I. J. Computer Network and Information Security, **2025, 4, 1-18**
Published Online on August 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2025.04.01

Software-defined Networking Controller for Detection of DDoS Attacks Based on Deep Neural Networks

Jeyavim Sherin R. C.
Vellore Institute of Technology/School of Computer Science and Engineering, Chennai, 600127, India
E-mail: jeyavimsherin.rc2021@vitstudent.ac.in
ORCID iD: https://orcid.org/0000-0003-4453-7806

Parkavi K.*
Vellore Institute of Technology/School of Computer Science and Engineering, Chennai, 600127, India
E-mail: parkavi.k@vit.ac.in
ORCID iD: https://orcid.org/0000-0002-3617-3590
*Corresponding author

Received: 10 January 2025; Revised: 11 March 2025; Accepted: 02 May 2025; Published: 08 August 2025

Abstract: Advancements in technology contribute to an increased vulnerability to cyberattacks, with Distributed Denial of Service (DDoS) attacks being a prominent threat. Attackers overwhelm network servers with excessive data, hindering legitimate users from accessing them. Software Defined Networking (SDN) is particularly susceptible due to its centralized architecture, making it a prime target for DDoS attacks aimed at the control planes. As cloud computing has grown rapidly, software-defined networks have been developed to provide dynamic management and enhanced performance. Several security concerns are growing, especially as DDoS attacks and malicious actors become more interested in SDN controllers. Many researchers have proposed detecting DDoS attacks. Due to their unqualified features and non-realistic data sets, these approaches have high false positive rates and low accuracy. As a result, SDN controllers can be protected against DDoS attacks using deep learning algorithms (DL). Furthermore, the suggested method comprises three phases: The process involves pre-processing the data, selecting significant features for DDoS detection based on correlation, and utilizing Deep Neural Networks (DNNs) for the detection. In order to evaluate the efficiency of the method proposed, we employ a benchmarking dataset to evaluate the false positive rate as well as detectability, with the traditional assessment indicators. In this paper, we propose a deep learning method for detection of DDoS attacks called DNNADSC, which is the first anomaly detection method based on deep neural network for DDoS attacks. The method proposed efficaciously recognizes DDoS attacks, with the detection rate of 99.39%, with a precision of 97.41% with a false-positive rate (FPR) that is 0.0665 with the F1 measure of 99.32%.

IndexTerms: DDoS Attack, Deep Neural Network, Machine Learning Algorithm, Random Search, Logistic Regression, Naive Bayes, Software Defined Networking.

1. Introduction

DDoS attacks stand out as a significant security concern within SDN networks. In a network attack, one objective is to interfere with the data plane, disrupt the communication channel, or destabilize the entire network. Presently, a vast array of business sectors relies on Software Defined Networking (SDN) to deliver tailored services, catering to diverse needs and preferences [1]. Traditional networks fail to meet specific demands, including the need for extensive bandwidth, rapid speeds, seamless accessibility, virtualization, and dynamic management. Network managers may now create and maintain network services in a proactive manner by using software-based solutions rather than depending just on physical infrastructure [2]. It improves network connectivity for client service, document sharing, and corporate communication. A customized network infrastructure and service can be created using both hardware and software from different suppliers. As SDN networks can be deployed on cloud computing and virtualization platforms, they are flexible and reliable [3]. Through the utilization of controllers to manage packet forwarding and employing the OpenFlow protocol, it establishes an intelligent centralized system. The user favours SDN over traditional networks because of the ease of network control facilitated by programmable tools [4]. OpenFlow switches are specifically

designed for this purpose in SDN to allow incoming packets to pass across an open interface.

SDN architectures consist of OpenFlow switches, routers, and data planes. Fig. 1 depicts the control layer as control planes and the application layer as application planes. The control layer, represented by the SDN controller, oversees traffic monitoring and deployment procedures, while application layers handle load balancing and routing operations [5]. Two crucial networking components that function at the infrastructure layer are switches and routers. The control plane assists in the transfer of data packets in the data plane and follows an established set of guidelines. Utilities use the Northbound and Southbound interfaces to program hardware components within the data plane [6]. The SDN controller makes use of the northbound interface to communicate with the services on the application layer. It uses the southbound interface for communication with devices within the plane of data.

A DDoS attack against the control plane has the potential to bring down the entire network due to its centralized architecture. DDoS attacks can cause destruction of packets as well as the destruction of resources in networks regardless of whether they attack the data plane, or the communications channel [7]. Botnets, which consist of networks of coordinated bots, are used to execute DDoS attacks. Botnets gain access to legitimate systems, and convert them into zombies or bots by using malware that is malicious. They are able to participate in the communications between open flow switches as well as the SDN controller once they have been targeted. Such attacks use SDN controllers to break network resources and introduce a single point of failure [8]. When a large number of fake packets from zombies or bots flood the SDN Controller, it overloads its resources and prevents the transmission of valid packets. As a result, as shown in Fig. 2. SDN controller is unable to act in the role of a hub. It then shuts itself down, which leads to a decrease in overall performance [9].

Current detection methods for DDoS attacks in SDN environments are ineffective due to the various intricacies inherent in SDN architecture [10]. Given the increased risk of DDoS attacks on SDN systems, the SDN controller needs to constantly oversee the network traffic that passes through the OpenFlow switches.

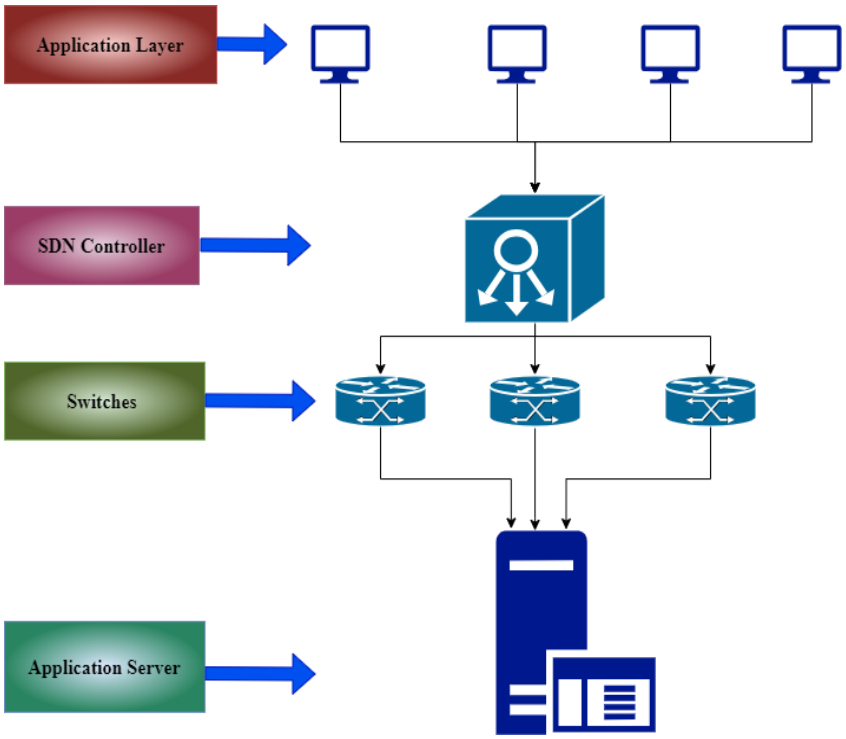

Fig.1. SDN architecture

In order to overcome the limitations, it is crucial to include pertinent features as well as use accurate data sources when designing DDoS attacks detection techniques to be used by SDN controllers.

The main contributions presented in this paper are:

- We introduce DNNADSC, a novel DDoS detection system using deep neural networks.
- Introducing a feature selection process based on correlation. The objective of this method is to pinpoint the most informative elements within network traffic. In addition, a strategy with feature intersections is suggested to increase the process of selection as well as improve identification of DDoS attacks that target SDN Network Controllers.
- Constructing a resilient DNN (Deep Neural Network) model trained with the chosen attributes. This model captures the intricate characteristics associated with TCP, ICMP, and UDP attacks, making it easier to identify them precisely.

- Analyzing the suggested model and presenting its positive results using a variety of assessment criteria, including F1 score, false positive rate, precision, and detection accuracy.

This study paper's ensuing sections are arranged as follows: Background details are given in Section 2. Sections 3 through 5 cover relevant works, the DNNADSC approach, and the preliminary stages of the suggested approach, in that order. The findings of the experiment are shown in Section 6. In-depth analysis and interpretation of the research findings are provided in Section 7. The conclusion of Section 8 summarizes the major contributions, and also suggests the direction for further research making the research paper.

