JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 09 Issue 02
2025

*Review Article*
https://doi.org/10.56979/902/2025

# Evaluating Machine Learning -Based Intrusion Detection in Software Defined Networks Using NSL-KDD Dataset

**Khurram Zeeshan Haider[1], Qamas Gul Khan Safi[2], Muhammad Awais[1], Urooj Fatima[1], and Muhammad Munwar Iqbal[2]**

[1]Department of Software Engineering, Government College University, Faisalabad, Pakistan.
[2]Department of Computer Science, University of Engineering and Technology, Taxila, Pakistan.
Corresponding Author: Qamas Gul Khan Safi. Email: qamas.gul@uettaxila.edu.pk

**Received:** July 03, 2025 **Accepted:** August 15, 2025

**Abstract:** Software Defined network is an emerging and evolving network technology. A significant advantage of SDN is that it offers centralized control of the network, where all controller operations are centralized. The open-source Software Defined Networking (SDN) emulator, Mininet, has been utilized for generating and simulating virtual networks, in conjunction with POX, an open-source remote controller. A publicly available dataset, NSL KDD, is utilized for intrusion detection, evaluation, and comparison among several classification algorithms. In this proposed work, a supervised machine learning algorithm, a Decision Tree, is utilized for intrusion detection. Python is used as a tool to create, test, and compare different approaches to detect malicious attacks and identify the best one among them. These experiments are conducted to achieve results based on accuracy, recall, false positives, F-measure, and precision. Our proposed ML approach possesses more potential for intrusion detection using an accuracy measure. The decision tree is the best approach to detect intrusion based on detection speed and effectiveness. The precision of a decision tree is also the highest and most accurate among machine learning techniques.

**Keywords:** Software-Defined Networks (SDN); Machine Learning; Intrusion Detection; SVM; Decision Tree

## 1. Introduction

Software-defined networks have been widely adopted in various organizations over the past few years. In SDN, hardware-based control of network systems has been transformed into software-based control. The decoupling of the control plane and the forwarding plane provides ease of network programmability. SDN enables the control of networking behavior through software, eliminating the need for physical connectivity. It is centrally controlled, providing easy restoration, security, and bandwidth management [1]. Unlike traditional network architecture, SDN is a new technology with a dynamic nature. The primary communication interface between the control and data layers is OpenFlow. Any group of IP packets that have common properties is called a flow. One of the significant benefits of SDN is multivendor interoperability. At the same time, SDNs face some security threats as they are more prone to vulnerabilities [2]. An Intrusion Detection System is the most critical countermeasure for network security. It has two main categories: signature-based and Anomaly-based intrusion detection systems. The former checks incoming packets based on a signature database of previous attacks, while the latter is based on a baseline model and identifies the deviations.

The primary reason that puts SDN in the forefront is the emergence of new security threats that significantly impact this networking architecture. One of the attacks that is harmful to this kind of network

is DDoS. Distributed Denial of Service (DDoS) attacks can cause significant loss to SDN architecture, whether it involves controllers or OpenFlow switches, if not adequately secured; the system becomes highly vulnerable to attacks. According to CVNI, Cisco Visual Networking Index, a survey showed that about 20 million DDOS attacks could happen by 2020, three times more than in 2015, and they are getting stronger with multiple attack vectors. Studies have shown that SDN can experience significant delays in delivering messages and a high packet loss rate when a saturation attack compromises the OpenFlow medium. Many businesses, industries, and social media platforms are at risk, resulting in financial and reputational losses. Therefore, it's a pressing need of the hour to implement safety measures that detect these attacks and apply a suitable solution to enhance their security.

To support the ongoing use of SDN, various techniques and methods have been employed to detect and mitigate potential losses associated with these anomalies. An important technique utilized globally is the application of machine learning algorithms that can be very beneficial for attack detection and mitigation [3]. When we compare the solutions of attack detection in a network using ML and non-ML approaches, we find that machine learning algorithms prove themselves to be more accurate than other techniques. Many machine learning research pieces have been conducted in different domains, but there are only a few on SDN. So, there is still considerable potential for research in the area of protecting SDN from threats. Using machine learning in SDN, many factors may affect the efficiency of every approach. One of the significant factors is the selection of attributes for the training and testing datasets; the second is the selection of classification models, the domain, and many others. We can't say that some specific machine learning classifier can perform best in all domains.

In this research, we demonstrated an SDN environment with the POX controller and detected intrusion attacks using some algorithms and compared their performance with one another in terms of accuracy, speed of detection, recall, F-measure, and precision. The ML classifiers we use here are KNN, Decision Tree, Naïve Bayes, and Support Vector Machine.

## 2. Background and Literature Review

The SDN architecture, as compared with traditional networking architectures, decouples planes in the new architecture that expose new attack surfaces, thus providing a larger attack surface area. Attacks such as Denial of Service (DoS), Distributed Denial of Service (DDoS), IP spoofing, unauthorized access to various network elements, configuration issues, protocol flooding, and many other threats can harm its infrastructure if not adequately protected. Each layer of SDN is vulnerable to different types of attacks. In the application plane, the software is used in a way that consumes sources, generating fake transactions that make them unavailable to the legitimate users' transactions that make them unavailable to the legitimate users. One example of an application-layer attack is the HTTP flood. In a control plane attack, which can be devastating to the entire network, the controller is compromised, affecting its performance and availability. Hence, disturbs the communication with different components. When we discuss data-plane attacks, the first devices that come to mind are switches and routers, among others. These could be attacked by two options: first, by switches, and second, by the southbound API, where massive malicious traffic could be sent to attack the switches through new and unknown IPs.

