technologies

Article

An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers

James Dzisi Gadze, Akua Acheampomaa Bamfo-Asante, Justice Owusu Agyemang *, Henry Nunoo-Mensah and Kwasi Adu-Boahen Opare

Faculty of Electrical and Computer Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana; jdgadze@gmail.com (J.D.G.); aquisante10@yahoo.com (A.A.B.-A.); hnunoo-mensah@knust.edu.gh (H.N.-M.); opare@knust.edu.gh (K.A.-B.O.)

* Correspondence: justiceowusuagyemang@gmail.com

Abstract: Software-Defined Networking (SDN) is a new paradigm that revolutionizes the idea of a software-driven network through the separation of control and data planes. It addresses the problems of traditional network architecture. Nevertheless, this brilliant architecture is exposed to several security threats, e.g., the distributed denial of service (DDoS) attack, which is hard to contain in such software-based networks. The concept of a centralized controller in SDN makes it a single point of attack as well as a single point of failure. In this paper, deep learning-based models, long-short term memory (LSTM) and convolutional neural network (CNN), are investigated. It illustrates their possibility and efficiency in being used in detecting and mitigating DDoS attack. The paper focuses on TCP, UDP, and ICMP flood attacks that target the controller. The performance of the models was evaluated based on the accuracy, recall, and true negative rate. We compared the performance of the deep learning models with classical machine learning models. We further provide details on the time taken to detect and mitigate the attack. Our results show that RNN LSTM is a viable deep learning algorithm that can be applied in the detection and mitigation of DDoS in the SDN controller. Our proposed model produced an accuracy of 89.63%, which outperformed linear-based models such as SVM (86.85%) and Naive Bayes (82.61%). Although KNN, which is a linear-based model, outperformed our proposed model (achieving an accuracy of 99.4%), our proposed model provides a good trade-off between precision and recall, which makes it suitable for DDoS classification. In addition, it was realized that the split ratio of the training and testing datasets can give different results in the performance of a deep learning algorithm used in a specific work. The model achieved the best performance when a split of 70/30 was used in comparison to 80/20 and 60/40 split ratios.

Keywords: SDN; DDoS; machine learning; deep learning

check for updates

Citation: Gadze, J.D.; Bamfo-Asante, A.A.; Agyemang, J.O.; Nunoo- Mensah, H.; Opare, K.A.-B. An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers. *Technologies* **2021**, *9*, 14. https://doi.org/10.3390/technologies9010014

Received: 17 December 2020
Accepted: 20 January 2021
Published: 11 February 2021

1. Introduction

With the current surge in the number of devices with networking capabilities, complex management strategies are required to provide a good quality of service (QoS). Achieving a good QoS becomes a hurdle in current traditional networks due to the vertical integration of the control and data planes. Furthermore, network optimization becomes difficult due to a high dependence on vendor-specific hardware and software.

Software-Defined Networking (SDN) is a new paradigm that solves the issues existing in traditional Internet architectures. It provides flexibility in management by making the networking programmable from a logically centralized control point. SDN decouples the control plane from the data plane present in traditional networks and deploys it in a remote device called the *controller* or *control layer*, as shown in Figure 1. It comes with the benefits of the centralized control functionalities, applications running on the network operating system, the unique capture of the global view of the architecture, the public interface of the north and south bounds, and its dynamic programmability in forwarding packets. Devices in the data plane, such as switches, forward the packets according to the control decisions or rules sent from the controller. The controller communicates with the application layer through the northbound application programming interface (API) and communicates with the data plane through the southbound API. Controller–switch communication is carried out using the OpenFlow protocol [1].

