International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V73I6P124

Volume 73 Issue 6, 292-301, June 2025
© 2025 Seventh Sense Research Group®

Original Article

Ensemble Deep Learning-Based Ddos Classification in Sflow-Enabled Software-Defined Networking

S. Natesan[1], C. Sivakumar[2]

[1,2]School of Computing, Department of CSE, Mohan Babu University, Tirupati, Andhra Pradesh, India.

[1]Corresponding Author: natesan12345@gmail.com

Received: 06 March 2025 Revised: 10 May 2025 Accepted: 07 June 2025 Published: 28 June 2025

Abstract - *Software-Defined Networking (SDN) exhibits a programmable architecture that decouples the control plane from the data plane, improving network management. On the other hand, this centralizing makes the network more susceptible to Distributed Denial of Service (DDoS) attacks, which might easily overwhelm the resources of SDN and lead to network failure. Early identification of such risks is necessary for the continuous stability of SDN. The slow approach's real-time traffic monitoring capabilities enable one to efficiently sample flow-based data. This helps reduce processing overhead in network switches and provides good attack detection. Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), and ResNet make up the Stacked Deep Ensemble Model applied in this research project; all these components contribute to accurate DDoS classification achievement. GNS 3 models the topology of the network while the SDN environment is built using the ONOS SDN Controller. SFlow helps with data collecting; Prometheus acts as a time-series database to store traffic data. Using both the NSL-KDD and UNSW-NB15 benchmark datasets, the proposed approach is assessed to be resilient over a wide range of attack scenarios. Since the ensemble model reduces the number of false positives found and efficiently achieves higher classification accuracy, the experiments show that it performs better than conventional detection methods.*

Keywords - *DDoS detection, Software-Defined Networking, sFlow, Deep learning, Ensemble model.*

1. Introduction

A revolution in network management follows from the increasing acceptance of Software-Defined Networking (SDN). By decoupling the data plane and the control plane, one can attain this and so enable centralized network programmability and flexibility [1-3]. At the same time, this architectural modification allows dynamic traffic control, security policies, and network optimization while increasing scalability and automation.

On the other hand, The centralized nature of SDN generates serious security problems, mostly connected to vulnerabilities in Distributed Denial of Service (DDoS) attacks. These attacks might overwhelm network controllers, influencing performance or causing complete failure. DDoS attacks use the architectural vulnerability of SDN, so depleting resources and disrupting regular operations by flooding the control plane with hostile traffic [1-3].

2. Challenges

Several challenges arise in SDN systems due to the detection and mitigation of DDoS attacks. First, as a single point of failure, software-defined networking controllers are prime targets for attackers who aim to disturb network operations [4]. Second, particularly those generated by botnets and advanced adversaries, conventional DDoS detection methods depending on rule-based or signature-based intrusion detection find it difficult to adapt to evolving attack patterns [5]. Moreover, effective traffic monitoring without unnecessarily running computational overhead on network switches determines real-time threat detection in SDN. This is so because more processing could compromise the general network performance [6]. Current solutions often have low adaptability to modern attack strategies, ineffective use of resources, and high false positives [4-6].

2.1. Problem Definition

Standardized controllers in SDN expose the network to mass DDoS attacks, compromising the functionality of the data plane and control planes' functionality [7]. Regarding ever-shifting threat environments, traditional security systems are insufficient since they rely on predefined attack signatures or limited computational capacity [8].

Moreover, most conventional approaches depend on deep packet inspection, which increases processing costs in environments running SDN services [9] and generates latency. A lightweight but potent solution combining intelligent traffic monitoring with sophisticated deep learning models is needed to improve real-time attack detection [10].

3. Objectives

A DDoS detection system built on sFlow-based real-time traffic monitoring will help lower the load on SDN switches. To propose a deep ensemble learning approach to reduce false positives count and increase classification accuracy. Combining ResNet, Recurrent Neural Networks (RNN), and Deep Neural Networks (DNN)

3.1. Novelty

The proposed approach maximizes the accuracy and resource economy of the DDoS detection process by combining sFlow-based sampling with a Stacked Deep Ensemble Model. By means of real-time traffic data acquired by the sFlow system, this model ensures flexibility enough to match evolving attack strategies. By contrast, conventional deep learning-based methods rely on stationary datasets. Moreover, several deep learning architectures improve detection robustness, surpassing single models in especially difficult attacks.