A Denial of Service (DoS) occurs when an intruder consumes network resources with fake traffic, rendering them unavailable to legitimate users. A Distributed Denial of Service (DDoS) is a large-scale variant in which multiple compromised systems overwhelm a target host or network, often causing severe disruption. Figure 1 illustrates various threats, including DoS and DDoS, that can degrade network performance.

- Volumetric Attack
- Application Layer Attack
- Protocol Attack

The machine learning-based methods for intrusion detection in the network, both supervised and unsupervised algorithms, have been utilized to develop models. When the dataset is selected, different classifiers are used for feature selection for the optimal subset. A significant number of algorithms have been used over the years for classification and other real-world problems. For classification purposes, decision trees, KNN, support vector machines, naïve Bayes, logistic regression, and others are widely used. When these algorithms are used, keep in mind some measures that assure their performance based on

some specific tasks. As we are discussing intrusion detection here, the best algorithms to develop an intrusion detection system for a network are KNN, ANN, Decision Tree, etc. Many researchers have proposed their own NIDS based on one or more algorithms. In the methodology section, we provide a detailed discussion of the algorithms used. Every algorithm has its own characteristics for classification and prediction, and performs differently in various environments. We can't propose a single best-used algorithm for all situations and purposes.

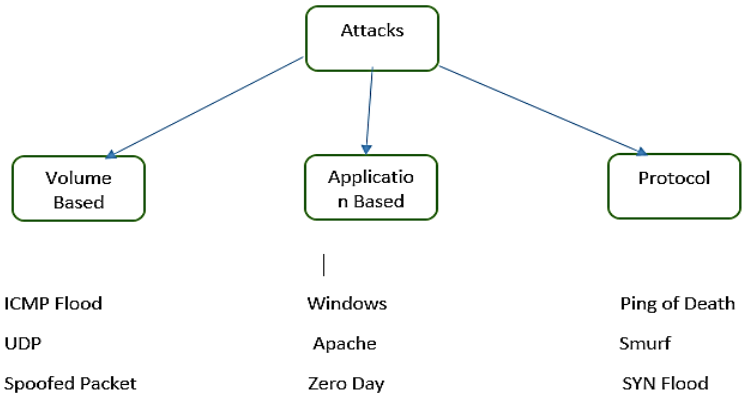


**Figure 1.** Categories of Attack

A well-known dataset, NSL KDD, which has been used for intrusion detection globally over the years, is also used in our experiment. It is the modified version of KDDCUP'99. It has four classes: basic, content, host, and traffic. The dataset we will be using in our experiments is NSL-KDD, which has 42 attributes. It is basically the modified version of KDD 99. A well-known dataset, NSL KDD that has been used for intrusion detection globally over the years is also used in our experiment. It is the modified version of KDDCUP'99. It has four classes: basic, content, host, and traffic. The dataset we will be using in our experiments is NSL-KDD, which has 42 attributes. It is basically the modified version of KDD 99. This dataset is labelled as two classes: Normal connection instance, and Attack/Anomaly instance.

SDN software-defined network anomalies are now detected with the help of machine learning algorithms. To detect anomalies like DDoS, Viruses and worm in SDN [4] used NIDS type of network the work is an excellent success with DL (Deep learning) yet challenges like self-intruding are cause of significant concern in contrary to it [5] developed IDs by using the classic ML mechanisms such as SVM, KNN, ANN and RF these used methods were successful in the regards yet limited to some extent. Work contributed by [6] to create tools for user assistance in developing machine learning (ML). The results were helpful for configurative and testing frameworks from a perspective. Another progress in the model of NIDS with the help of DL (Deep learning) algorithms by [7], although the results were not good at the commercial level, it still has positive potential to be applied in the future. A variety of ML techniques, such as KNN, Naive Bayes [3], and SVM, were used by the attacker, including DD. The paper clearly classified and distinguished among focused documents on the machine learning (ML) technique of neural networks by [8], tests the IDS model on several attacks, including DDoS, U2R, and R2L, and presents the results for R2L. The results were as expected and are helpful for many purposes. Some machine learning (ML) techniques, such as J-48, SVM, KNN, and RF algorithms, are employed by [9] to address various types of DDoS attacks. The J-48 decision tree algorithm is found to be the most suitable for the devised model and is expected to remain helpful in the future.

Recent studies have placed considerable emphasis on developing robust intrusion detection systems (IDS) tailored for Software Defined Networks (SDN), leveraging machine learning and deep learning methodologies to address evolving cyber threats. [10] presented a comprehensive survey in IEEE Access that categorized IDS approaches for SDN into machine learning, deep learning, and hybrid paradigms. Their review underscored the critical need for realistic, SDN-specific datasets, such as those found in SDN, while also highlighting the limitations of conventional datasets like NSL-KDD and UNSW-NB15, which often fail to capture the dynamic and programmable nature of SDN. The survey concluded that cross-domain generalization and scalability remain significant challenges, motivating further research into adaptive models capable of handling heterogeneous SDN environments.