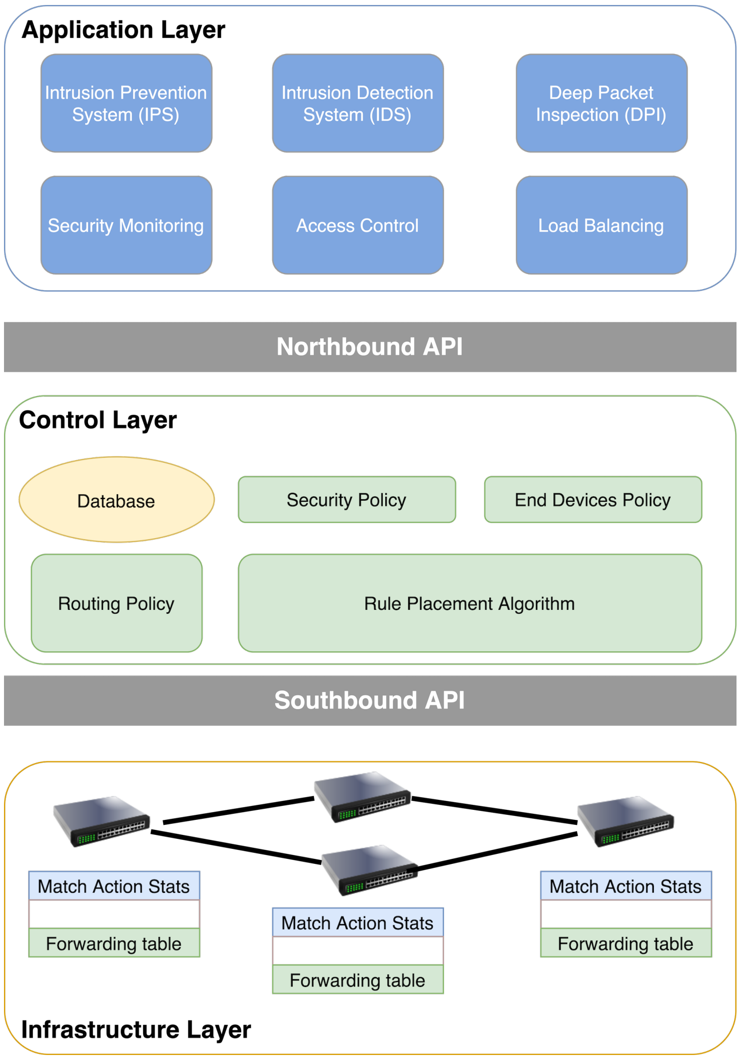

Figure 1. A Simplified SDN architecture consisting of the southbound and northbound APIs and the application layer [2].

Due to the flexibility in network control it offers, SDN has become an alternative approach for traditional security infrastructures. However, absolute security of the system is at stake if the SDN framework itself gets compromised. The controller is always prone to a single point of failure. Hence, an attack on the controller can lead to the failure of the entire network [3].

Major security problems in the SDN are issues of unauthorized controller access (intrusion), man-in-the-middle attack, and a flow rule change that modifies packets. Other pertinent issues are malicious packets hijacking the controller, denial of service by switch–controller communication flood, and configuration problems. Distributed denial of service (DDoS) is one the most common and dreadful threats that are aimed at successfully disrupting regular traffic from arriving at the controller. The attack is achieved by flooding the controller with more malicious packets than it can accommodate, thus rendering it inoperable. The attack is made possible by making use of multiple compromised switches (bot) for the production of malicious packets. The attacker forms a botnet, a group of bots, from the switches connected to the controller and then gains control over the entire network to operate after rendering the controller inoperable, as depicted in Figure 2.

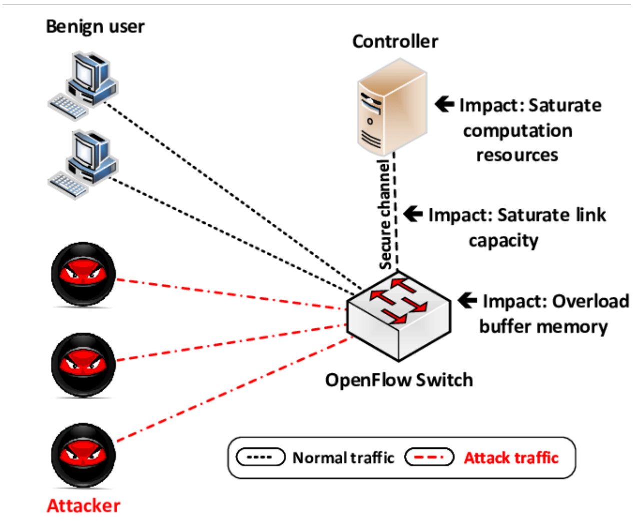
Benign user
Controller
Impact: Saturate computation resources
Secure channel
Impact: Saturate link capacity
Impact: Overload buffer memory
OpenFlow Switch
Normal traffic
Attack traffic
Attacker

Figure 2. SDN DDoS attack resulting from compromised nodes [4].