2. Background

2.1. Software-defined Networking (SDN) Controllers and the OpenFlow

The centrally-planned SDN controller performs the tasks of surveillance, statistical analysis and inspection of packets inbound networks traffic. This is in addition to changing switch table information. Because of its central location in the network, the controller controls every traffic on the network [11]. It takes decisions based upon the details it's collected. SDN software applications, such as firewalls, load balancers and firewalls make use of northbound APIs. Southbound APIs such as OpenFlow allow the interaction of switches [12]. SDN controllers are available in a variety of varieties, ranging including commercial ones such as Big Switch Networks' BNC to VMware's vCloud, vSphere Trema by NEC Nicira's NVP, Floodlight. Symmetric, asynchronous, and controller-to-switch message types are used in the protocol used by SDN-enabled networks to communicate between their forwarding and control planes. Fig. 2 illustrates the use of OpenFlow as one of the communication protocols in SDN networks.

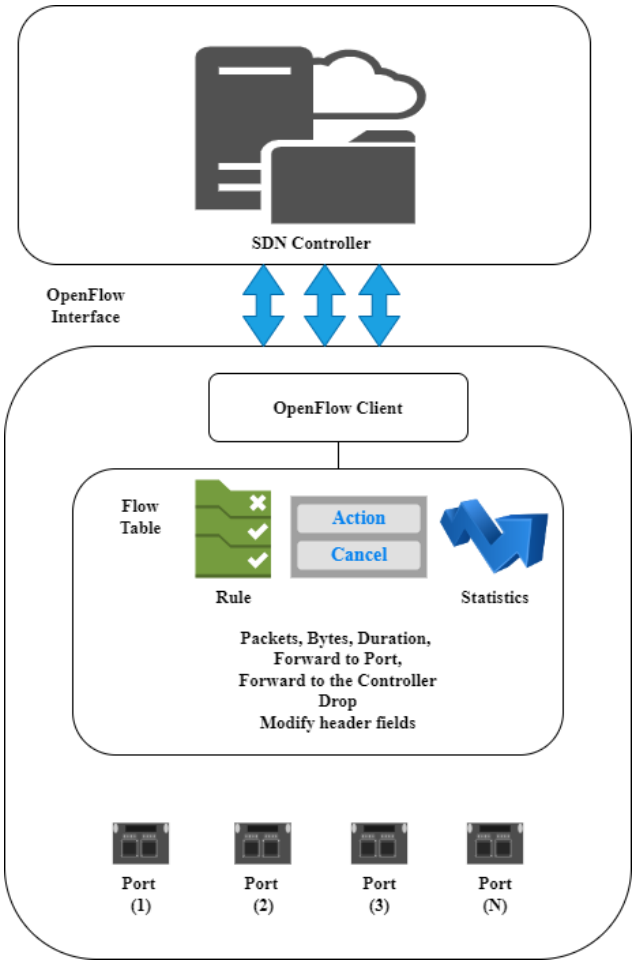

Fig.2. OpenFlow protocol communication

OpenFlow switches as well as other forwarding devices provide abstraction layers that are based on flow tables. OpenFlow allows SDN controllers and switches at the infrastructure layer to safely communicate with one another. The entries inside the flow tables define how packets are routed, and handled [13]. The sequence of operations, count, and match fields make up the flow entry. A match field and a count field are the two fields used for flow input. Counters keep track of flow statistics, and match fields use packet headers to determine the packets that are coming in. The OpenFlow

switch analyzes the header fields in a packet and looks for its similar entries in the table of flow. If flow entries are in sync when flow entries match, switches follow the appropriate instruction. If there is no match then the switch is set to discard the packet on the basis of instructions by the controller [14].

2.2. Methodologies for DL

A growing number of practical applications of artificial intelligence (AI) have led to a growing interest in the field, which is subject to ongoing research. A smart infrastructure can be created, individuals with disabilities can be empowered, scientific research can be supported, and tasks can be automated, images analyzed, conversations understood, medical diagnoses made, and smart infrastructure can be built. Most AI solutions are based on machine learning (ML) [15]. Due to the increasing amount of data, today's artificial intelligence models are developed to anticipate and uncover insights through the identification of patterns and connections within dataset. AI Deep Learning and AI share an intimate relationship. While machine learning can be integrated into several AI applications, it's not applied to every AI system [16]. DL is a kind of representational learning. Every segment of the Fig. 3 diagram demonstrates how AI is used.

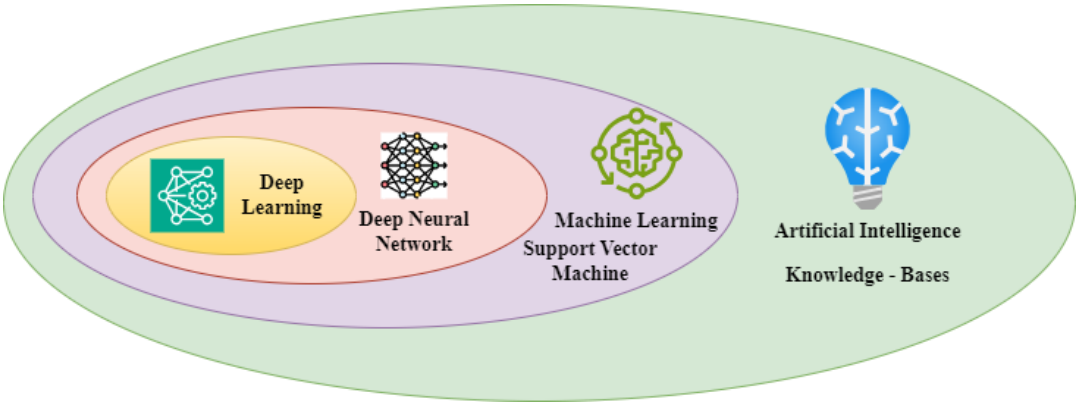

Fig.3. Relationship between AI, machine learning, and deep learning

As the model becomes more complex, DL allows multi-layer abstraction of data representation, thereby improving the capabilities of classical ML. Feature learning, which automatically takes raw data features and integrates them into higher-level features, is a major benefit of deep learning [17]. In addition to tackling complicated problems, DL also enables for efficient parallelization. Deep learning architectures can include elements such as memory cells, pooling layers, gates, convolutions fully connected layers, activation functions, as well as encoding/decoding methods, dependent on the particular network design.

New kinds of neural networks are being explored alongside the ones that already exist, thanks to research. The architecture, data flow, neuron count, density, layering, depth, activation functions, and filtering methods are all factors used to categorize a neural network. Deep Neural Networks (DNNs), Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks are among the top often used types of neural networks [18]. Every kind of neural network has advantages and disadvantages that vary depending on usage situations, use cases, and input/output data needs. We compare the inputs or outputs, architectures, real-world applications, and appropriate use scenarios for CNNs, DNNs, and LSTMs and LSTMs in Table 1.

Each of these three deep learning techniques has been widely applied in diverse fields and is appropriate for a range of tasks due to its individual advantages and disadvantages. The choice of deep-learning models is determined by the domain in which they are applied and characteristics of the dataset.

3. Related Works

This article focuses on cutting-edge ML and DL-based methods that are used to spot DDoS attacks against SDN controllers. Two subsections constitute this section: approaches based on ML and approaches based on DL.

3.1. Approaches Based on Machine Learning

Celesova et al. [19] created the Deep Neural Network (DNN) designed to protect the data planes as well as taking care of DDoS attacks within SDN networks. The study used the UNSW-NB15 data set, which wasn't intended to be originally developed to be used in SDN networks. It was used to train, test, as well as evaluating the proposed method. The calculated metrics showed poor performance as a consequence.

Sudar et al. [9] employed Decision Trees (DTs) and Support Vector Machines (SVMs) in SDN networks to identify DDoS attacks. The KDD CUP dataset was used to evaluate their proposed approach. As a result, Decision Trees only achieved a 78 out of 100, which is inadequate performance.

Deepa et al. [3] combined self-organizing Map (SOM) as well as Support Vector Machine (SVM) to determine DDoS attacks in SDN networks. The hybrid model showed the false alarm rate at 0.082%, an accuracy of 96.77% and a detection percentage of 90.45%. Nevertheless, hybrid techniques for target detection did not yield significant success or precision. Furthermore, the authors did not include a lot of details regarding DDoS attacks or the dataset.

Machine learning algorithms are utilized to advance and refine methodologies based on machine learning. Machine learning algorithms are employed in the development and enhancement of machine learning-based methodologies. The analysis of network traffic characteristics allows IDSs to detect the threat of DDoS attacks. According to Khashab et al. [20] SDN techniques can detect and reduce DDoS attacks. Six machine learning methods were employed: Random Forest (RF), Logistic Regression, Naive Bayes, K-Nearest Neighbors (K-NN), Support Vector Machine (SVM), and Decision Trees. By their methodology, RF is the best classifier and outperforms the other methods. The proposed approach increases overhead, especially during DDoS attacks, when implemented on the SDN controller. ML-based approaches and their drawbacks are outlined in Table 2.