3.2. Contributions

1. In SDN, using flow-based sampling with sFlow lowers processing overhead and helps to detect real-time attacks.
2. DNN, RNN, and ResNet improve DDoS classification accuracy using traffic characteristics, such as spatial-temporal traffic characteristics.
3. Real-world SDN systems present practical relevance of the ONOS SDN Controller and GNS3-based topology simulation.

4. Related Works

Using a range of techniques leveraging machine learning, statistical analysis, and flow-based monitoring systems [11-16], DDoS detection in Software-Defined Networks (SDN) is extensively investigated. Early works mostly depended on rule-based and anomaly detection techniques, which lacked the flexibility to fit changing attack patterns even if they were good against known hazards [11]. These methods proved successful against known hazards. Recently, deep learning-based solutions have drawn the most interest in new advancements. These solutions comprise hybrid models, Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNNs [12]). Better classification performance depends on these components. In many studies, research on DDoS detection has focused much on flow-based monitoring.

As a means of network behavior analysis, statistical feature-based approaches employing entropy, correlation, and threshold-based anomaly detection have been proposed [13]. Although they successfully identify volumetric attacks [], these methods cannot manage more complex distributed denial of service attacks, such as low-rate or multi-vector attacks, in which traffic patterns are more difficult to distinguish from legitimate fluctuations [14]. It is shown that deep learning models show encouraging performance in terms of increasing detection accuracy and adaptation. Spatial and temporal aspects from network traffic have been extracted using conventional architectures, including Long Short-Term Memory (LSTMs) and Convolutional Neural Networks (CNNs).

However, when applied to real-world traffic patterns showing dynamic changes, single-model techniques sometimes have limitations in terms of their generalizability. Combining several deep learning architectures is shown as a solution to this problem [16] using ensembles of learning techniques. These methods aim to improve such systems' general resilience and classification performance.

Moreover, under research is the possibility of combining sFlow-based monitoring with machine learning approaches. Flow sampling offers traffic insights that are not only lightweight but also instructive, thus reducing the processing demand of SDN controllers, according to previous studies [13]. However, most of the studies conducted up until now have concentrated on traditional machine learning classifiers, including Support Vector Machines (SVM) and Random Forest. Even if they are efficient, these classifiers do not fully exploit the possibilities of deep learning for intricate attack scenarios [14-16].

However, to offer complete DDoS classification, our proposed approach combines real-time sFlow monitoring with a Stacked Deep Ensemble Model using DNN, RNN, and ResNet. This hybrid architecture improves detection accuracy and adaptability to evolving threats, overcoming the limitations of single-model architectures. Moreover, evaluation over a spectrum of attack paths guarantees that the model performs better using the NSL-KDD and UNSW-NB15 datasets. Among other things, the results of the experiments reveal improved accuracy, false positive rates, and computational economy. Based on SDN, this performance shows that deep learning can be included in security systems. To provide a scalable and adaptable solution for DDoS mitigating, the proposed model advances state-of-the-art SDN security significantly. We reach these advances with real-time sFlow traffic monitoring, ensemble deep learning, and efficient SDN integration applications.

4.1. Proposed Method

To propose an accurate classification of attack patterns in SDN systems, the proposed DDoS detection system combines sFlow-based real-time traffic monitoring with a Stacked Deep Ensemble Model. The sFlow collector starts the process in the first phase. It regularly samples network traffic, at which point it can record important flow statistics and concurrently lower processing overhead on SDN switches. For further processing, traffic measurements are derived from Prometheus, the time-series database containing the sampled data. First, a feature extraction module will select the features pertinent to classification to standardize the obtained flow data. DNN, RNN, and ResNet connections are subsequently stacked in a Stacked Deep Ensemble Model from the processed data. While the RNN detects temporal dependencies in traffic patterns, the DNN catches high-level abstract features; ResNet addresses vanishing gradients through skip connections, enhancing feature extraction. Still, the DNN is responsible for generating high-level abstraction by aggregating the forecasts generated by many networks using the ensemble model, one to increase resilience and classification accuracy. The last classification decision will determine whether the arriving traffic is normal or a part of a distributed denial of service attack. The recommended approach is a great fit for SDN systems since it guarantees real-time detection and calls only minimum computational overhead.