Building on this foundation, [11] compared Transformer-based models with CNN-LSTM architectures for intrusion detection in SDN. Using the In SDN dataset, they demonstrated that Transformer-based deep models achieved near-perfect accuracy (~99.4%), especially when combined with feature reduction techniques. Their work highlighted the trade-off between accuracy and computational efficiency, suggesting that while Transformer architectures offer state-of-the-art performance, their deployment in real-time SDN environments requires further optimization. Similarly, [12] proposed ML-IDSDN, a machine learning-based IDS tested on synthetic SDN datasets generated through Mininet and Ryu. Their evaluation showed that Random Forest classifiers achieved up to 97% accuracy with relatively low computational overhead, proving that lightweight ML methods can still deliver competitive results for SDN-based security monitoring [13].

More recently, researchers have focused on hybrid deep learning strategies to improve the detection of complex attacks such as DDoS in SDNs. In a research paper [14] proposed a CNN–GRU hybrid model for DDoS detection, demonstrating superior performance over standalone models with accuracies above 98%, particularly excelling in recall. Extending this direction, [15] introduced a meta-parameter optimized CNN-BiGRU with attention mechanisms, also targeting DDoS attack detection in SDN. Their model reported an accuracy of ~99.5%, demonstrating that meta-optimization techniques significantly enhance convergence stability and robustness. However, both works primarily focused on DDoS detection, leaving broader categories of intrusion underexplored. Collectively, these studies indicate a promising trajectory toward highly accurate IDS in SDNs, but also reveal critical gaps in dataset diversity, real-time deployment feasibility, and attack-type generalization that future research must address.

Transformers capture long-range traffic relationships with near-perfect precision, but their computing cost limits real-time deployment. By combining spatial and temporal learning, hybrid models like CNN–GRU or CNN–BiGRU with attention enhance the detection of complex threats; nevertheless, they are still limited in scope, frequently concentrating on a single incursion type. This is addressed by adaptive machine learning, which provides flexibility where static frameworks fail by dynamically updating models to manage changing traffic and zero-day threats [16, 17]. They struggle with adversarial robustness, scalability, and stability, though. All things considered, these methods improve SDN security, but more effort is needed to strike a balance between precision, effectiveness, and generality in large-scale settings.

Elsayed et al [18] carried out some results, irrespective of the algorithm used, with an IDS developed system on DDoS and botnet attacks. This gives a positive result on mitigation mirrors and can be effective at any level (Commercial and Domestic). [19] have used the algorithms SVM, KNN, and NB against a variety of attacks such as DDoS, IP spoofing, UDP, and SYN flooding attacks. The current work is supervised in a single-controller SDN environment. Over 20 different data sets were used, and the results were full of ups and downs [20]. It's hard to say any single statement about how the selected technique acts against the generated attack, which gives a lead to future work that uses unsupervised ML algorithms and multiple controllers. A work for SDN based on DL (Deep learning) and ML (machine learning) algorithms, as focused on by [21], utilized switches such as POX and NOX, which are based on Python and C++, respectively. There was a significant difference between the two approaches used, and in the near future, they planned to implement their proposed model in a real-life network and traffic. HMIPv6 is a mobility management protocol designed to optimize handover performance in IPv6 networks [22]. Researchers have extensively explored both supervised and unsupervised machine learning and data mining techniques to enhance anomaly detection capabilities. [23, 24].

A study carried out by [25] on the IOT networks by deploying the ML (machine learning) algorithms is of great significance, where they worked regardless of the architecture they used. The outcomes of deploying all the ML techniques show that the regression tree models are the most suitable of all. A Middle East researcher [26] also proposed that the IDs architecture follows the ML (machine learning) algorithms such as RF, Artificial neural network, SVM, and K-nearest neighbor. The average calculated accuracy was around 76% which cannot be considered an encouraging result. A flaw was encountered in the architecture where ML does not collaborate positively. SOM is an unsupervised ML (machine learning) algorithm tested on SDN architecture to detect all DDoS attacks by [27] and Surprisingly the detection rate was up to 98% which was very high for such a setup, Where other ML algorithms such SVM, RF, J-48, where not even close the readings taken by SOM algorithms in the series of experiment, Such results were carried out in

the presence of NOX controller (C++ based controllers) and it is believed that POX (Python based controllers) are not so much accurate for the respective setup. MADMAS, an architecture for SDN, is being tested by [28] with the ML (machine learning) algorithms such as SOM, M-SOM, LVQ1, M-LVQ1 and H-LVQ1 on the generated traffic with attacks such as DDoS, U2R, R2L and different kinds of probes, The advantage of using MADMAS architecture is being showed by experimentation that controllers may remain unsupervised and still the architecture would manage to detect the malicious activities around, The future works include the detailed studies on H-LVQ1 because they believe that it would be the key to success with such an architecture to detect the saturated attacks. A series of experiments and a bunch of results were carried out using KDDCup-99 and NSL-KDD datasets with ML (machine learning) techniques by Elsayed. The results concluded were of average type, like they are between 75% and 80% accurate [29]. Still, they believe that with such a setup, room for improvement exists, so their future work includes testing on the KDDCup-99 dataset using a DL (Deep learning) algorithm. A detailed concept of NFV-enabled cloud over a 5G network is presented in a journal [30]. In this very journal, they used AI to detect a possible malicious attack in a multi-layered cloud-enabled 5G network. The results were quite disturbing because the used technology hardware does not fully coordinate with the developed algorithm and the applied technique, yet in the future, they will carry out the result with advanced hardware with a processor frequency greater than 5 GHz.