Table 1. An analysis of CNN, DNN, and LSTM

	LSTM	DNN	CNN
Architecture	LSTMs and RNNs are gate groups that can store data for a long time in memory.	A DNN consists of multiple layers of neurons leading to an output layer.	An algorithm that combines filters and pooling in order to produce feed-forward neural networks.
Inputs or Outputs	Input and output sizes may differ.	Input and output sizes of a DNN depend entirely on its task and architecture.	There is no difference in the size of the inputs and outputs.
Examples of practical applications	Image processing, speech recognition, and handwriting recognition.	Image classification, Object detection, Sentiment Analysis and Text generation.	Analysing faces, analysing images, recognizing images, and classifying images.
Scenarios of suitable usage	Text or video that is sequential or temporal.	Large Scale Data, Complex Pattern, Feature Learning, Transfer Learning, Unstructured data and Multi-modal Data Fusion.	Data that is spatial, such as Image Classification, Semantic Segmentation and Document Analysis.

3.2. Approaches based on Deep Learning

IDSs have relied heavily on DL algorithms in recent years to detect DDoS attacks in SDN networks. In terms of efficacy and efficiency, DL algorithms perform better than conventional ML techniques. Moreover, deep learning algorithms are able to automatically extract deep structures from unprocessed data, emulating and gaining features from the human brain. Diverse methods have been suggested for DL-based techniques. A variety of DL-based strategies have been put forth. Lee et al. [21] utilized three deep learning algorithms to build an Intrusion Detection System (IDS): The Multilayer Perceptron (MLP), the Stacked Autoencoder (SAE), as well as the Convolutional Neural Network (CNN). Using MLP, SSH attacks and DDoS attacks were identified with 98.3% and 99% accuracy, respectively. Other deep learning systems struggled to identify these kinds of attacks because they lacked sufficient knowledge about the dataset and its properties.

Table 2. The associated works for ML and DL methods

Author & Ref	Method		SDN Dataset	SDN Domain	Accuracy		Limitations
	ML	DL			Low	High	
Celesova et al.[19]	✓	×	×	✓	✓	×	Low performance is achieved and UNSW-NB15 data is used.
Sudar et al. [9]	✓	×	×	✓	✓	×	Low performance is achieved and KDDCUP data is used.
Deepa et al. [3]	✓	×	✓	✓	✓	×	Low performance is achieved. The dataset and different kinds of DDoS attacks are not sufficiently described.
Khashab et al.[20]	✓	×	✓	✓	×	✓	When attacks occur, the approach increases workload at the Controller.
Lee et al. [21]	×	✓	-	✓	×	✓	The dataset and the features that were utilized to assess the traffic are not well described.
Alanazi et al. [22]	×	✓	×	✓	×	✓	An inadequate dataset for SDN environments is used in this study's evaluation of the technique.
Boukria et al. [23]	×	✓	×	✓	×	✓	A non-SDN dataset is used to evaluate it. It solely safeguards avenues of communication.
Wan et al. [2]	×	✓	×	✓	×	✓	The method is assessed using a dataset that is not SDN.

Alanazi et al. [22] suggested an ensemble method that blends Gated Recurrent Units (GRU), Convolutional Neural Networks (CNN) as well as Long-Short-Term Memory (LSTM) to identify DDoS attacks within SDN networks. A high degree of detection accuracy was obtained by using the CICIDS2020 dataset to choose only four features for examination.

Boukria et al. [23] utilized deep learning algorithms to detect attempted attacks on the communication channel (southbound API) connecting the infrastructure plane to the SDN controller. The authors reported an accuracy of 99.6% in detection. Nevertheless, these findings were evaluated on non-SDN datasets and were based on non-SDN datasets that do not represent SDN networks. Furthermore, it alone safeguards avenues of communication.

Wan et al. [2] utilized a bidirectional long-short-term Memory (LSTM) network to identify dangers in SDN networks. With low false-positive rates and excellent accuracy, this method effectively identified false-positives. As Table 2 illustrates, DL-based approaches have a number of drawbacks.

As Table 2 illustrates, there are a number of methods that have produced low detection accuracy in the literature. In addition, the majority modern methods are based on SDN controllers. This increases costs during DDoS attacks [24,25]. Through the use of a DNN-based technique to recognize DDoS attacks inside the environment of SDN that has the lowest false positive rate as well as excellent detection accuracy the suggested method gets over the drawbacks.

4. The Preliminary Phase

Security concerns arising from SDN layers are discussed in this preliminary section. SDN architectures consist of several layers, such as the applications layer, control layer, and infrastructure layer. The vulnerabilities and threats that exist at each of these layers differ.

4.1. Problems Associated with SDN Security

Despite all of Software-Defined Networking's (SDN) many features and capabilities, security remains a major problem. The most vulnerable parts of the system are the SDN controllers, since they are the main parts in charge of controlling data flow. A breach of an SDN controller can have serious consequences for the network as a whole. The programmability of SDN controllers makes them vulnerable to potential attacks, which compromises network integrity, authentication, and security [26]. Unconfigured controllers could result in significant consequences. Monitoring, analysis, and reaction methods inside SDN can be implemented to improve network security. Cyber-attacks targeting SDN can result in greater damage than attacks targeting traditional networks [27].

Table 3 offers a comprehensive overview of all the different types of attacks that could occur within SDN deployments, based on risk levels in different SDN layers [28]. A configuration error must be prevented at each layer of SDN. Security threats and attacks are exposed to the network if these requirements aren't met. An attack on all system levels can be launched if the switch-to-controller communication link is compromised. Users may be unable to access the controller if attacks are made on enforcement security [29].

DDoS assaults, ARP spoofing attacks, packet sniffing, API exploits, brute force attacks, and password guessing are examples of malicious attacks on SDN networks [30]. This study includes the development of a technique for aggregating feature selection that aids in DDoS detection. Below, we discuss an approach based on DL DNNs.

Table 3. SDN security threats at different layers

Issues	Application Layer	Controller Layer	Data Layer	SDN Layer Threats
Leakage of Data	✗	✓	✗	Management of credentials.
DoS Attack	✗	✓	✓	Controller for flooding.
Unauthorized access	✓	✓	✗	Applications that are not authorized.
Information about modifications	✗	✓	✓	Tampering with data flow.
Issues with configuration	✓	✓	✓	Policy enforcement.

4.2. DNN-Based Approach

In addition to the growing use of SDN technology, securing these networks has become a top priority. DDoS assaults are a significant danger to SDNs because they impair service availability and interfere with regular network operations. Such attacks are detectable by IDS. Systems that detect intrusions based on signatures have found it difficult to keep pace with the constantly evolving strategies employed by DDoS attackers. This challenge has led DL to become a prominent player in the IDS industry.

Due to the following reasons, DNN is our preferred DL algorithm in this study. A deep neural network solves real world problems such as classification by receiving inputs, performing increasingly complex calculations, and producing an output. In comparison to conventional techniques, Deep Neural Networks (DNNs) provide a number of benefits for identifying and countering Distributed Denial of Service (DDoS) assaults. For identifying and mitigating DDoS attacks effectively, DNNs excel at detecting patterns and anomalies within large and complex datasets. By using past traffic data to understand a network's typical behavior, DNNs are able to identify DDoS attacks in real time. In addition, DNNs are capable of adapting to changing attack strategies and patterns, making them resistant to sophisticated DDoS attacks

that attempt to evade traditional detection methods. A further advantage of DNN architectures is their scalability, which enables them to handle large volumes of network traffic typically associated with DDoS attacks. Using DNNs to detect and mitigate DDoS attacks enhances network protection against disruptive and damaging cyber-attacks.

5. DNNADSC Approach

This chapter outlines a procedure to identify DDoS attacks by identifying distinct aspects that differentiate DDoS from normal activity together a Deep Neural network-based Method for detecting DDoS attacks using the Software-Defined Networking Control (DNNADSC). The three stages of DNNADSC consist of data processing as well as categorical feature choice as well as a DNN-based system to aid in DDoS security.

Another step for identifying DDoS attacks that target the SDN controller involves categorical feature choice, important to DDoS attacks detection. Detecting DDoS attacks more effectively can be achieved by selecting the most relevant features.

The chosen features are used to train a DNN-based detection system. Another step for identifying DDoS attacks that target the SDN controller involves categorical feature choice, important to DDoS attacks detection. The model can distinguish between malicious and legitimate incoming traffic using the features that have been chosen.

Fig. 4 illustrates the proposed approach using these three stages. Detailed descriptions of each stage will follow, emphasizing their importance in detecting DDoS attacks accurately and robustly.