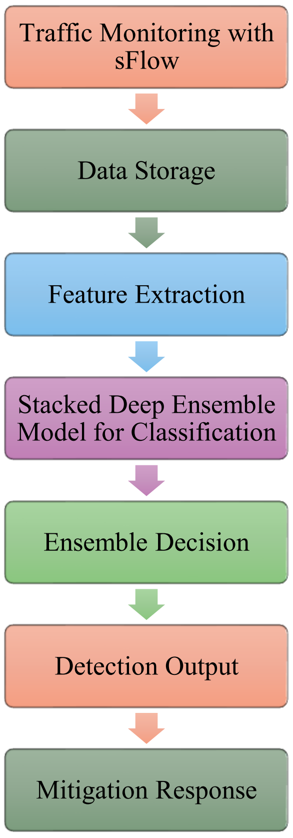

Fig. 1 Proposed process

4.2. Pseudocode for DDoS Detection using Stacked Deep Ensemble Model

```
# Import necessary libraries
import tensorflow as tf
from tensorflow. Keras. models import Sequential
from tensorflow. Keras. layers import Dense, LSTM, Conv1D, Flatten, Dropout
import numpy as np
# Step 1: Load and preprocess data from Prometheus (sFlow samples)
def load_data():
    data = retrieve_from_prometheus()  # Fetch traffic data from time-series database
    features, labels = preprocess(data)  # Normalize and extract relevant features
    return features, labels
# Step 2: Define individual models
def build_dnn(input_shape):
    model = Sequential([
        Dense(128, activation='relu', input_shape=input_shape),
        Dense(64, activation='relu'),
        Dense(32, activation='relu'),
        Dense(1, activation='sigmoid')  # Binary classification (Normal or DDoS)
    ])
    return model
def build_rnn(input_shape):
    model = Sequential([
        LSTM(64, return_sequences=True, input_shape=input_shape),
        LSTM(32),
        Dense(1, activation='sigmoid')
    ])
    return model
def build_resnet(input_shape):
    model = Sequential([
        Conv1D(64, kernel_size=3, activation='relu', input_shape=input_shape),
        Conv1D(64, kernel_size=3, activation='relu'),
        Flatten(),
        Dense(32, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    return model
# Step 3: Train models independently
def train_models(X_train, y_train):
    den = build_dnn((X_train.shape[1],))
    rnn = build_rnn((X_train.shape[1], 1))
    resnet = build_resnet((X_train.shape[1], 1))
    dnn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    rnn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    resnet.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    dnn.fit(X_train, y_train, epochs=10, batch_size=32, verbose=1)
    rnn.fit(X_train.reshape(-1, X_train.shape[1], 1), y_train, epochs=10, batch_size=32, verbose=1)
    resnet.fit(X_train.reshape(-1, X_train.shape[1], 1), y_train, epochs=10, batch_size=32, verbose=1)
    return dnn, rnn, resnet
# Step 4: Aggregate predictions from ensemble model
def ensemble_predict(dnn, rnn, resnet, X_test):
    dnn_pred = dnn.predict(X_test)
    rnn_pred = rnn.predict(X_test.reshape(-1, X_test.shape[1], 1))
    resnet_pred = resnet.predict(X_test.reshape(-1, X_test.shape[1], 1))
    final_pred = (dnn_pred + rnn_pred + resnet_pred) / 3  # Averaging predictions
    return (final_pred > 0.5).astype(int)  # Apply threshold for classification
# Step 5: Deploy the detection system
def detect_ddos():
    X, y = load_data()
    dnn, rnn, resnet = train_models(X, y)
    predictions = ensemble_predict(dnn, rnn, resnet, X)
    if np.any(predictions == 1):
        alert_admin("DDoS Attack Detected!")  # Trigger mitigation response
    else:
        print("No attack detected.")
# Execute the detection process
detect_ddos()
```

5. Traffic Monitoring with sFlow

Sampled Flow, the packet sampling system sFlow, makes real-time traffic monitoring in high-speed networks possible. Where the controller is the central location for network intelligence, sFlow helps reduce the amount of processing overhead by sampling just a subset of packets instead of processing every Flow in SDN. Every SDN switch boasts a sFlow agent built on top of it. This agent gathers flow statistics from network packets and forwards them, at regular intervals, to a centralized sFlow collector.

Aggregating the data acquired by this collector helps to keep overhead to a minimum and clearly shows the activity on the network.

1. Packet Sampling: sFlow agent embedded in SDN switches randomly samples network packets as they pass across the switch.
2. Statistical Aggregation: the analyzed sampled packets allow the extraction of extensive properties, including the source and destination IP addresses, the protocol type, the packet size, and the flow duration.
3. Transmission to sFlow Collector: The switch forwards the sampled data to the sFlow collector, which then compiles and stores the flow data for further analysis.
4. Forwarding to Prometheus: Prometheus, a time-series database, receives the processed data from the sFlow collector via forwarding for structured storage and retrieval.

