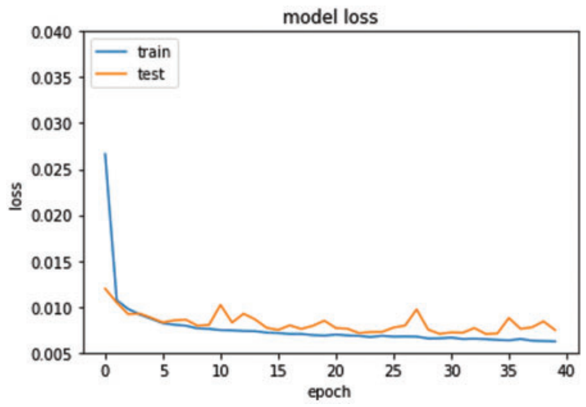

Figure 7: Model loss

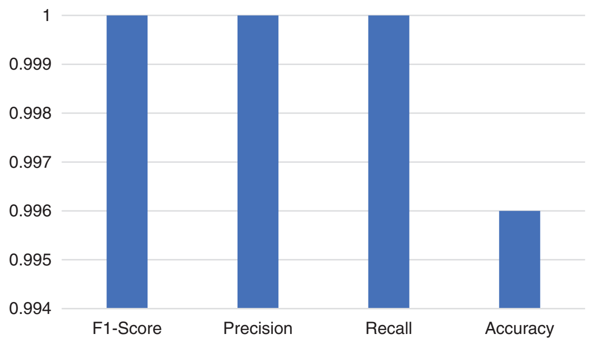

Figure 8: Accuracy, precision, recall, and F1-score of the model

5.5 *F1-score*

For calculating the F1-score, we use precision and Recall for the comprehensive estimation of the model. It is represented the consonant mean of both of these.

$$\text{F1} - \text{Score} = \frac{2\,(\text{P} \times R)}{(\text{P} + R)} \times 100 \tag{8}$$

5.6 *Statistical Analysis*

We also tested all metrics' resulting values and found that their distribution is normal. Within this case, a parametric test involving two variables is needed since we have taken one baseline at a time and compared it to DDNN. The optimal measure for two variables of normal distribution in statistics is the ANOVA test given in Table 4. Similarly, we can see values like mean, standard deviation (SD), *p*-value, and t-value in Table 4. The level of significance, meanwhile, is set to $p < 0.05$. At this stage, we need to define the hypothesis in the following manner:

H0: DDNN and other baselines have no difference.

H1: A significant difference exists between DDNN and other baselines.

In all cases, we could see that *p*-values are lower than the significance level <0.05, which indicates that there is a significant difference between the values of DDNN and other baselines. Therefore, we are right in denying the null hypothesis and supporting the alternative hypothesis. Similarly, in terms of energy consumption, migration time, and optimization time, we can say that a substantial difference occurs in the resulting values.

Table 4: Statistical comparison of DDNN with baselines

ANOVA test (ISDNSDN)						
Source of variation	SS	Df	MS	F	*p*-value	F crit
Between groups	5653.84226	6	942.307	3.033953	0.011384	2.246408
Within groups	19566.997	63	310.5873			
Total	25220.8392	69				
ANOVA test (ADABOOST)						
Source of variation	SS	Df	MS	F	*p*-value	F crit
Between groups	21710.07	6	3618.344	2.202807	0.037471	0.246408
Within groups	1124004	63	17841.34			
Total	1145714	69				
ANOVA test (SAFETY)						
Source of variation	SS	Df	MS	F	*p*-value	F crit
Between groups	1.13E + 08	6	18830338	4.658749	0.000559	2.246408
Within groups	2.55E + 08	63	4041930			
Total	3.68E + 08	69				

5.7 Threats to the Validity of the Proposed System

The proposed system is one of the best solutions to detect and mitigate about the DDoS attacks from SDN internal network structure. The proposed system according to the results discussed in precision, recall and F1-score computed through Machine learning DDNN approach the proposed system is one of the best defenses against DDoS attacks from SDN network. Moreover, the accuracy of the model 99.76% to detection and mitigation of the attacks which is one of the most effective comparisons for the proposed methodology.

6 Conclusion

SDN decouples the control plane from the data plane and centralizes the control plane. It simplifies the management of a network and provides superior performance in large-scale networks. The centralization of the control plane makes the SDN prone to DDoS attacks. DDoS attacks are carried out by different compromised devices. It is the most extensive kind of attack and faced by different financial and public sector organizations. We proposed a Deep Neural Network (DNN) model to detect DDoS attacks in SDN. The Proposed DNN model scans the data to find the correlated features and delivers high-quality results. The model enhances the security of SDN and has better accuracy as compared to previously proposed models. We also use the latest state-of-the-art dataset that overcome all the shortcomings and limitations of the existing datasets. We choose the best twenty (20) to train our model on different kinds of DDoS attacks. Our model has better accuracy as compared to previously proposed models. In the future, we can apply this model to detect IDS attacks and other several types of attacks. We can also improve our model for the classification of different attacks available in the dataset.

Acknowledgement: Thanks to our families & colleagues who supported us morally.