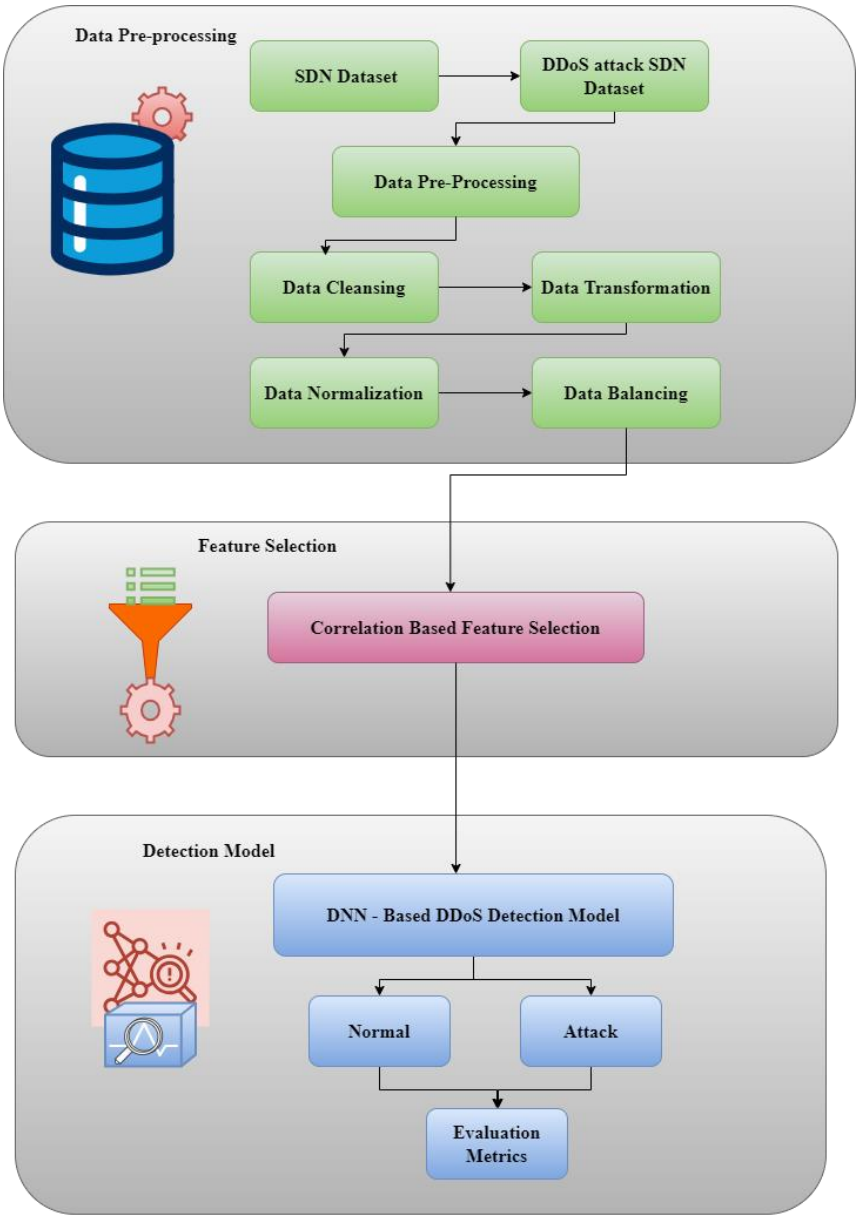

Fig.4. Proposed architecture

5.1. Dataset Preprocessing

DDOS attack SDN Dataset has been used as a benchmark dataset for evaluating the proposed approach. A testbed composed of ten Mininet topologies connected to a single Ryu controller was used to generate this dataset. The dataset was created using simulations of malicious traffic (TCP Syn, ICMP, and UDP Flood assaults) as well as benign traffic (TCP, UDP, and ICMP). Table 4 presents a detailed description of the benchmark dataset, emphasizing the features that are most similar to SDN networks in the actual world. Before feeding the dataset into a deep learning model, thorough preprocessing is necessary to ensure its quality and effectiveness.

- **Data Cleaning:** The process of eliminating duplicates, corrupted, incorrectly formatted, or incomplete data from a dataset is known as data cleaning.
- **Data Transformation:** A data transformation procedure involves changing one file format or structure into another. For DL applications, this could entail changing string data into numerical data.
- **Data Normalization:** Normalization is the process of minimizing or removing duplicate data. Normalizing input features or variables with the Min-Max scaler is a common method. This transforms all features into 0-1 ranges. The normalization method helps to reduce bias due to measurement at different scales which do not contribute equally to the process of fitting models. The equation (1) illustrates how we standardize feature vectors based on feature-wise normalization.

$$x_{Scale} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

- **Data Balancing:** In data-balancing, the normal classes and the attack classes are equally distributed throughout the dataset. There are 41.25% assault records and 58.75% normal records, as shown in Fig. 5. To enhance the quantity of data points for normal classes, the SMOTE oversampling approach was applied. This balancing process greatly enhances data categorization accuracy.

5.2. Correlation Based Feature Selection

Finding and choosing characteristics in a dataset that are connected to a target variable or other features is done using the correlation-based feature selection technique (CFS). In order to avoid redundancy or multicollinearity among the features, we must retain only those features with a high correlation to the target variable the equation 2 denoted the correlation based feature selection. A DDoS attack is detected when a linear relationship is found between individual features and the occurrence of DDoS attacks. Furthermore, we identify potential redundant features by examining correlations between pairs of features.

Table 4. SDN DDoS dataset description

Description	Details
Number of records in total	104345
Attack Records	41.25%
Normal Records	58.75%
Category	Attack and Normal
Abnormal Classes	ICMP, TCP Syn, UDP Flooding Attacks
Normal Classes	TCP, UDP, ICMP
Number of Features in the Dataset	22
Features	'dt', 'switch', 'src', 'dst', 'pktcount', 'bytecount', 'dur', 'dur_nsec', 'tot_dur', 'flows', 'packetins', 'pktperflow', 'yteperflow', 'pktrate', 'Pairflow', 'Protocol', 'port_no', 'tx_bytes', 'rx_bytes', 'tx_kbps', 'rx_kbps', 'tot_kbps'.

$$All\ features = \{'dt','switch','src','dst','pktcount','bytecount','dur', 'dur\_nsec','tot\_dur','flows','packetins','pktperflow', 'byteperflow','pktrate','Pairflow','Protocol','port\_no', 'tx\_bytes','rx\_bytes','tx\_kbps','rx\_kbps','tot\_kbps'\}$$

$$Merit_S = \frac{k \cdot \overline{r_{cf}}}{\sqrt{k + k(k-1) \cdot \overline{r_{ff}}}} \tag{2}$$

Where:

- $Merit_S$ is the heuristic "merit" of a feature subset *S*.
- k is the number of features in the subset *S*.

- $\overline{r_{cf}}$ is the average feature-class correlation (the average correlation between the features and the class).
- $\overline{r_{ff}}$ is the average feature-feature correlation (the average inter-correlation between the features).

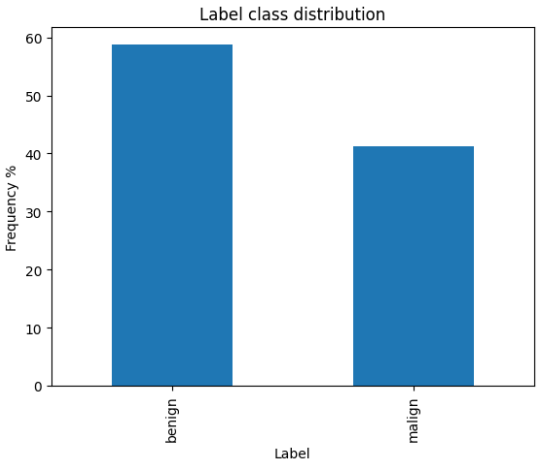

Fig.5. SDN DDoS dataset distribution

Algorithm 1: Feature Selection Algorithm

Input: dataset, features, target variable
Output: selected features

```
Load dataset;
Define features = {'dt','switch','src','dst','pktcount','bytecount','dur',
                   'dur_nsec','tot_dur','flows','packetins','pktperflow',
                   'byteperflow','pktrate','Pairflow','Protocol','port_no',
                   'tx_bytes','rx_bytes','tx_kbps','rx_kbps','tot_kbps'}
Define target_variable = 'DDoS_attack';
Initialize selected_features = [ ];
Set correlation_threshold = 0.5;
for each feature in features do
    correlation = compute_correlation (dataset [feature], dataset [target_variable]);
    if abs(correlation) > correlation_threshold then
        Add feature to selected_features;
    end
end
Set redundancy_threshold = 0.7;
correlation_matrix = compute_correlation_matrix(dataset [selected_features]);
for i from 0 to length (selected_features) - 1 do
    for j from i + 1 to length (selected_features) do
        if abs(correlation_matrix [i, j] > redundancy_threshold then
            Remove selected_features [j] from selected_features;
            Break;
        end
    end
end
return selected_features;
```

Step-by-Step Description:

Load the Dataset: The dataset containing all the features and the target variable (label indicating DDoS attack or not) is loaded for processing.

Define Initial Feature Set: A predefined set of features is considered, such as pktcount, bytecount, dur, packetins, tx_kbps, etc. These are extracted from the network flow statistics.

Set Target Variable: The target variable is defined as the presence or absence of a DDoS attack.

Initialize Selected Features List: An empty list is prepared to store features that meet the selection criteria.

Set Correlation Threshold: A threshold value (0.5 in this case) is set to decide which features have a strong enough relationship with the target variable to be considered relevant.