Table 1. sFlow traffic data

Timestamp	Source IP	Destination IP	Protocol	Packet Size (Bytes)	Flow Duration (ms)
10:02:30	192.168.1.2	10.0.0.5	TCP	1500	120
10:02:31	172.16.4.8	192.168.2.1	UDP	512	80
10:02:32	10.0.0.7	192.168.1.3	ICMP	64	5

Prometheus then stores the acquired sFlow data so, enabling real-time monitoring as well as historical analysis of network traffic, as in Table 1.

5.1. Data Storage and Feature Extraction

Sampled flow data is stored by the sFlow collector; then, it is processed for feature extraction to ready it for presentation to the DDoS classification model. Transposing raw flow data into relevant inputs for the deep learning classifier is a crucial first step achieved during feature extraction. The obtained features mirror the network's behaviour, enabling exact differentiation between benign and malicious traffic.

5.2. Key Features Extracted

1. Packet Rate (PR): The Packet Rate (PR) is the quantity of packets passing a given source in one second.
2. Flow Entropy (FE): Higher values of Flow Entropy (FE) indicate attack possibility; it measures the randomness of source-destination connections.
3. Byte Count (BC): The total byte count moved over a given period of time is known as byte count, or BC.
4. Flow Duration (FD): The time interval separating the first from the last packet in a flow is Flow Duration (FD).

Table 2. Feature extraction from sFlow data

Timestamp	PR (pps)	FE	BC (Bytes)	FD (ms)	Label (0 = Normal, 1 = DDoS)
10:02:30	150	0.7	3000	120	0
10:02:31	8000	0.9	60000	5	1
10:02:32	16000	0.95	120000	2	1

Normal traffic has moderate values; anomalous flows-which may be DDoS attacks-have a very high packet rate, a short flow duration, and great entropy, as shown in Table 2.

The Packet Rate (PR) is calculated as:

$$PR = \frac{N_p}{T_w}$$

Where:

N_p = Number of packets observed in the time window
T_w = Time window duration in seconds

A high PR value indicates suspicious activity, possibly part of a DDoS attack. The Flow Entropy (FE) measures the randomness of network traffic and is computed as:

$$FE = -\sum_{i=1}^{n} P\,(i)\,log_2\,P(i)$$

Where:
$P(i)$ = Probability of occurrence of Flow iii
n = Total Number of unique flows

Increasing the randomness in packet source-destination couples results in a higher entropy value, indicating the probability of a distributed denial of service attack. Effective capture of network traffic patterns by a sFlow-based traffic monitoring system reduces the overhead on SDN switches.

The system can effectively differentiate between DDoS and normal traffic by means of the extraction of basic properties, including packet rate, flow entropy, byte count, and flow duration. Using the acquired features as input, stacked deep ensemble models can identify distributed denial of service attacks with minimum of false positives.

5.3. Stacked Deep Ensemble Model for Classification

The proposed Stacked Deep Ensemble Model combines three deep learning architectures: ResNet, RNN, and DNN. This model seeks to improve classification accuracy for distributed denial of service attacks in SDN settings. Every single model adds specifically to the whole learning process:

1. DNN: The DNN allows one to detect high-level abstract patterns in the network traffic.

2. RNN: RNN searches for sequential dependencies in flow-based traffic.
3. ResNet: ResNet improves feature extraction and helps to eliminate problems with disappearing gradients.

Training starts with traffic data gathered feature-extracted from the sFlow collector. Prometheus then keeps this information as a time-series dataset. The dataset consists of two sets: the training and the testing ones. In independent training, every deep learning model uses extracted features, including packet rate, flow entropy, byte count, and flow duration.

Every model learns to depict the traffic patterns on the network differently. DNN uses dense layers to detect complicated relationships; RNN uses Long Short-Term Memory units (LSTMs) to recognize temporal dependencies; ResNet uses convolutional layers with residual connections to improve learning in deep neural networks.

Every model trained will produce a probability score between 0 and 1, indicating whether the Flow is an attack or normal. These specific forecasts are collected and investigated during the ensemble decision phase to produce the last classification. Table 3 shows model predictions prior to ensemble decisions.

Table 3. Model predictions before ensemble decision

Flow ID	DNN Output (0-1)	RNN Output (0-1)	ResNet Output (0-1)	Actual Label (0=Normal, 1=DDoS)
1	0.10	0.15	0.20	0
2	0.85	0.80	0.90	1
3	0.40	0.45	0.50	0
4	0.95	0.90	0.98	1

6. Ensemble Decision

An ensemble averaging approach helps one to reach the final classification decision. This approach estimates the attack likelihood using the combined DNN, RNN, and ResNet predictions.