It is, therefore, necessary to implement a system that addresses this security threat. Traditional methods are insufficient, so machine learning based DDoS detection techniques have received more attention. In this paper, the feasibility and efficiency of applying variants of deep neural networks, namely convolutional neural network (CNN) and long short-term memory (LSTM), in training an ML model to detect and mitigate DDoS attack on SDN controllers are investigated. LSTM is an artificial recurrent neural network (RNN) architecture which is well-suited for data classification, processing and making predictions. According to the literature, many machine learning algorithms such as Support Vector Machine (SVM), K Nearest Neighbor (KNN), Artificial Neural Network (ANN), and Naïve Bayes (NB) have been explored in detecting DDoS attacks in the various layers of the SDN architecture. However, only the deep reinforcement learning-based algorithm has been applied in the application layer of the SDN to mitigate such attacks.

The main contribution of this work include:

- A new dataset comprising of normal and malicious (DDoS) traffic developed using Mininet and the Floodlight controller is collated.
- A DDoS defence mechanism based on the trained model for the identification and mitigation of DDoS attacks on the SDN controller is introduced.
- The performance of the selected deep learning candidate is compared with that of other machine learning linear models. These models are k-nearest neighbor (KNN), logistic regression, linear support vector classifier (LinearSVC), support vector classifier (SVC), decision tree, random forest, gradient boosting, Gaussian naïve Bayes (NB), Bernoulli NB, and multinomial NB. These models and the selected candidate model are trained based on the same generated dataset.
- The performance analysis of linear-based ML and neural network models in the detection and mitigation of DDoS flood attacks was done using various train–test split ratios (60/40, 70/30, and 80/20).

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed model and methodology. Section 4 discusses the results. Section 5 concludes the paper.

2. Related Work

In the field of network security, the advent of SDN has provided researchers' unparalleled control over network infrastructure by establishing a single control point for data flows that routes the entire network [5]. A range of literature was reviewed that is relevant to this current work and highlights from them are stated in the subsequent texts. To handle the issue of DDoS attack in SDN, researchers have proposed and implemented DDoS identification mechanisms based on artificial intelligence, mainly machine learning.

In [6], the authors used KNN, SVM, and Naïve Bayes to detect DDoS packets. KNN was most suitable with 97% accuracy, while SVM had 82% and Naïve Bayes 83%. The authors in [7] used Support Vector Machine (SVM) together with their own proposed algorithm, idle timeout adjustment (IA). They showed that their proposed approach differed from previous works and did better than the initial methods used. Neural network, Naive Bayes and SVM have been used in [3]. The neural network and Naive Bayes models provided 100% accuracy, while SVM presented 99% accuracy.

In [8], the authors used a support vector machine (SVM), and their results showed an average accuracy of 95.24%. The authors in [9] used Linear regression, Naïve Bayes, KNN, Decision Tree, Random Forest, SVM, and ANN. Their linear regression model achieved the highest accuracy, precision, and recall results at 98.65%. Naïve Bayes, on the other hand, showed the worst result at 97.45%. All others had accuracy between that of linear regression and naïve Bayes. In [10], the authors used Naïve Bayes. They had an average precision of 0.98 for training dataset with all features inclusive. They also recorded an average precision of 0.81 for training dataset with seven of the features removed. SVM has been used in detecting DDoS attacks; the model produced an accuracy of 99.8% [11].

In [12], the authors used Naïve Bayes, SVM, and Neural network. Naïve Bayes had an accuracy of up to 70% while SVM and the neural network had the same accuracy of 80%. The authors in [13] used SVM for DDoS attack detection. It was observed that the SVM algorithm achieved more than 98% accuracy on both the attacker and victim side for SYN flooding, ICMP flooding, and DNS reflection attacks. In [14], the authors used a deep neural network signature-based Intrusion Detection System (IDS). Their results show that the collaborative detection mechanism developed produced a true-positive rate of more than 90% with less than 5% false positives.

In [15], the authors worked on a reinforcement learning-based smart DDoS flood mitigation agent. Their findings demonstrate that the agent could effectively mitigate DDoS flood attacks of various protocols. Deep learning algorithms have also been used in SDN-based architectures to solve the problem of intrusion detection [16–18]. Other deep learning algorithms [19,20] have been applied in non-SDN architectures to detect DDoS and intrusion detection.

From the related works discussed, it is evident that machine learning has been used to identify DDoS attacks at all levels of the SDN architecture. Deep learning has been used in both SDN and non-SDN architectures for intrusion detection but not DDoS classification in SDN [17,18]. This circumstance makes it necessary to explore the feasibility and efficiency of applying CNN or RNN LSTM algorithms in the identification and mitigation of DDoS attacks on the controller. Table 1 shows a summary of the related works.