Feature Relevance Check (Loop through each feature):

For each feature in the dataset: Calculate the correlation between the feature and the target variable. If the absolute value of this correlation exceeds the predefined threshold, the feature is added to the selected features list.

Set Redundancy Threshold: A second threshold (0.7) is set to identify and remove highly correlated features among those already selected, to avoid redundancy.

Compute Correlation Matrix: A pairwise correlation matrix is generated for the features in the selected list to check for inter-feature redundancy.

Remove Redundant Features (Nested Loop):

Each pair of selected features is compared: If two features are highly correlated (above the redundancy threshold), the latter feature in the pair is removed. Once a redundant feature is removed, the inner loop breaks to prevent cascading removals.

Return Final Selected Features: After filtering out irrelevant and redundant features, the remaining list of selected features is returned for use in model training.

5.3. DDoS Attack Detection Model Based on DNN

To detect DDoS attacks targeting the SDN controller, a Deep Neural Network (DNN) model was trained using features selected through Correlation Feature Selection. The detection function is defined as $f(x) = y_i$, where $y_i \in \{0, 1\}$ a value of 0 indicates normal traffic, and 1 indicates an attack. The overall DNN-based detection process is illustrated in Fig. 6.

In Fig. 6. we present the general workflow of the submitted DDoS attack detection model using the Deep Neural Network (DNN) in SDN. The first step in the process is to identify the important features from the dataset which is done using Correlation Feature Selection (CFS) which is used to reduce the dimensionality of the data and increase the efficiency of modeling by excluding some redundant features. The selected features are used to construct the training data set that serves as the input of the DNN-based detection model. After processing this input, the trained DNN model divides the network traffic into two groups: attack and normal. The performance of the detection system is evaluated by further analyzing the model outputs using evaluation metrics based on this classification. These metrics, which together assess how well the model separates malicious from legitimate traffic, usually consist of accuracy, precision, and F1-score. The accurate and effective detection of DDoS attacks directed at the SDN controller is ensured by this methodical approach.

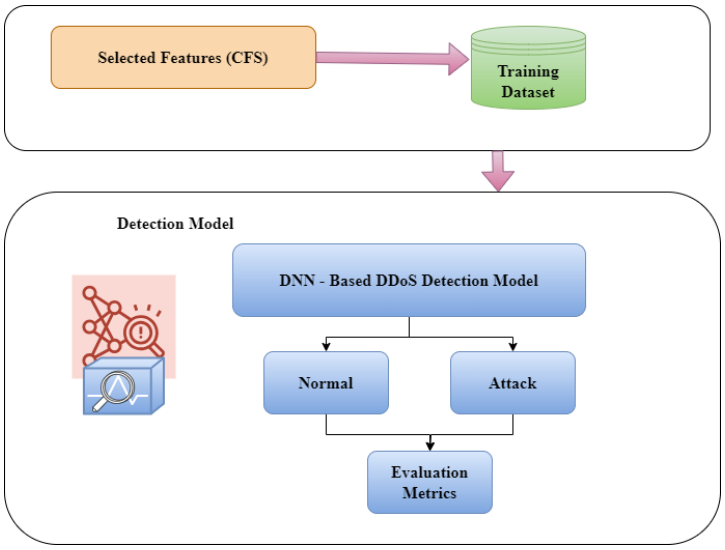

Fig.6. DNN-based detection model process

6. Results of Experiment

The following results of the experiment and the metrics used to test the proposed method. The details on the hardware used for experimentation: an Intel i5 processor with a CPU speed of 3.20 GHz, 16 GB RAM, and a high-speed Ethernet/Wi-Fi network environment.

6.1. Evaluation Metrics

The accuracy in detection as well as the FPR, false-positive rate (FPR) and precision as well as the F1 measurement were utilized to evaluate the proposed method. The confusion matrix found in Table 5 was utilized to calculate these measurements.

True Positive (TP) denotes that the classifier has correctly identified the assault, whereas False Negative (FN) describes circumstances in which the classifier has mistakenly classified an attack as normal. The terms False Positive (FP) and True Negative (TN) describe situations in which the classifier properly classifies normal cases and wrongly labels benign examples as malicious. Additional assessment measures that researchers employ in their research include F-measure, detection accuracy (DA), false alarm rate (FAR), and precision. Additional assessment measures that researchers employ in their research include F-measure, detection accuracy (DA), false alarm rate (FAR), and precision.

Table 5. Confusion matrix

	Class Prediction	
	Normal	Attack
Actual Class	TN	FP
	FN	TP

- The percentage of attacks successfully predicted from a set of test samples is represented by a precision metric. Precision can be calculated using equation (3).

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

Divide the number of attack samples by the number of normal samples to find the false-alarm rate (FAR) of an assault. FAR is calculated using equation (4).

$$FAR = \frac{FP}{TN+FP} \tag{4}$$

Detection Accuracy: This statistic calculates the percentage of cases that have been correctly labeled relative to the total number of cases. Therefore, only a balanced dataset can make it beneficial. Equation (5) can be used to calculate accuracy.

$$Detection\ Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{5}$$

- The F1-Measure calculates the harmonic mean of precision and recall to evaluate a system's accuracy. The F1 measure can be calculated using equation (6) below:

$$F1\ Measure = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{6}$$

Table 6. Number of unique values in the numerical features

Numerical Features	Unique Value
dt	781
switch	10
pktcount	8644
bytecount	8844
dur	838
dur_nsec	1000
tot_dur	3928
flows	14
packetins	161
pktperflow	2049
byteperflow	2730
pktrate	445
Pairflow	2
port_no	5
tx_bytes	10976
rx_bytes	10378
tx_kbps	1724
rx_kbps	1655
tot_kbps	2138

In this study, IDS performance was evaluated using performance evaluation metrics that are widely accepted in the field. As a result, all these metrics are utilized in evaluating the proposed approach.

6.2. Correlation-based Feature Selection Results

A feature's correlation coefficient determines whether it is strongly correlated with the target variable or lowly correlated with other features. Multicollinearity may be mitigated by removing features with low correlation coefficients. Features with high correlation coefficients are deemed informative for DDoS detection. The correlation threshold refines the feature selection process by removing features below the threshold. Correlation threshold is set at 0.5. By setting this threshold, the number of selected features can be controlled, ensuring that only the most relevant ones are retained. Finally, our selected features are validated using cross-validation techniques. DDoS detection models are strengthened by selecting features that contribute effectively to their predictive performance. In order to prevent overfitting, we validate the selected features and ensure generalization capability. There are two types of features that can be classified as numerical features: 19 and categorical features: 3. Table 6 shows the number of unique values among the numerical features and equation (7) demonstrated the CFS.

Numerical Feature = {'dt', 'switch', 'pktcount', 'bytecount', 'dur', 'dur_nsec', 'tot_dur', 'flows', 'packetins', 'pktperflow', 'byteperflow', 'pktrate', 'Pairflow', 'port_no', 'tx_bytes', 'rx_bytes', 'tx_kbps', 'rx_kbps', 'tot_kbps'}

Categorical Feature = {'src', 'dst', 'Protocol'}

$$CFS = bytecount, byteperflow, pktrate, Pairflow, tot\_kbps \tag{7}$$

The subsequent stage uses the CFS as an input. Feature selection based on correlation is shown in Fig. 7. This demonstrates how the proposed feature based on correlation choice was made to select the most significant characteristic CFS that has the greatest value to detect DDoS attacks against SDNs. Moreover, it decreases the number of features from 22 to 5, which lowers the processing time required to train the deep learning model. A detection model is generated from the features CFS using a DNN.

7. Results and Discussion

The CFS features are used to generate a DNN-based training model based on the architectures listed in Table 7. Due to the fact that DNN needs extensive training as well as lengths of inference it is designed to be developed offline in order to avoid the issue. To identify DDoS attacks, detection models are created for SDN controllers and run online. Because the experiments were based on simulated datasets, they were repeated three times to ensure consistency. In addition, a cross-validation test technique is also used. 25% of the data are used to test the system, and 75% is utilized for training.

7.1. DNN Model Setup

This subsection's goal is to give comprehensive details on the learning model's architecture and configuration. Table 7 shows the architecture of a DNN hyper parameter that has been trained using selected features of the CFS.

Table 7. DNN model architecture

Layer (type)	Output Shape	Param #
Hidden_Layer_1 (Dense)	(None, 28)	1568
Hidden_Layer_2 (Dense)	(None, 10)	290
Output_Layer (Dense)	(None, 1)	11
Total params: 1869		
Trainable params: 1869		
Non-trainable params: 0		

7.2. Hyperparameter Tuning

The parameter "hp" is assumed to be an instance of Hyperparameter. Accordingly, the function is intended to be used within the context of hyperparameter tuning, where different hyperparameters are tested to determine which configuration performs best. A sequential model is formed by stacking layers linearly. The model is updated with three additional layers. Activation functions of the first layer are ReLU (Rectified Linear Unit), which has 28 units/neurons. The first layer has an input shape of 55. With 10 units and ReLU activation, the second layer is also dense. An output layer with a single neuron and sigmoid activation is present in the third layer, making it appropriate for applications involving binary classification.