The research finds the aggregated prediction score by means of a computation using the following equation:

$$P_{\text{final}} = \frac{P_{\text{DNN}}+P_{\text{RNN}}+P_{\text{ResNet}}}{3}$$

Where:

PDNN	= Output probability from DNN
PRNN	= Output probability from RNN
PResNet	= Output probability from ResNet

The traffic is said to be a distributed denial of service attack (1) if the last packet (Pfinal) outperforms a predefined threshold, for example, 0.5; otherwise, it is normal traffic (0).

Table 4. Final ensemble decision output

Flow ID	Aggregated Score Pfinal	Final Prediction (0=Normal, 1=DDoS)
1	$\frac{0.10+0.15+0.20}{3} = 0.153$	0 (Normal)
2	$\frac{0.85+0.80+0.90}{3} = 0.85$	1 (DDoS)
3	$\frac{0.40+0.45+0.50}{3} = 0.45$	0 (Normal)
4	$\frac{0.95+0.90+0.98}{3} = 0.94$	1 (DDoS)

Table 4 shows the ensemble decision output. A weighted ensemble approach allows us to make even more improved decisions. Every model adds different degrees of significance depending on its accuracy under this approach; the weighted prediction score can be calculated as follows:

$$P_w = w_1 P_{\text{DNN}} + w_2 P_{\text{RNN}} + w_3 P_{\text{ResNet}}$$

Where:

w1,w2 and w3 are the model-specific weights (determined based on validation accuracy).

P_w is compared against a threshold to make the final classification.

Using weighted contributions, models with higher performance levels guarantee a more significant impact on the decision, so enhancing the general process accuracy. The stacked Deep Ensemble Model improves distributed denial of service attack detection in SDN systems by leveraging several deep learning architectures.

By residual learning, ResNet enhances feature extraction; the DNN finds high-level patterns; the RNN finds attack behaviors dependent on time. Predictions are compiled during the ensemble decision phase by weighted approach or simple averaging. This keeps the minimum of false positives and ensures higher classification accuracy.

6.1. Detection Output and Mitigation Response

As soon as the Stacked Deep Ensemble Model detects that the traffic on the network is either normal or a distributed denial of service attack to safeguard the SDN from disturbance in service, the system starts a mitigating reaction. The predefined mitigating actions depend on the degree of attack once the detection output reaches the SDN Controller (ONOS).

6.2. Detection Output Process

Prometheus, a database with time series, is kept safe after the ONOS SDN Controller constantly monitors the detection results from the deep learning model. Should an anomaly arise, the system will categorize the attack based on metrics including packet rate, entropy, and flow length into Low, Medium, or High threat levels. Driving these threat levels is the system.

1. Low Threat (Minor Anomaly): Temporary traffic spikes are observed, but normal behavior resumes quickly. No action is taken.
2. Medium Threat (Potential Attack): Packet rates and entropy indicate suspicious behavior, but the attack is not fully confirmed. The system reduces bandwidth allocation for the suspicious IP.
3. High Threat (Confirmed DDoS Attack): Extremely high packet rates and abnormal flow patterns indicate a clear attack. The SDN controller blocks the malicious source IP and reroutes traffic to mitigate congestion.

As in Table 5, the detection results are kept in a database to be closely inspected and used in forensic investigations.

Table 5. Detection output

Timestamp	Source IP	Packet Rate (pps)	Flow Entropy	Attack Probability	Threat Level	Mitigation Action
10:02:30	192.168.1.2	150	0.7	0.15	Low	None
10:02:31	172.16.4.8	8000	0.9	0.85	High	Block IP
10:02:32	10.0.0.7	16000	0.95	0.94	High	Block IP
10:02:33	192.168.5.3	500	0.8	0.50	Medium	Limit Bandwidth

7. Mitigation Response Mechanism

The SDN Controller takes mitigating action upon the identification of a high-probability attack. The way the risk is lowered is by routing traffic, limiting the connection rate, or deleting packets arriving from hostile sources.

The responses are defined by the two equations below. When the traffic from a given source IP exceeds a predefined threshold (Pth), the rate limiter reduces the bandwidth connected to that source IP to prevent congestion on the network.

$$R_{\text{new}} = R_{\text{current}} \times \left(1 - \frac{P_{\text{attack}}}{P_{\max}}\right)$$

Where:

Re_{new}	=	Adjusted bandwidth rate for the suspected IP
$Re_{current}$	=	Current bandwidth allocated
P_{attack}	=	Attack probability detected by the model
P_{max}	=	Maximum probability (1.0 for confirmed attacks)

For a source IP with an attack probability of 0.85 and bandwidth allocated of 100 Mbps, the new rate would be 0.85, for example.

$$R_{\text{new}} = 100 \times (1 - 0.85) = 15$$

This reduces the attacker's bandwidth, compromising their ability to properly overwhelm the system.