**Table 1.** Summary and Critical Evaluation of ML/DL-Based SDN Intrusion Detection

| Citation(s) | Approach & Contributions | Critical Evaluation |
|---|---|---|
| [4]–[9] | Early ML/DL IDS (SVM, KNN, ANN, RF, neural networks, decision trees) for SDN anomaly detection | Limited scalability, real-time applicability, and generalization |
| [10]–[16] | Hybrid and advanced DL models (Transformers, CNN-LSTM, CNN–GRU, CNN-BiGRU); evaluated on multiple datasets | High accuracy for specific threats; broader coverage and real-time deployment remain challenges |
| [17]–[20] | ML/DL and unsupervised models (SOM, MADMAS) on different controllers (POX, NOX) | Controller-dependent performance; high-volume/multi-controller setups need optimization |
| [21]–[22] | Dataset-based evaluations (KDDCup-99, NSL-KDD) and AI for NFV-enabled 5G networks | Dataset realism and hardware constraints limit real-world applicability. |

Table 1 includes key ML and DL approaches used for SDN intrusion detection, summarizing their main contributions and focus areas. It also provides a critical evaluation, emphasizing limitations in scalability, real-time applicability, and threat coverage.

## 3. Methodology

In this section, we talk about the methodology and simulation environment. For the experimental setup, we install Ubuntu in VirtualBox. After that, Miniedit in Mininet and Wireshark are installed. Some configurations are made to make sure the setup will work fine; after that, the relevant commands of the topology creation can be seen in the Mininet CLI. And also set up the remote controller in the controller settings of the Miniedit. In a new SSH session, Wireshark is opened so that packet flow can be visible. In another SSH session, some commands are written to make POX the remote controller. When the 'Run' button is clicked in the Miniedit, the topology structure can be seen in the Mininet Environment also.

The Topology that is created can be seen in Figure 2. The experimental setup was created using a centralized controller to manage several switches, each of which is connected to an independent host. This layered structure reflects the separation of the control and data planes, with the controller handling decision-making while switching forward traffic between end hosts. An IDS module based on ML/DL approaches was also integrated to receive mirrored traffic from the switch to maintain real-time analysis together with the data plane. This arrangement allows for testing intrusion detection within a realistic yet controlled environment, enabling assessment of scalability, detection accuracy, and the interaction between centralized control and distributed traffic monitoring. This star topology is considered a campus management system, shown in Table 2.

In the experimental Topology, the remote controller is used, which is POX, and is connected with eight switches, and all eight switches are connected with eight separate hosts. Switches are associated with one another also. Below is Figure 3 for regular traffic and packet transmission between hosts. This indicates the packets flow after a gap of a few seconds. This is the IO Graph from Wireshark.

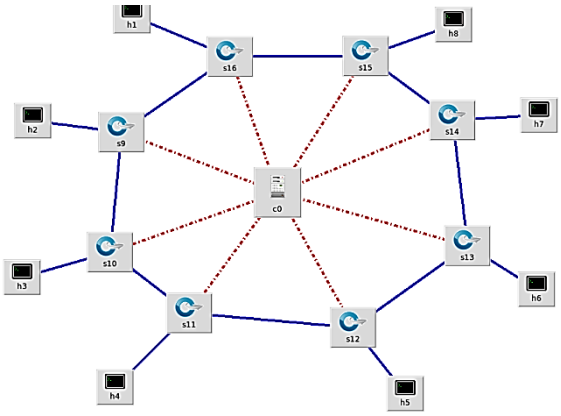


**Figure 2.** Miniedit Topology in Mininet

**Table 2.** Description of Topology

| Description | Value |
|---|---|
| No. of Controllers | 1 |
| No. of Switches | 8 |
| No. of Hosts | 8 |
| Topology | Star/Campus |

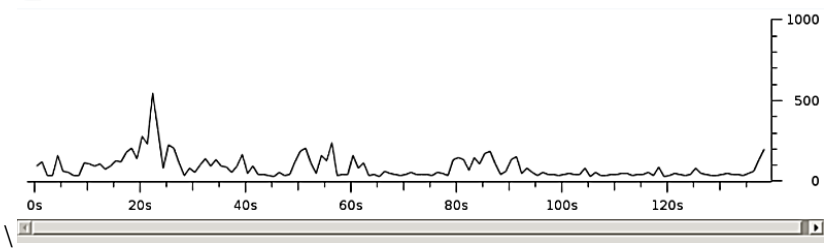


**Figure 3.** Wireshark IO Graph

```
64 bytes from 10.0.0.6: icmp_seq=200 ttl=64 time=0.071 ms
64 bytes from 10.0.0.6: icmp_seq=201 ttl=64 time=0.048 ms
64 bytes from 10.0.0.6: icmp_seq=202 ttl=64 time=0.091 ms
64 bytes from 10.0.0.6: icmp_seq=203 ttl=64 time=0.089 ms
64 bytes from 10.0.0.6: icmp_seq=204 ttl=64 time=0.074 ms
64 bytes from 10.0.0.6: icmp_seq=205 ttl=64 time=0.080 ms
64 bytes from 10.0.0.6: icmp_seq=206 ttl=64 time=0.429 ms
64 bytes from 10.0.0.6: icmp_seq=207 ttl=64 time=0.085 ms
^C
--- 10.0.0.6 ping statistics ---
207 packets transmitted, 207 received, 0% packet loss, time 206045ms
rtt min/avg/max/mdev = 0.037/2.628/87.781/13.279 ms
mininet>
```

**Figure 4.** Interface in Mininet

Figure 4 shows the interface of Mininet for the normal flow of traffic. SDN Network creation and adding POX Controller have been completed. In the POX library, the l3_learning module was modified, and ML was used. A training set of NSL KDD was used to train the model.