Funding Statement: This work was funded by the University Malaysia Sabah.

Author Contributions: Conceptualization and original draft preparations are performed by Gul Nawaz and Muhammad Junaid, methodology developed by Adnan Akhunzada and Abdullah Ghani, validation of the study is conducted by Shamyla Nawazish, visualization is performed by Adeel Ahmed, formal analysis is done by Asim Yaqub and resources including data curation, reviewing and editing are performed by Huma Ajab. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials: Data and materials are within the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Y. Jarraya, T. Madi and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[2] A. Reem, A. Ali, F. Jamil, M. Nawaz, F. Mehmood *et al.,* "Intelligent transmission control for efficient operations in SDN," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 2807–2825, 2022.

[3] K. Suzuki, K. Sonoda, N. Tomizawa, Y. Yakuwa, T. Uchida *et al.,* "A survey on openflow technologies," *IEICE Transactions on Communications*, vol. E97-B, no. 2, pp. 375–386, 2014.

[4] A. Shabir, F. Jamil, A. Ali, E. Khan, M. Ibrahim *et al.,* "Effectively handling network congestion and load balancing in software defined networking," *Computers, Materials & Continua*, vol. 70, no. 1, pp. 1363–1379, 2022.

[5] S. S. Hayward, G. Callaghan and S. Sezer, "SDN security: A survey," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, pp. 1–7, 2013.

[6] K. Kalkan, G. Gur and F. Alagoz, "Defense mechanisms against DDoS attacks in SDN environment," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 175–179, 2017.

[7] J. S. Perrone and J. Sims, "Securing cloud, sdn and large data network environments from emerging DDoS attacks," in *2017 7th Int. Conf. on Cloud Computing, Data Science & Engineering-Confluence*, Noida, India, pp. 466–469, 2017.

[8] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang *et al.,* "Detection and defense of DDoS attack–based on deep learning in openflow-based SDN," *International Journal of Communication Systems*, vol. 31, no. 5, Article ID e3497, 2018.

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

[10] A. Javaid, Q. Niyaz, W. Sun and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. of 9th EAI Int. Conf. on BioInspired Information Communication Technologies*, New York City, USA, pp. 21–26, 2016.

[11] N. Meti, D. Narayan and V. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *2017 Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI)*, Mangalore, India, pp. 1366–1371, 2017.

[12] O. Rahman, M. G. Quraishi and C. H. Lung, "DDoS attacks detection and mitigation in SDN using machine learning," in *2019 IEEE World Congress on Services (SERVICES)*, Milan, Italy, pp. 184–189, 2019.

[13] H. Polat, O. Polat and A. Cetin, "Detecting DDoS attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, vol. 12, no. 3, 2020.

[14] M. Myint, S. Kamolphiwong, T. Kamolphiwong and S. Vasupongayya, "Advanced support vector machine (ASVM) based detection for distributed denial of service (DDoS) attack on software defined networking," *Journal of Computer Networks and Communications*, vol. 2019, 2019.

[15] R. Kokila, S. T. Selvi and K. Govindarajan, "DDoS detection and analysis in SDN-based environment using support vector machine classifier," in *2014 Sixth Int. Conf. on Advanced Computing (ICoAC)*, Chennai, India, pp. 205–210, 2014.

[16] M. Junaid, A. Sohail, A. Ahmed, A. Baz, I. A. Khan *et al.,* "A hybrid model for load balancing in cloud using file type formatting," *IEEE Access*, vol. 8, pp. 118135–118155, 2020.

[17] M. Junaid, A. Sohail, R. Naveed, A. Ahmed, I. Khan *et al.,* "Modeling an optimized approach for load balancing in cloud," *IEEE Access*, vol. 8, pp. 173208–173226, 2020.

[18] M. Junaid, A. Sohail, F. A. Turjman and R. Ali, "Agile support vector machine for energy-efficient resource allocation in IoT-oriented cloud using PSO," *ACM Transactions on Internet Technology*, vol. 22, no. 1, pp. 1–35, 2022.

[19] M. Junaid, A. Sohail, M. Alkinani, A. Ahmed, M. Ahmed *et al.,* "Enhancing cloud performance using file format classifications," *Computers, Materials & Continua*, vol. 70, no. 2, pp. 3985–4007, 2021.

[20] M. Junaid, A. Shaikh, M. U. Hassan, A. Alghamdi, K. Rajab *et al.,* "Smart agriculture cloud using AI based techniques," *Energies*, vol. 14, 2021.

[21] M. Junaid, M. Hussain, A. Masood, F. Kausar, A. Noreen *et al.,* "Evaluation of framework for the comparative analysis of symmetric block ciphers," in *2nd Int. Conf. on Computer Science and Its Applications*, Jegu, South Korea, pp. 1–4, 2009.

[22] M. Junaid, F. Khan, A. Jehangiri, Y. Saeed, M. Ahmed *et al.,* "An indigenous solution for SYN flooding," *Revista Geintec Gestao Inovacao E Technologias*", vol. 11, no. 4, pp. 2998–3022, 2021.