The SDN controller will use a flow rule to guide OpenFlow switches to block the IP address should the traffic anomaly score (S_a) be higher than a blocking threshold (S_b).

$$\text{Block IP} \quad \text{if} \quad S_a > S_b$$

Where:

S_a = Weighted sum of packet rate, flow entropy, and byte count
S_b = Predefined threshold for blocking (e.g., 0.80)

The system automatically blocks the false IP address, as in Table 6.

The Detection Output and Mitigating Response system ensures real-time reactions to distributed denial of service attacks. The Stacked Deep Ensemble Model oversees correctly classifying network traffic, while the SDN Controller (ONOS) is in charge of dynamically changing bandwidth and blocking dangerous sources depending on the degree of different threats.

One can quickly prevent attackers from overwhelming the network through rate limiting and IP blocking equations, ensuring minimum disturbance of services.

Table 6. Post-mitigation actions

Timestamp	Source IP	Threat Level	Initial Bandwidth (Mbps)	New Bandwidth (Mbps)	Mitigation Action
10:02:30	192.168.1.2	Low	50	50	None
10:02:31	172.16.4.8	High	100	15	Block IP
10:02:32	10.0.0.7	High	100	10	Block IP
10:02:33	192.168.5.3	Medium	50	25	Limit Bandwidth

7.1. Results and Discussion

The experimental configuration was evaluated with help from a virtualized environment SDN testbed. Topology was done using GNS3, and ONOS was chosen as the software-defined networking controller. Real-time network traffic was collected with sFlow-RT; the data was kept in Prometheus for more inspection. TensorFlow, Keras, and Scikit-learn were used to build the Stacked Deep Ensemble Model (DNN, RNN, ResNet) and then trained on the NSL-KDD and UNSW-NB15 datasets. Another Python implementation was used. A dedicated server in the experimental setup had a 3.6 GHz Intel Xeon processor, 64 GB of RAM, and an NVIDIA RTX 3090 Graphics Processing Unit (GPU), accelerating deep learning computations. We assessed the proposed method against three existing methods: CNN-Based Intrusion Detection System (CNN-IDS), Random Forest Classifier (RF-IDS) and LSTM-Based Anomaly Detection (LSTM-IDS).

7.2. Dataset Description

Two well-known cybersecurity datasets were used:

1. NSL-KDD: The NSL-KDD dataset is a modified form of KDD99, including labeled attack traffic. Among these are DDoS, probe, U2R, R2L, and attacks.
2. UNSW-NB15: UNSW-NB15 shows regular traffic and several types of attacks, including exploits, fuzzers, and denial of service attacks.

Table 7. Dataset

Timestamp	Source IP	Destination IP	Protocol	Packet Rate	Flow Duration	Attack Type
10:02:30	192.168.1.2	10.0.0.5	TCP	500	2.5 sec	Normal
10:02:31	172.16.4.8	10.0.0.6	UDP	8000	1.2 sec	DDoS
10:02:32	10.0.0.7	10.0.0.9	ICMP	16000	0.8 sec	DDoS
10:02:33	192.168.5.3	10.0.0.10	TCP	500	2.1 sec	Probe

Table 8. Experimental setup and parameters

Parameter	Value
SDN Controller	ONOS
Simulation Tool	GNS3
Traffic Collector	sFlow-RT
Database	Prometheus
Deep Learning Framework	TensorFlow, Keras
Datasets Used	NSL-KDD, UNSW-NB15
Training Data Split	80% Training, 20% Testing
Batch Size	64
Learning Rate	0.001
Optimizer	Adam
Number of Epochs	50
GPU Used	NVIDIA RTX 3090
CPU Used	Intel Xeon (3.6 GHz)
RAM	64 GB

Table 9. Accuracy over 50 Epochs

Epochs	CNN-IDS	RF-IDS	LSTM-IDS	Proposed Model
10	82.1%	85.3%	87.2%	**90.5%**
20	84.5%	87.1%	89.5%	**92.8%**
30	86.3%	88.4%	91.2%	**94.6%**
40	87.8%	89.9%	92.8%	**96.2%**
50	88.9%	91.0%	94.1%	**97.5%**