3.1. Selected Features

We selected six features from the NSL KDD dataset in light of previous research based on their characteristics relevant to SDN. Below is the subset of features that was chosen from the NSL KDD dataset, and these features were traffic-based and belong to an introductory class.

**Table 3.** Selected Features of NSL KDD

| Feature | Description |
|---|---|
| flag | Status of the connection |

| | |
|---|---|
| srv count | Request for the same services from a number of connections |
| source bytes | Number of bytes from the sender in a single connection |
| protocol type | Protocol type to make a connection |
| destination bytes | Number of bytes received in a single connection |
| duration | The time duration of making the network connection |

The selected features (flag, srv_count, source_bytes, protocol_type, destination_bytes, and duration) were prioritized as they represent fundamental traffic characteristics that significantly positively influence anomaly detection in SDN [1]. After training the model, we used the same setup along with ML Algorithms using the Testing dataset, having more than 22500 instances with both attack and regular classes. The controller then shows both the good and bad traffic.

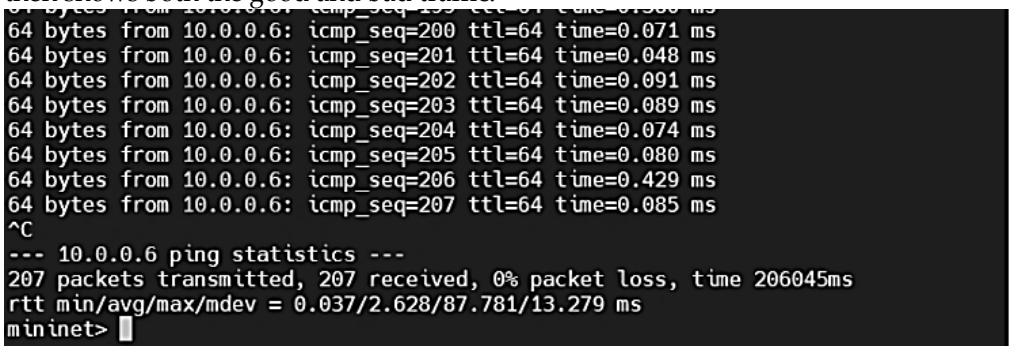


**Figure 5.** Successful Normal Flow in Mininet

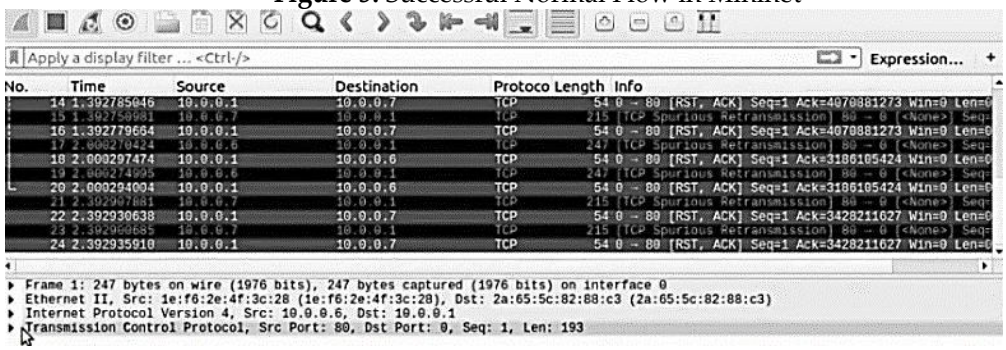


**Figure 6.** Attack Traffic

The regular and attack traffic can be seen in Figure 7.

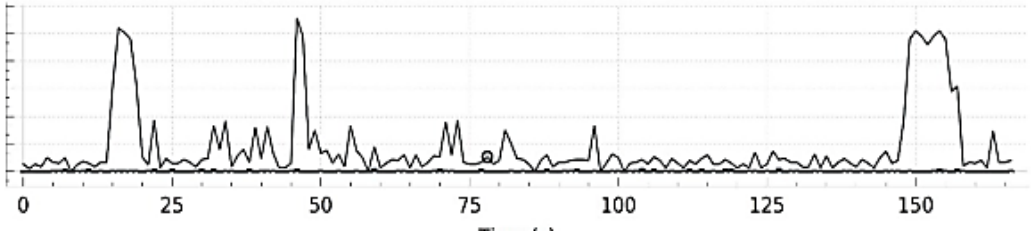


**Figure 7.** IO Graph of Normal and Attack Traffic

## 4. Results and Discussion

### 4.1. Performance of ML Algorithms

The performance of algorithms can be measured with a confusion matrix. It consists of a matrix that has n classes, the same number of rows, and columns. In this research, we have two classes, so the matrix will be of 2 x 2, one is normal and the other is an attack.

- True positive: Number of times the malicious traffic is correctly observed
- False positive: Number of times the regular traffic is incorrectly observed
- True negative: Number of times the regular traffic is correctly observed
- False negative: Number of times the malicious traffic is incorrectly observed

Precision (P) is calculated by dividing the total number of true positives by the sum of true positives and false positives. Precision is the result of the system making correct conclusions.

The evaluation parameters are Precision, Recall, Accuracy, and F1 Score. The formulas for the respective parameters are calculated by Equations (1), (2), (3), and (4) as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision} \quad (4)$$

Where TP, FP, TN, and FN indicate the true positive, false positive, true negative, and false negative values.

4.2. Results

The final results of the experiments are shown in Table 3.

The results of precision, recall, F-measure, and build time of each algorithm are given below.