Deep learning algorithms rely heavily on the learning rate, which determines the magnitude of the model's weight updates during each iteration. The Adam optimizer was tested with learning rates of 0.01, 0.001, and 0.0001. Among these, the 0.01 learning rate resulted in the highest detection rate, as presented in Fig. 9. To mitigate overfitting, a method of early stopping was implemented. If the validation loss does not decrease over several consecutive iterations (patience), the training is halted automatically. This not only avoids unnecessary computations but also helps preserve the generalization capability of the model.

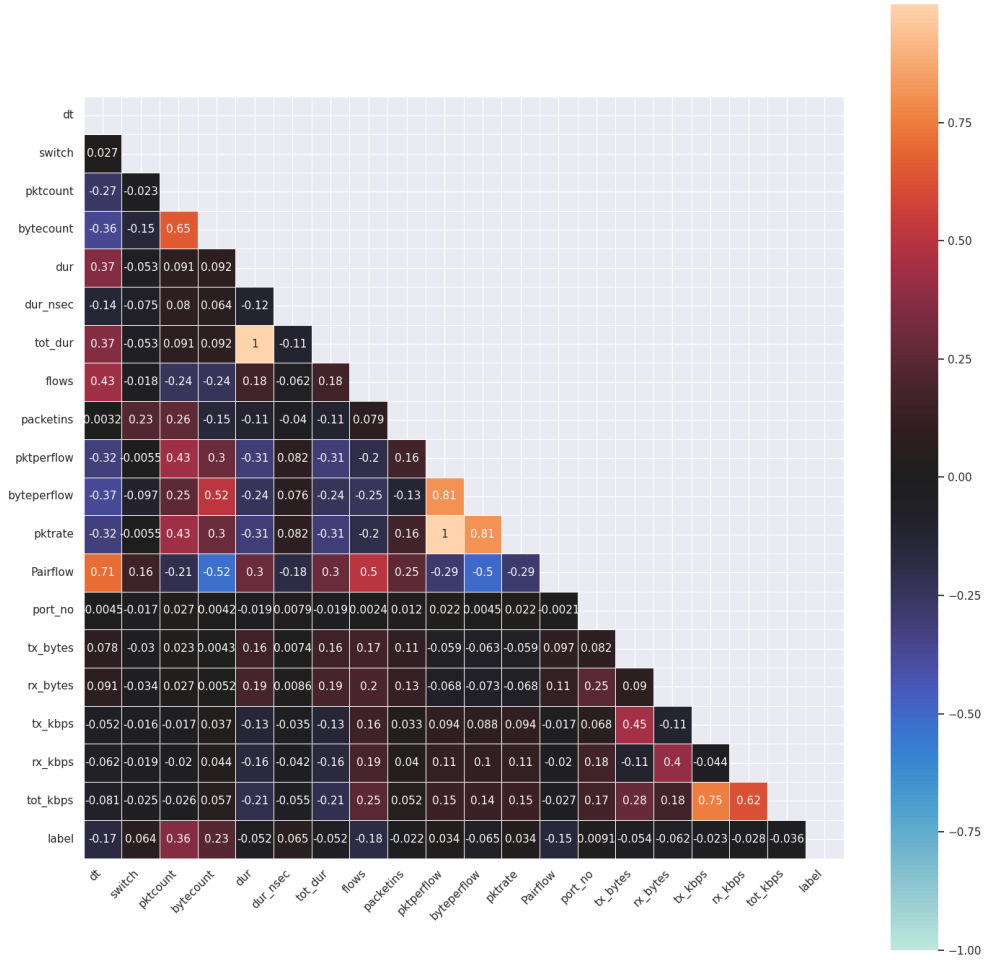

Fig.7. Correlation based feature selection

To ensure reproducibility and effective convergence, epoch scheduling and batch size were carefully selected. The number of epochs was capped at 100, as experiments showed that the model's accuracy plateaued beyond this point. Batch size was set to 32, balancing training stability and computational efficiency. This choice was made after empirical evaluations, where smaller batch sizes led to noisy gradient updates and larger ones slowed convergence. The selected settings allowed the model to converge reliably while maintaining performance across validation sets. We consider the DNN model to be optimal since it successfully converged within 100 iterations using the most effective learning rate and batch configuration. Fig. 8 shows the DNN hyperparameter model code, and Fig. 9 presents the performance metrics across the three evaluated learning rates (0.01, 0.001, and 0.0001).

```
def model_builder(hp):
    model = keras.Sequential()

    model.add(Dense(28 , input_shape=(55,) , activation="relu" , name="Hidden_Layer_1"))
    model.add(Dense(10 , activation="relu" , name="Hidden_Layer_2"))
    model.add(Dense(1 , activation="sigmoid" , name="Output_Layer"))

    opt = keras.optimizers.Adam(learning_rate=hp.Choice('learning_rate',[1e-2, 1e-3, 1e-4]))

    model.compile(optimizer=opt, loss='binary_crossentropy', metrics=['accuracy'])

    return model
```

Fig.8. DNN model architecture code

Fig.9. Performance metrics with learning rate

Sigmoid functions, also known as logistic functions, are used in neural networks, especially for binary classification tasks. This algorithm maps any real-valued number on a real-valued scale to the range [0, 1]. The sigmoid function is in the following equation (8).

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{8}$$

In neural networks, particularly in deep learning models, Rectified Linear Units (ReLUs) are considered activation functions. Input values that are negative are replaced with zero. Positive values remain unchanged. Equation (9) defines ReLU's activation function.

$$ReLU(x) = \max(0, x) \tag{9}$$

Table 8. Result performance with 3 times

Learning Rate	Accuracy %	Precision %	F1-Measure	False Positive Rate
0.01	99.39	97.41	99.32	0.0665
0.001	99.25	97.18	99.13	0.0689
0.0001	98.77	96.76	98.67	0.0747

7.3. Accuracy Results Comparison between DNN and ML Approaches

Using the same dataset and testing environment, this subsection evaluates Deep Neural Networks vs Machine Learning approaches fairly. Based on accuracy, Table 9 compares the proposed approach DNN with other Machine Learning approaches "KNN", "RBF SVM", "Decision Tree", "Naive Bayes", "Quadratic", "SGD", "Logistic Regression" and "XGBoost". The accuracy results of DNN with different machine learning approaches are shown in Fig. 10.

Table 9. Accuracy results comparison between DNN and ML approaches

Name	Accuracy
DNN	99.394410
XGBoost	98.251293
RBF_SVM	97.230921
KNN	96.470919
Decision Tree	96.126104
Quadratic	84.289786
Logistic Regression	83.318673
SGD	83.262376
Naive Bayes	69.606981