Regarding accuracy, the proposed Stacked Deep Ensemble Model generated routinely better results than the previously used methods. Comparing it to 88.9% (CNN-IDS), 91.0% (RF-IDS), and 94.1% (LSTM-IDs) at 50 epochs, it reached 97.5% accuracy. This is a significant improvement. Time analysis and deep feature extraction were applied in the ensemble architecture to improve classification performance. Notably, higher than the other three models, the proposed model obtained an F1-score of 96.8% after 50 epochs against CNN-IDS (86.0%), RF-IDS (89.2%), and LSTM-IDS (92.5%). This development indicates that the model can correctly balance recall and accuracy, lowering false alarms and preserving high attack detection rates. Over 50 epochs, the proposed model exceeded CNN-IDS (86.8%), RF-IDS (89.7%), and LSTM-IDs (93.0%), with a precision of 97.2%. Real-world adaptive learning mechanisms of the ensemble structure and enhanced feature selection significantly reduced the false positive count, improving the dependability of detection of distributed denial of service attacks. The proposed model exhibited the best recall, indicating that real-time threat detection is strengthened and that the capacity to correctly identify attack traffic has developed, thus reducing the number of false negatives. The results revealed that the proposed Stacked Deep Ensemble Model exceeded the current methods. Its 98.3% training accuracy outperforms those of CNN-IDS (91.2%), RF-IDS (93.5%), and LSTM-IDS (95.1%).

Table 10. F1-Score over 50 Epochs

Epochs	CNN-IDS	RF-IDS	LSTM-IDS	Proposed Model
10	78.5%	81.8%	84.1%	**88.7%**
20	80.9%	84.3%	86.9%	**91.2%**
30	83.0%	86.1%	89.2%	**93.5%**
40	84.7%	87.9%	91.0%	**95.4%**
50	86.0%	89.2%	92.5%	**96.8%**

Table 11. Precision over 50 Epochs

Epochs	CNN-IDS	RF-IDS	LSTM-IDS	Proposed Model
10	80.2%	83.0%	85.7%	**89.9%**
20	82.5%	85.2%	88.1%	**92.3%**
30	84.1%	87.0%	90.0%	**94.1%**
40	85.5%	88.4%	91.6%	**95.9%**
50	86.8%	89.7%	93.0%	**97.2%**

Table 12. Recall over 50 Epochs

Epochs	CNN-IDS	RF-IDS	LSTM-IDS	Proposed Model
10	76.9%	80.7%	83.4%	**87.9%**
20	79.2%	83.1%	86.3%	**90.8%**
30	81.6%	85.3%	88.5%	**93.0%**
40	83.4%	87.1%	90.4%	**94.9%**
50	84.9%	88.5%	91.9%	**96.3%**

Table 13. Accuracy of training and testing data

Method	Training Accuracy	Testing Accuracy
CNN-IDS	91.2%	88.9%
RF-IDS	93.5%	91.0%
LSTM-IDS	95.1%	94.1%
Proposed Model	**98.3%**	**97.5%**

Table 14. F1-Score on training and testing data

Method	Training F1-Score	Testing F1-Score
CNN-IDS	87.4%	86.0%
RF-IDS	89.8%	89.2%
LSTM-IDS	92.5%	92.5%
Proposed Model	**97.1%**	**96.8%**

Table 15. Precision on training and testing data

Method	Training Precision	Testing Precision
CNN-IDS	88.7%	86.8%
RF-IDS	90.3%	89.7%
LSTM-IDS	93.1%	93.0%
Proposed Model	**97.5%**	**97.2%**

Table 16. Recall of training and testing data

Method	Training Recall	Testing Recall
CNN-IDS	86.1%	84.9%
RF-IDS	88.9%	88.5%
LSTM-IDS	91.7%	91.9%
Proposed Model	**96.5%**	**96.3%**

The proposed model shows a generalizing gap by achieving a higher degree of accuracy in testing, 97.5%. The F1-score, which came out at 96.8% on test data, offers still more evidence that the proposed model is robust. This relates to the score of 86.0% for CNN-IDS, 89.2% for RF-IDS, and 92.5% for LSTM-IDS. Clearly, there is a better balance between recall and accuracy that would generate less false classifications. Precision and recall values followed a similar trend as the proposed model got the best performance (97.2% accuracy and 96.3% recall on testing data).

The researchers arrived at this as their conclusion. This will raise attack detecting capability and help to reduce the false positive rate. Though generalizing performance across both the training and testing datasets is maintained high, the results show that deep ensemble learning is a useful technique for exactly identifying DDoS.