**Table 4.** Algorithms with Measures

| ML Algorithms | Accuracy % | F measure | Precision | Recall | Time |
|---|---|---|---|---|---|
| Naïve Bayes | 77.14% | 1.19/ 0.001 | 1.72 | 91.5% | 2.21 sec |
| KNN | 76.57% | 1.2/0.012 | 1.79 | 97.62% | 21.25 sec |
| Decision Tree | 79.75% | 1.3/0.013 | 2.0 | 97.03% | 17.04 sec |
| SVM | 78.14% | 1.30/0.013 | 1.93 | 98.07% | 872.8 sec |

The bar graphs shown below describe the variation of measures of ML approaches that have been used for experiments. For the accuracy comparison of different algorithms, in terms of accuracy, the Decision Tree comes at the top.

4.3. Graphical Representation of Results

The graph shown in Figure 8 describes the accuracy of four selected algorithms. So, experiments show that the decision tree is highly accurate compared to other algorithms. The accuracy of the decision tree is 79.75%. SVM comes second with 78.14% accuracy. The third and fourth are Naïve Bayes and KNN with 77.14% and 76.57% accuracy, respectively.

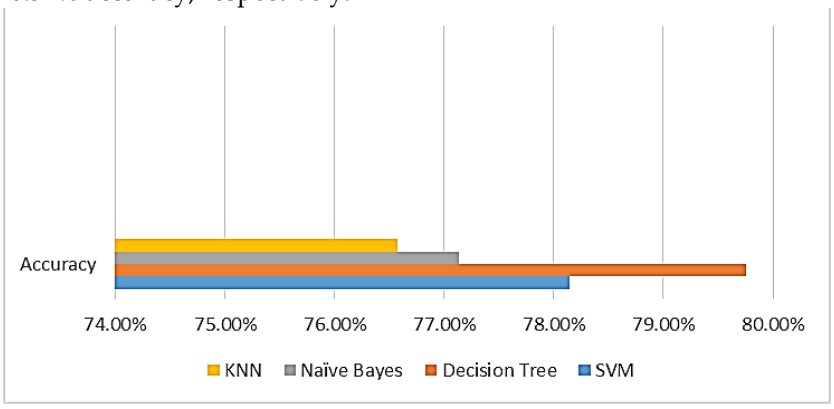


**Figure 8.** Accuracy Comparison

Precision is another factor that is evaluated in our experiments. And again, the decision tree saves first place as compared to other algorithms. Below is the Bar chart of the comparison.

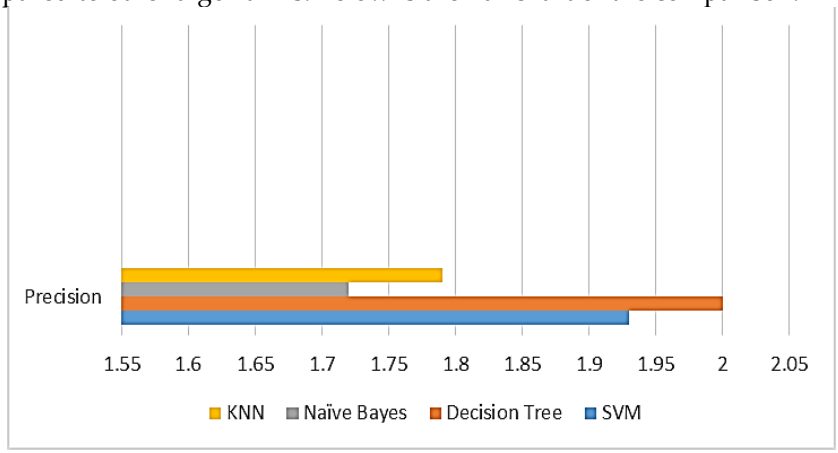

**Figure 9.** Precision Comparison

The above Figure 9 shows the precision of the used algorithms. Here, it can be seen that Naïve Bayes gives the lowest precision, which is 1.72. KNN bears the precision of 1.79. Support Vector Machine comes second with a precision of 1.93. The highest precision is of the decision tree that has 2.0 precision.

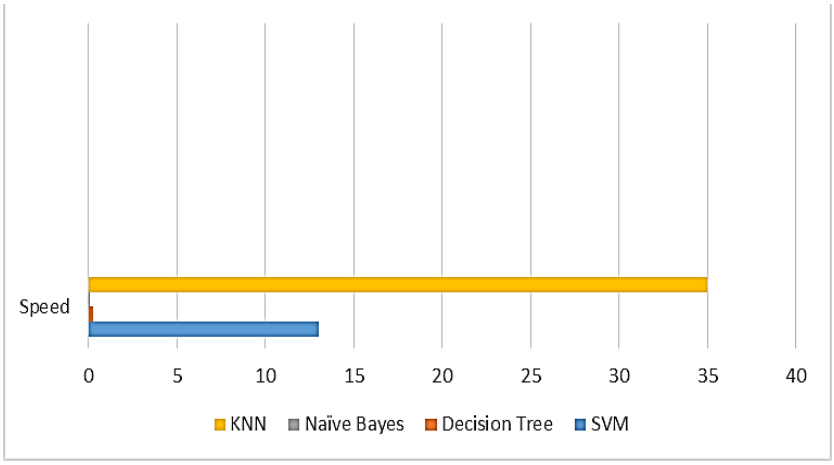


**Figure 10.** Speed Comparison

The bar chart shown in Figure 10 depicts the speed of the detection of different algorithms. But the results are astonishing. Naïve Bayes can detect the attack in 0.03 minutes. The decision tree also performs well when it comes to the detection of an attack; its detection speed is 0.28 minutes. The remaining two algorithms, SVM and KNN, do not give a reasonable time of detection and can be devastating for the network. SVM took almost 13 minutes to process and detect the attack, while KNN took 35 minutes to predict the attack, which is so unbearable for the network security.