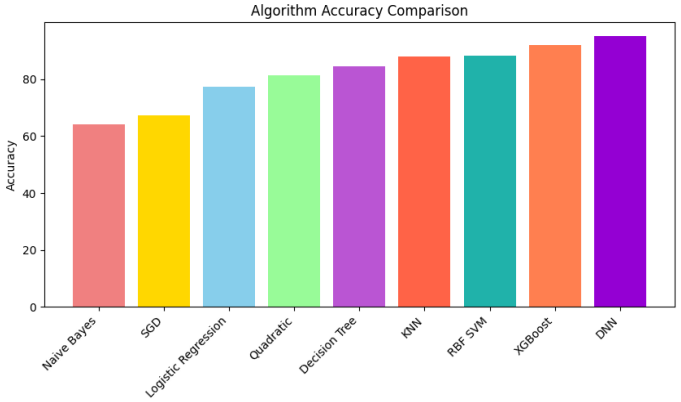

Fig.10. Accuracy results comparison between DNN and ML approaches

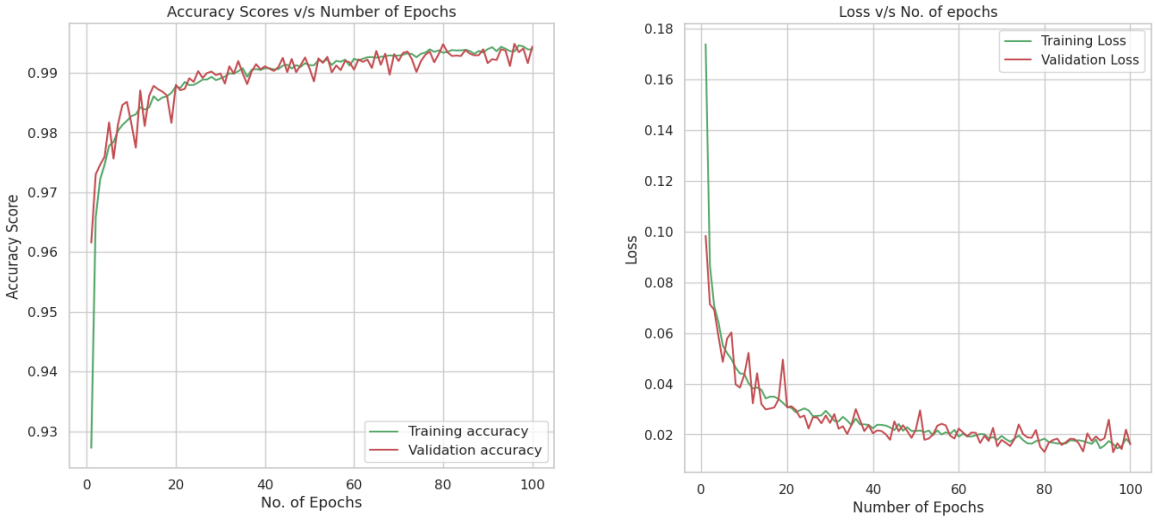

Fig.11. DNNADSC model accuracy and loss

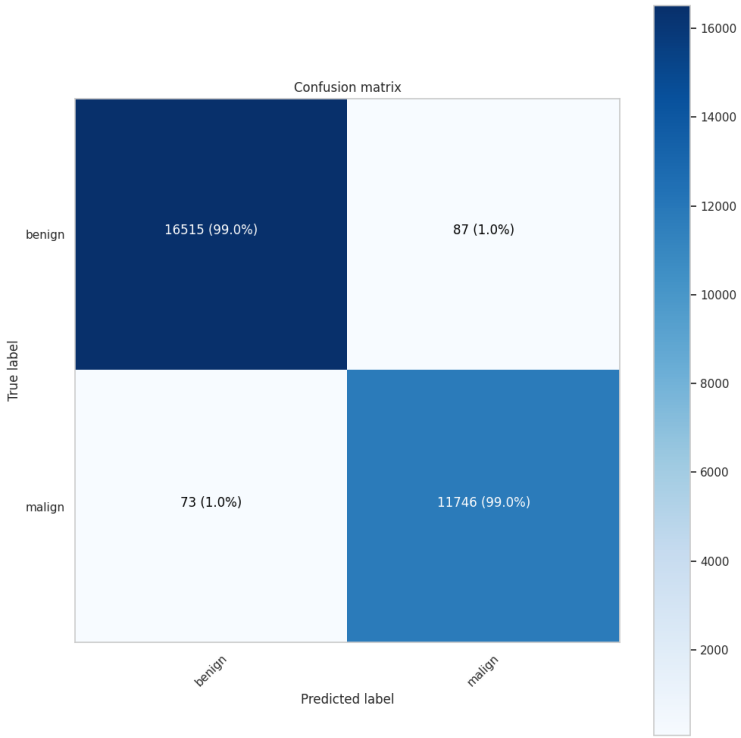

Fig.12. DNNADSC confusion matrix

Fig. 11 illustrates the performance of the proposed DNNADSC model over 100 training epochs. In this graph, we can see how the model has performed over a number of epochs during the training process, including its 99.39% accuracy. The validation losses for DNNADSC are shown in Fig. 11. Validation losses were lower when CFS was applied to the proposed model. Based on 100 validation epochs, DNNADSC achieved a validation loss of 0.0149. A strategy of feature selection might reduce the number of features used in the training phase, which was ignored in previous research.

The SDN DDoS attack dataset confusion matrix is shown in Fig. 12. Confusion matrix showing how classification models distinguish between DDoS attack types. Our model predicts the attacks accurately with 0.5 thresholds. There were 16,515 cases predicted as benign that were actually benign. A total of 87 cases were predicted to be malign but turned out to be benign. 73 cases were misidentified as benign when they were actually malignant. 11,746 cases were predicted as malign and were actually malignant.

Based on a number of metrics, including (i) the proposed approach, (ii) the kind of dataset used, (iii) average detection accuracy, (iv) precision, (v) the false positive rate (FPR), (vi) the F1-measure, and (vii) the detection time, Table 10 compares the proposed approach with alternative approaches. The Receiver Operating Characteristic (ROC) curve in Fig. 13. visually illustrates the classification performance of the proposed model across various threshold settings. Our model with perfect classification would pass through the point (0, 1) - meaning no false positives and 100% true positives. Since the plotted curve in Fig. 13. stays well above the diagonal and approaches the top-left corner, it shows that the model performs well in distinguishing between positive and negative classes.

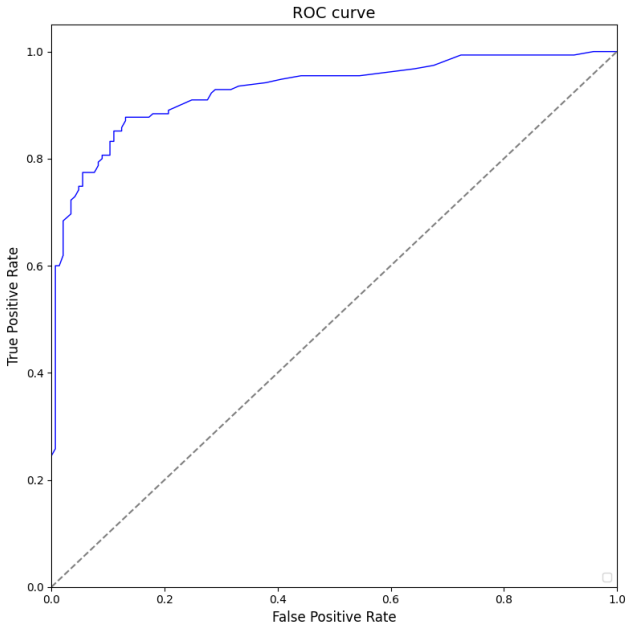

Fig.13. DNNADSC roc curve

Table 10. DNNADSC compared to existing approaches

Metrics	Model	Accuracy	F1-Measure	Precision	FPR	Dataset	Detection Time (s)
Proposed Approach	DNNADSC	99.39%	99.32%	97.41%	0.0665	DDoS SDN Dataset	1.538
Alanazi et al [22]	Ensemble Model (CNN, LSTM, GRU)	99.77%	97.65%	98.78%	0.0986	Non SDN Dataset	-
Wan et al [2]	LSTM	98.53%	97.64%	96.31%	-	Non SDN Dataset	-
Lee et al. [21]	MLP, CNN, SAE	98.35%	98.10%	96.63%	-	Non SDN Dataset	-
Boukria et al. [23]	DL Algorithm	96.77%	93.54%	88.47%	0.1274	Non SDN Dataset	1.681

Making use of the tabled comparison in Table 10, it becomes evident that the DNNADSC method proposed in our study exhibited exceptional performance metrics. In particular, 99.39% detection accuracy, 99.25% precision, 0.0665 false positive rate (FPR), and 99.32% F1-measure were demonstrated by the method. Furthermore, the detection process was completed within a swift time frame of 1.538 seconds. These findings underscore the efficacy and efficiency of the DNNADSC method in the context of our research. Alanazi et al. created an ensemble model consisting of CNNs, LSTMs, and GRUs. The accuracy was 99.77%, the F1-measure was 97.65%, the precision was 98.78%, and the false

positive rate was 0.0986. Wan et al. developed a model utilizing Long Short-Term Memory (LSTM) networks, achieving notable performance metrics. Specifically, their model attained an accuracy of 98.53%, an F1-measure of 97.54%, and a precision of 96.31%. Lee et al. devised a model employing Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), and Stacked Autoencoders (SAE), achieving impressive performance metrics. Specifically, their model yielded an accuracy of 98.3%, an F1-measure of 98.1%, and a precision of 96.6%. Boukria et al. developed a model leveraging Deep Learning (DL) algorithms, achieving notable performance across various metrics. Specifically, their model demonstrated an accuracy of 96.77%, an F1-measure of 93.546%, and a precision of 88.472%. Additionally, the false positive rate (FPR) was recorded at 0.1274, while the detection rate stood at 1.681. In addition to a much faster detection rate, the method proved its efficacy and effectiveness in the detection of DDoS attacks.

8. Conclusions and Future Work

Security weaknesses pose a grave threat for SDN networks since they could be exploited by hackers to carry out denial of service attacks on SDN controllers. This paper presents a unique DL-powered technique called DNNADSC, which aims to detect DDoS assaults on SDN controllers in order to address this difficulty. Three steps make up the suggested methodology, which effectively detects DDoS attacks. The initial stage involves data preprocessing, followed by the utilization of a correlation-based feature selection method in the second stage. Choosing the highest-scoring features to train the DNN model is the final step of the third stage.

Four common metrics are used to evaluate the proposed DNNADSC approach: average detection accuracy, average F1-measure, average detection duration and average FPR. With an average FPR of 0.0665, an average F1-measure of 99.32%, average detection accuracy of 99.39%, and precision of 97.41%, the testing findings show that the suggested strategy achieves exceptional results. These results highlight the capability of DNNADSC in accurately identifying DDoS attacks aimed at SDN controllers. Moreover, comparative analysis with other DL-based approaches demonstrates notable performance enhancements achieved by DNNADSC. The proposed DNNADSC approach shows promising potential for advancing DDoS detection techniques.