[23] B. Al-Duwairi, E. Al-Quraan and Y. A. Qader, "ISDSDN: Mitigating SYN flood attacks in software defined networks," *Journal of Network and Systems Management*, vol. 28, no. 9, pp. 1366–1390, 2020.

[24] P. Kumar, M. Tripathi, A. Nehra, M. Conti and C. Lal, "SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[25] R. F. Fouladi, O. Ermiş and E. Anarim, "A DDoS attack detection and defense scheme using time-series analysis for SDN," *Journal of Information Security and Applications*, vol. 54, no. 3, 2020.

[26] S. Sen, K. D. Gupta and M. M. Ahsan, "Leveraging machine learning approach to setup software-defined network (SDN) controller rules during DDoS attack," in *Proc. of Int. Joint Conf. on Computational Intelligence*, Budapest, Hungary, pp. 49–60, 2020.

[27] J. Haveri, H. Rutvij, S. V. Ramani, G. Srivastava, T. R. Gadekallu *et al.,* "Fault-resilience for bandwidth management in industrial software-defined networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3129–3139, 2021.

[28] J. A. Rehman, S. Rehman, M. Khan, M. Alazab and T. Reddy, "CANintelliIDS: Detecting in-vehicle intrusion attacks on a controller area network using CNN and attention-based GRU," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1456–1466, 2021.

[29] A. Shaashwat, A. Chowdhuri, S. Sarkar, R. Selvanambi and T. Gadekallu, "Temporal weighted veraging for asynchronous federated intrusion detection systems," *Computational Intelligence and Neuroscience*, vol. 21, Article Id. 5844728, 2021.

[30] M. Kim, "Supervised learning-based DDoS attacks detection: Tuning hyperparameters," *ETRI Journal*, vol. 41, no. 5, pp. 560–573, 2019.

[31] S. Haider, A. Akhunzada, G. Ahmed and M. Raza, “Deep learning based ensemble convolutional neural network solution for distributed denial of service detection in SDNs,” in *2019 UK/China Emerging Technologies (UCET)*, Glasgow, UK, pp. 1–4, 2019.

[32] N. Omnes, M. Bouillon, G. Fromentoux and O. L. Grand, “A programmable and virtualized network and IT infrastructure for the internet of things: How can NFV & SDN help for facing the upcoming challenges,” in *2015 18th Int. Conf. on Intelligence in Next Generation Networks*, Paris, France, pp. 64–69, 2015.

[33] B. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka and T. Turletti, “A survey of software-defined networking: Past, present, and future of programmable networks,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[34] O. N. Fundation, “Software-defined networking: The new norm for networks,” *ONF White Paper*, vol. 3, no. 11, 2012.

[35] I. Bueno, J. I. Aznar, E. Escalona, J. Ferrer and J. A. Espin, “An opennaas based sdn framework for dynamic qos control,” in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, Italy, pp. 1–7, 2013.

[36] R. D. Oliveira, C. M. Schweitzer, A. A. Shinoda and L. R. Prete, “Using mininet for emulation and prototyping software-defined networks,” in *2014 IEEE Colombian Conf. on Communications and Computing (COLCOM)*, Bogota, Colombia, pp. 1–6, 2014.

[37] A. K. Maddala, “Modeling of openflow based software defined networks using mininet,” M.S. Thesis, Dept. Electron. Eng., Iowa State University, Ames, Iowa, USA, 2019.

[38] S. Asadollahi, B. Goswami and M. Sameer, “Ryu controller’s scalability experiment on software defined networks,” in *2018 IEEE Int. Conf. on Current Trends in Advanced Computing (ICCTAC)*, Bangalore, India, pp. 1–5, 2018.

[39] N. J. Nilsson, *The Quest for Artificial Intelligence*. Cambridge, England: Cambridge University Press, pp. 1–707, 2009. [Online]. Available: https://ai.stanford.edu/~nilsson/QAI/qai.pdf

[40] J. M. Nazzal, I. M. El-Emary and S. A. Najim, “Multilayer perceptron neural network (MLPs) for analyzing the properties of jordan oil shale,” *World Applied Sciences Journal*, vol. 5, no. 5, pp. 546–552, 2008.

[41] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu *et al.,* “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, no. 1, pp. 11–26, 2017.

[42] A. H. Lashkari, G. D. Gil, M. S. I. Mamun and A. A. Ghorbani, “Characterization of tor traffic using time based features,” in *3rd Int. Conf. on Information Systems Security and Privacy (ICISSP)*, Porto, Portugal, pp. 253–262, 2017.

[43] S. Karamizadeh, S. M. Abdullah, A. A. Manaf, M. Zamani and A. Hooman, “An overview of principal component analysis,” *Journal of Signal and Information Processing*, vol. 4, no. 3B, 2013.

[44] T. B. Arnold, “kerasR: R interface to the keras deep learning library,” *Journal of Open Source Software*, vol. 2, no. 14, 2017.

[45] T. Kurth, M. Smorkalov, P. Mendygral, S. Sridharan and A. Mathuriya, “TensorFlow at scale: Performance and productivity analysis of distributed training with horovod, MLSL, and cray PE ML,” *Concurrency and Computation: Practice and Experience*, vol. 31, no. 16, 2019.