**Table 5.** Comparative analysis of ML algorithms highlighting behavioral tendencies, dataset suitability, and computational trade-offs.

| Algorithm | Behavior | Dataset Fit | Trade-offs |
|---|---|---|---|
| Naïve Bayes | Simple, assumes independence; weak F-measure. | Best for high-dimensional sparse data. | Very fast; low accuracy ceiling. |
| KNN | Local similarity: high recall, low precision. | Works with complex boundaries; sensitive to noise. | Slow with large data (distance calc). |
| Decision Tree | Captures non-linear patterns; balanced metrics. | Good for mixed/tabular data. | Moderate training; risk of overfitting. |
| SVM | Strong margins; highest recall. | Suits small/medium, well-separated data. | Very slow; poor scalability. |

Predictive behavior of ML algorithms, along with dataset suitability and trade-offs in terms of efficiency based on the experimental results, have been summarized in Table 5.

In a network environment, SDN provides the versatility that makes it easy to tackle the challenges of an inherited network. Decoupling the data plane with a control plane makes the network logically centralized and provides the ability to quickly modify and more easily improve the architecture and its implementation. SDN is agile. The applications of ML in almost every field are not astonishing for us, and when we apply this in networking architecture, its results are really commendable. The use of ML algorithms in Software Defined Networking has increased over the past few years, with various new aspects. In the intrusion dataset, if more features are selected, then it takes more energy and time for the ML algorithm to create the model [31]. Some features are difficult to distinguish between an attack and regular traffic. To distinguish attack traffic from normal traffic, we tried to select the most significant features in our work.

Another thing that counts is the selection of the dataset. Many datasets have been used for intrusion detection; some are publicly available, and some are generated through libraries and software. Most of the

datasets that are used by researchers are outdated, but for a good reference and for model generation, we can use them. We used the NSL KDD dataset that is publicly available for our experiments to train and test the model; at the same time, a similar form of traffic flow was also produced using Hping3 in Ubuntu Server. Real-time model training is the priority. The NSL KDD dataset was used to train the model using some algorithms.

Decision Tree outperforms SVM and KNN because it can naturally handle both categorical and numerical features, NSL KDD dataset features in this case, and capture non-linear relationships without extensive parameter tuning. In contrast, SVM relies on kernel optimization, and KNN is sensitive to noise and high-dimensional data, making them less effective for diverse and complex traffic patterns in intrusion detection.

The selection of algorithms also plays an important part. There are many algorithms that people have used, but here in our work, I chose to use SVM, Decision Tree, Naïve Bayes, and KNN. Another important thing is the controller. Many controllers provide a number of options to users. In our thesis work, we used the POX controller because it's easy to use, and our basic simulator, Mininet, is also built in Python. It was beneficial to use Python to apply ML algorithms following the POX controller's learning module, providing the ability to learn and identify traffic more easily. Our focus is on the detection of malicious activity rather than developing a prevention system, although ML is also used to create a prevention system. While talking about intrusion detection here, it basically means to use ML to show how effectively it can detect malicious activity in the network, while creating an intrusion prevention system means to avoid the attack and try to make a system that can reduce the effects or consequences. The ML algorithms perform better than the algorithms that are not ML. The purpose of using the Mininet environment for intrusion attack detection is to work on the evolving technology, not on the conventional one, which will not be used in the future for several decades.

## 5. Conclusion and Future Work

Software-defined networking has an architecture that is cost-effective, user-specific, and easy to use as compared to conventional networking architecture. When we talk about the separation of its forwarding plane and control plane, threats are also present, seeing more surface areas. There are also more security challenges because of a centrally managed and programmable control system. In this thesis work, this emerging network was selected to perform experiments. Mininet was used to simulate this architecture. The POX controller was used to control all operations. There are many other controllers with a vast number of options that developers can use. For ease of use, we used the POX controller. Miniedit was used for the simulation to make a custom topology and enabled CLI with Mininet with the remote controller. To make a model based on ML approaches, the NSL KDD dataset, which has more than 125000 instances and 42 attributes, was used. Feature selection of this dataset was done based on the characteristics of the SDN, and six basic features were chosen. Wireshark was used to monitor the flow. Wireshark IO graphs show the difference between regular traffic and attack traffic. For the training model, we utilized Python. The evaluation was made based on accuracy, recall, F-measure, speed of detection, and precision.