In future work to improve feature selection and model performance by incorporating more bio-inspired optimization algorithms like Ant Colony Optimization, Firefly and Bat Algorithm into the proposed DDoS detection model. We also intend to evaluate the model using real-world SDN traffic data outside of the present simulation-based environment to increase the practical usefulness of the approach. This will enable us to evaluate the flexibility of the model in adjusting traffic patterns and real network conditions. We also aim to look at the scalability and efficiency of distributed SDN systems.

References

[1] J. E. Varghese and B. Muniyal, "An Efficient IDS Framework for DDoS Attacks in SDN Environment," *IEEE Access*, vol. 9, pp. 69680–69699, 2021, doi: 10.1109/ACCESS.2021.3078065.

[2] L. Wan, Q. Wang, and S. Zheng, "Deep SSAE-BiLSTM Model for DDoS Detection in SDN," *Proc. - 2021 2nd Int. Conf. Comput. Commun. Netw. Secur. CCNS* 2021, pp. 34–37, 2021, doi: 10.1109/CCNS53852.2021.00015.

[3] V. Deepa, K. Muthamil Sudar, and P. Deepalakshmi, "Detection of DDoS attack on SDN control plane using hybrid machine learning techniques," *Proc. Int. Conf. Smart Syst. Inven. Technol. ICSSIT* 2018, no. February 2023, pp. 299–303, 2018, doi: 10.1109/ICSSIT.2018.8748836.

[4] V. Umarnani, D. R. Jitendra, and S. Chouhan, "Enhancing Network Security through Software Defined Networking (SDN)."

[5] M. Alshinwan et al., "Dragonfly algorithm: a comprehensive survey of its results, variants, and applications," *Multimed. Tools Appl., no. February, 2021*, doi: 10.1007/s11042-020-10255-3.

[6] S. Dong and M. Sarem, "DDoS Attack Detection Method Based on Improved KNN with the Degree of DDoS Attack in Software-Defined Networks," *IEEE Access, vol. 8, pp. 5039–5048, 2020*, doi: 10.1109/ACCESS.2019.2963077.

[7] N. Ahuja, G. Singal, and D. Mukhopadhyay, "DLSDN: Deep learning for DDOS attack detection in software defined networking," *Proc. Conflu. 2021 11th Int. Conf. Cloud Comput. Data Sci. Eng.*, pp. 683–688, 2021, doi: 10.1109/Confluence51648.2021.9376879.

[8] A. Mahalingam, G. Perumal, G. Subburayalu, and M. Albathan, "ROAST-IoT : A Novel Range-Optimized Attention Convolutional," 2023.

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

[10] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Intrusion detection in sdn-based networks: Deep recurrent neural network approach," *Adv. Sci. Technol. Secur. Appl.*, pp. 175–195, 2019, doi: 10.1007/978-3- 030-13057-2 8.

[11] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "Generating network intrusion detection dataset based on real and encrypted synthetic attack traffic," *Appl. Sci.*, vol. 11, no. 17, 2021, doi: 10.3390/app11177868.

[12] Q. Zhou and Di. P. Pezaros, "BIDS: Bio-Inspired, Collaborative Intrusion Detection for Software Defined Networks," IEEE Int. Conf. Commun., vol. 2019-May, pp. 1–6, 2019, doi: 10.1109/ICC.2019.8761410.

[13] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. El Moussa, "DeepIDS: Deep learning approach for intrusion detection in software defined networking," *Electron.,* vol. 9, no. 9, pp. 1–18, 2020, doi: 10.3390/electronics9091533.

[14] R. Ben Said, Z. Sabir, and I. Askerzade, "CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking with Hybrid Feature Selection," *IEEE Access*, vol. 11, no. November, pp. 138732–

138747, 2023, doi: 10.1109/ACCESS.2023.3340142.
[15] V. Sivagaminathan, M. Sharma, and S. K. Henge, "Intrusion detection systems for wireless sensor networks using computational intelligence techniques," *Cybersecurity*, vol. 6, no. 1, 2023, doi: 10.1186/s42400-023-00161-0.
[16] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Sci. Int. Digit. Investig.*, vol. 32, p. 200892, 2020, doi: 10.1016/j.fsidi.2019.200892.
[17] G. Kocher and G. Kumar, "Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges," *Soft Comput.*, vol. 25, no. 15, pp. 9731–9763, 2021, doi: 10.1007/s00500-021- 05893-0.
[18] M. Vishwakarma and N. Kesswani, "DIDS: A Deep Neural Network based real-time Intrusion detection system for IoT," *Decis. Anal. J.*, vol. 5, no. September, p. 100142, 2022, doi: 10.1016/j.dajour.2022.100142.
[19] B. Č elesova´, J. Val'ko, R. Grez˘o, and P. Helebrandt, "Enhancing security of SDN focusing on control plane and data plane," 7th *Int. Symp. Digit. Forensics Secur. ISDFS* 2019, pp. 1–6, 2019, doi: 10.1109/ISDFS.2019.8757542.
[20] F. Khashab, J. Moubarak, A. Feghali, and C. Bassil, "DDoS Attack Detection and Mitigation in SDN using Machine Learning," *Proc. 2021 IEEE Conf. Netw. Softwarization Accel. Netw. Softwarization Cogn. Age, NetSoft* 2021, pp. 395–401, 2021, doi: 10.1109/NetSoft51509.2021.9492558.
[21] J. Yu, X. Ye, and H. Li, "A high precision intrusion detection system for network security communication based on multi-scale convolutional neural network," *Futur. Gener. Comput. Syst.*, vol. 129, pp. 399–406, 2022, doi: 10.1016/j.future.2021.10.018.
[22] F. Alanazi, K. Jambi, F. Eassa, M. Khemakhem, A. Basuhail, and K. Alsubhi, "Ensemble Deep Learning Models for Mitigating DDoS Attack in Software-Defined Network," *Intell. Autom. Soft Comput.*, vol. 33, no. 2, pp. 923–938, 2022, doi: 10.32604/iasc.2022.024668.
[23] O. Sbai and M. Elboukhari, "Intrusion Detection System For Manets Using Deep Learning Approach," *Int. J. Com- put. Sci. Appl.*, vol. 18, no. 1, pp. 85–101, 2021.
[24] B. Lee, J. Kim, and M. Choi, "Federated Learning Based Network Intrusion Detection Model," 2023 24st 24st *Asia-Pacific Netw. Oper. Manag. Symp.*, pp. 330–333, 2023.
[25] M. Mittal, K. Kumar, and S. Behal, "Deep learning approaches for detecting DDoS attacks: a systematic review," *Soft Comput.*, vol. 27, no. 18, pp. 13039–13075, 2023, doi: 10.1007/s00500-021-06608-1.
[26] A. H. Mohammad, T. Alwada'n, O. Almomani, S. Smadi, and N. ElOmari, "Bio-inspired Hybrid Feature Selection Model for Intrusion Detection," *Comput. Mater. Contin.*, vol. 73, no. 1, pp. 133–150, 2022, doi: 10.32604/cmc.2022.027475.
[27] Z. D. A. Zaripova and D. Anvarovna, "Network security issues and effective protection against network attacks."
[28] A. E. Ibor, O. B. Okunoye, F. A. Oladeji, and K. A. Abdulsalam, "Novel Hybrid Model for Intrusion Prediction on Cyber Physical Systems' Communication Networks based on Bio-inspired Deep Neural Network Structure," *J. Inf. Secur. Appl.*, vol. 65, no. January, p. 103107, 2022, doi: 10.1016/j.jisa.2021.103107.
[29] M. T. Ali A. Ghorbani, Wei Lu, "Network Intrusion Detection and Prevention Advances in Information Security," *Inf. Syst.*, p. 223, 2010.
[30] M. S. El Sayed, N. A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 4, pp. 1862–1880, 2022, doi: 10.1109/TCCN.2022.3186331.

Authors' Profiles

Jeyavim Sherin R. C. received her B.E. and M.E. in Computer Science and Engineering in 2007 and 2011, respectively, from C.S.I Institute of Technology, Thovalai, Tamilnadu, and Karpagam University, Coimbatore, Tamilnadu. She is currently pursuing a Ph.D. at Vellore Institute of Technology, Chennai, in the School of Computer Science & Engineering. Her research interests include deep learning, the Internet of Things, and network security.

Parkavi K. has served as a Senior Assistant Professor at Vellore Institute of Technology in Chennai since 2020. She has almost nineteen years of experience in both teaching and research. She oversees research scholars and teaches undergraduate and graduate courses in addition to spearheading research initiatives. In addition to books at the undergraduate level, she has written over thirty articles in national and international publications and conferences. Her areas of interest in study include computer vision, machine learning, deep learning, network security, and cyber-physical systems.

How to cite this paper: Jeyavim Sherin R. C., Parkavi K., "Software-defined Networking Controller for Detection of DDoS Attacks Based on Deep Neural Networks", International Journal of Computer Network and Information Security(IJCNIS), Vol.17, No.4, pp.1-18, 2025. DOI:10.5815/ijcnis.2025.04.01