8. Conclusion

Real-time sFlow-based traffic monitoring in the proposed Stacked Deep Ensemble Model effectively detects Distributed Denial of Service (DDoS) attacks in SDN. Here, we combine DNN, RNN, and ResNet. With a 97.5% accuracy rate, a 96.8% F1-score, a 97.2% precision rate, and a 96.3% recall rate, the experimental results on testing data showed better performance than the previously used approaches.

The proposed method shows enhanced generalization and detection accuracy comparatively to CNN-IDS, RF-IDS, and LSTM-IDS. It also reduced the number of false positives and negatives the system produces. The model efficiently processes sampled network traffic using ONOS as the SDN controller and Prometheus as the time-series data storage, reducing the computational overhead while enabling real-time threat detection. Improved feature extraction is obtained by considering spatial and temporal changes in attack patterns in the deep ensemble architecture. The proposed approach, as the results reveal, precisely detects malicious traffic, ensuring network stability and resilience against DDoS hazards and increasing the SDN security of SDNs.

References

[1] Muruganantham Ponnusamy et al., "Design and Analysis of Text Document Clustering using Salp Swarm Algorithm," *The Journal of Supercomputing*, vol. 78, no. 14, pp. 16197-16213, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Imran et al., "A Topical Review on Machine Learning, Software Defined Networking, Internet of Things Applications: Research Limitations and Challenges," *Electronics*, vol. 10, no. 8, pp. 1-28, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Manivel Kandasamy et al., "QOS Design using Mmwave Backhaul Solution for Utilising Underutilised 5G Bandwidth in GHZ Transmission," *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, Coimbatore, India, pp. 1615-1620, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[4] N. Yuvaraj et al., *An Artificial Intelligence Based Sustainable Approaches-IoT Systems for Smart Cities*, AI Models for Blockchain-Based Intelligent Networks in IoT Systems, Springer, pp. 105-120, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Muhammad Afaq, Shafqat Ur Rehman, and Wang-Cheol Song, "A Framework for Classification and Visualization of Elephant Flows in SDN-Based Networks," *Procedia Computer Science*, vol. 65, pp. 672-681, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[6] Meng Wang, Yiqin Lu, and Jiancheng Qin Source-Based, "Source-Based Defense against DDoS Attacks in SDN Based on sFLOW and SOM," *IEEE Access*, vol. 10, pp. 2097-2116, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[7] Nadhir Fachrul Rozam, and Mardhani Riasetiawan, "XGBoost Classifier for DDOS Attack Detection in Software Defined Network using sFlow Protocol," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 13, no. 2, pp. 718-725, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[8] Adeniji Oluwashola David, and Oluwabusayo Israel Omotosho, "Scalable Flow Based Management Scheme in Software Define Network (SDN) Using sFlow," *WSEAS Transactions on Computers*, vol. 22, pp. 64-69, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[9] Neelam Gupta et al., "DDoS Attack Defence Mechanism Using sFlow," *2022 International Conference on Data Analytics for Business and Industry (ICDABI)*, Sakhir, Bahrain, pp. 302-308, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Afsaneh Banitalebi Dehkordi, MohammadReza SoltanaghaeI, and Farsad Zamani Boroujeni, "The DDoS Attacks Detection through Machine Learning and Statistical Methods in SDN," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2383-2415, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Mahmoud Said El Sayed et al., "A Flow-Based Anomaly Detection Approach with Feature Selection Method against Ddos Attacks in SDNs," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862-1880, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[12] Rochak Swami, Mayank Dav, and Virender Ranga, "Detection and Analysis of TCP-SYN DDoS Attack in Software-Defined Networking," *Wireless Personal Communications*, vol. 118, no. 4, pp. 2295-2317, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[13] Mohammad Javad Shayegan, and Amirreza Damghanian, "A Method for DDoS Attacks Prevention Using SDN and NFV," *IEEE Access*, vol. 12, pp. 108176-108184, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Rochak Swami, Mayank Dave, and Virender Ranga, "QR-based Approach for DDoS Detection and Mitigation in SDN," *Defence Technology*, vol. 25, pp. 76-87, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[15] Kun Jia et al., "A lightweight DDoS Detection Scheme under SDN Context," *Cybersecurity*, vol. 5, no. 1, pp. 1-15, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Naziya Aslam, Shashank Srivastava, and M.M. Gore, "ONOS Flood Defender: An Intelligent Approach to Mitigate DDoS Attack in SDN," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 9, 2022. [CrossRef] [Google Scholar] [Publisher Link]