**Funding:** No external funding has been received for this research.

**References**


1. D. Han, H. Li, X. Fu, and S. Zhou, "Traffic Feature Selection and Distributed Denial of Service Attack Detection in Software-Defined Networks Based on Machine Learning," Sensors, vol. 24, no. 13, pp. 4344, 2024.
2. J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in SDN: A comprehensive survey," Journal of Network and Computer Applications, vol. 159, pp. 102595, 2020.
3. J. Ashraf, and S. Latif, "Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques," in National Software Engineering Conference, 2014.
4. R. Sultana, I. Chlamtac, X. Peng, and R. J. C. N. Alhadad, "Deep learning based NIDS for anomaly detection in SDN," vol. 162, pp. 106861, 2019.
5. F. Tang, S. Zaidi, D. MacLaren, L. J. J. o. I. S. Mhamdi, and Applications, "Intrusion detection in SDN using classical ML algorithms," vol. 54, pp. 102562, 2020.
6. B. Suba, J. Bautista, E. Ledesma, and E. Yu, "User-assistance tools for ML-based IDSec configuration and testing," in International Conference on Information Technology (ICIT), 2019.
7. F. Tang, N. Mohandi, D. McLornon, S. Zaidi, and M. J. I. A. Ghogho, "Deep learning approaches for network intrusion detection in SDN," vol. 9, pp. 69855-69865, 2021.
8. A. Abubakar, and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in Seventh International Conference on Emerging Security Technologies (EST), 2017, pp. 138-143.
9. M. A. Rahman, M. H. Quraishi, and C. H. Lung, "A comparative study of machine learning algorithms for DDoS attack detection in SDN," in IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), 2019.
10. A. H. Janabi, T. Kanakis, and M. J. I. A. Johnson, "Survey: Intrusion Detection System in Software-Defined Networking," vol. 12, pp. 123456-123478, 2024.
11. M. S. Ataa, E. E. Sanad, and R. A. J. S. R. El-khoribi, "Intrusion detection in software defined network using deep learning approaches," vol. 14, no. 2221, 2024.
12. A. O. Alzahrani, M. J. F. J. C. Alenazi, C. Practice, and Experience, "ML-IDSDN: Machine learning-based intrusion detection system for software-defined network," vol. 35, no. 12, pp. e12345, 2023.
13. Q. G. K. Safi, S. Luo, C. Wei, L. Pan, and G. Yan, "Cloud-based security and privacy-aware information dissemination over ubiquitous VANETs," Computer standards & interfaces, vol. 56, pp. 107-115, 2018.
14. A. M. Elshewey, J. S. Reports, and et al., "DDoS classification of network traffic in SDN using a hybrid CNN–GRU model," vol. 15, no. 3012, 2025.
15. C. L. Kumar, J. S. Reports, and et al., "Metaparameter optimized hybrid deep learning model for next-generation cybersecurity in SDN," vol. 15, no. 4123, 2025.
16. M. A. Ahsan, M. M. Iqbal, H. Akbar, S. Ramzan, H. B. U. Z. Khan, U. Khadam, and M. U. Chaudhry, "Dynamic Malware Detection in Wireless Networks using Deep Learning," Journal of Computing & Biomedical Informatics, vol. 8, no. 01, 2024.
17. D. Kilichev, D. Turimov, and W. Kim, "Next–generation intrusion detection for iot evcs: Integrating cnn, lstm, and gru models," Mathematics, vol. 12, no. 4, pp. 571, 2024.
18. M. Manso, J. Moura, and C. Serrao, "Intrusion detection system in SDN against DDoS and botnet attacks," in International Conference on Cyber Security and Protection of Digital Services (Cyber Security), 2019.
19. S. Khamaiseh, E. Serra, Z. Li, and D. Xu, "Detecting saturation attacks in SDN via machine learning," in International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2019.
20. W. Liu, S. Luo, Y. Liu, L. Pan, and Q. G. K. Safi, "A kernel stack protection model against attacks from kernel execution units," Computers & Security, vol. 72, pp. 96-106, 2018.
21. R. Kumar, and M. J. I. J. o. C. A. Rehman, "Machine learning and deep learning based intrusion detection for SDN using POX and NOX controllers," vol. 182, no. 44, pp. 10-15, 2018.
22. A. Ahmed, S. Jabbar, M. M. Iqbal, M. Ibrar, A. Erbad, and H. Song, "An efficient hierarchical mobile ipv6 group-based bu scheme for mobile nodes in iot network," IEEE Internet of Things Journal, vol. 10, no. 10, pp. 8684-8695, 2022.
23. S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," IEEE access, vol. 6, pp. 48231-48246, 2018.
24. M. M. Iqbal, A. Ahmed, and U. Khadam, "Sinkhole attack in multi-sink paradigm: Detection and performance evaluation in RPL based IoT." pp. 1-5.
25. P. Amangele, M. J. Reed, M. Al-Naday, N. Thomos, and M. Nowak, "Hierarchical machine learning for IoT anomaly detection in SDN," in International Conference on Information Technologies (InfoTech), 2019.

26. M. A. J. S. Albahar, and C. Networks, “Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments,” 2019.
27. J. Santos, A. Souza, R. Santo, P. Ribeiro, and J. Moreno, “Detecting DDoS attacks in SDN with self-organizing maps,” in 8th International Conference on Information Systems and Computer Science (INCISCOS), 2019.
28. D. Jankowski, M. J. I. J. o. E. Amanowicz, and Telecommunications, “On efficiency of selected machine learning algorithms for intrusion detection in software defined networks,” vol. 62, no. 3, pp. 247-252, 2016.
29. M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. J. a. p. Jurcut, Machine-learning techniques for detecting attacks in SDN, 2019.
30. I. H. Abdulqadder, S. Zhou, D. Zou, I. T. Aziz, and S. M. A. J. C. N. Akber, “Multi-layered intrusion detection and prevention in the SDN/NFV enabled cloud of 5G networks using AI-based defense mechanisms,” vol. 179, pp. 107364, 2020.
31. S. K. Dey, M. R. Uddin, and M. M. Rahman, "Performance analysis of SDN-based intrusion detection model with feature selection approach." pp. 483-494.