Table 1. Summary of related works.

No.	Paper Title	Research Method	Results	Strengths and Limitations
1.	OpenflowSIA: An optimized protection scheme for software-defined networks from flooding attacks [7]	Support Vector Machine (SVM) combined with their own proposed algorithm, idle timeout adjustment (IA)	They showed that their proposed approach differs from previous works and did better than the initial methods used to save the SDN resources	• Used only TCP and ICMP flood attacks. • Training and testing split ratio not mentioned.
2.	A DDoS attack detection method based on SVM in a software-defined network [5]	Support Vector Machine (SVM)	Their results show an average accuracy rate of 95.24%	• Used 6-tuple characteristic valued related to DDOS. • Training and testing split ratio not mentioned.
3.	A machine learning approach for detecting DoS attacks in SDN switches [3]	Neural Network, Naïve Bayes, SVM	Neural network and naive Bayes provided 100% accuracy with extracted features in their tests, while SVM provided 99% accuracy	• Used 60/40 training and testing dataset ratio. • Used to detect DDOS in the data plane.
4.	A machine learning approach for predicting DDoS traffic in software-defined networks [9]	Linear regression, Naïve Bayes, KNN, Decision tree, Random forest, SVM, ANN	Linear regression achieved high accuracy, precision, and recall results at 98.65%. Naïve Bayes showed the worst results at 97.45%. All others had accuracy between those of linear regression and naïve Bayes.	• Used only UDP and ICMP flood attacks. • Used 90/10, 80/20 and 70/30 split ratios.
5.	A machine learning-based collaborative DDoS mitigation mechanism in the software-defined network [10]	Naïve Bayes	They had an average precision of 0.98 for training dataset with all features inclusive and had an average precision of 0.81 for training dataset with seven features removed.	• Used a 41 feature DDOS dataset. • Training and testing split ratio not mentioned.
6.	An intelligent software-defined network controller for DDoS attack [6]	KNN, SVM, Naïve Bayes	KNN was most suitable with 97% accuracy than the other two algorithms deployed. SVM had 82%, and Naïve Bayes had 83%	• Used 67/33 training and testing dataset ratio. • Specific flood protocols used in research work not mentioned.
7.	DDoS attack identification and defence using SDN based on machine learning method [11]	SVM	The algorithm produced an accuracy of 0.998.	• Used 75/25 training and testing dataset ratio. • Eight feature dataset.
8.	Deep reinforcement learning-based smart mitigation of DDoS flooding in software-defined networks [15]	Deep reinforcement learning	Their findings demonstrated that the agent could effectively mitigate DDoS flooding attacks of various protocols.	• Eight feature dataset. • Used TCP, UDP, and ICMP floods.

Table 1. *Cont.*

No.	Paper Title	Research Method	Results	Strengths and Limitations
9.	Detection of distributed denial of service attacks using machine learning algorithms in software-defined networks [12]	Naïve Bayes, SVM, Neural network.	The naïve Bayes had an accuracy of up to 70% and SVM and the neural network had the same accuracy of 80%	• Two feature processed dataset. • Used only TCP flood.
10.	Multi-SDN based cooperation scheme for DDoS attack defence [13]	SVM	It was observed that the SVM algorithm would achieve more than 98% accuracy on both the attacker and victim side of SYN flooding, ICMP flooding, and DNS reflection attack.	• Used TCP, UDP, and ICMP floods • Training and testing split ratio not mentioned.

3. Methodology

Our anomaly detection technique is based on gathering certain parameters of the network when operating in a normal and also when subjected to a DDoS attack. These features include:

- The number of packets received at each switch
- The number of packets transmitted at each switch
- Packet count (number of packets of each flow)
- Protocol type (TCP, UDP or ICMP)
- Source IP
- Destination IP

The following assumptions were made in this research:

- The normal operation of the network is constant (the exchange of information between nodes has a particular profile), which forms the basis of our anomaly detection and defence mechanism.
- The training of the detection engine is done off-device; the model is only exported and used on the controller.

3.1. Architecture

A three-tier architecture consisting of seven switches, eight hosts (two hosts per switch) and an external controller (the single host connected to a switch) was used in this research. Figure 3 shows the three–tier topology implementation in Mininet.

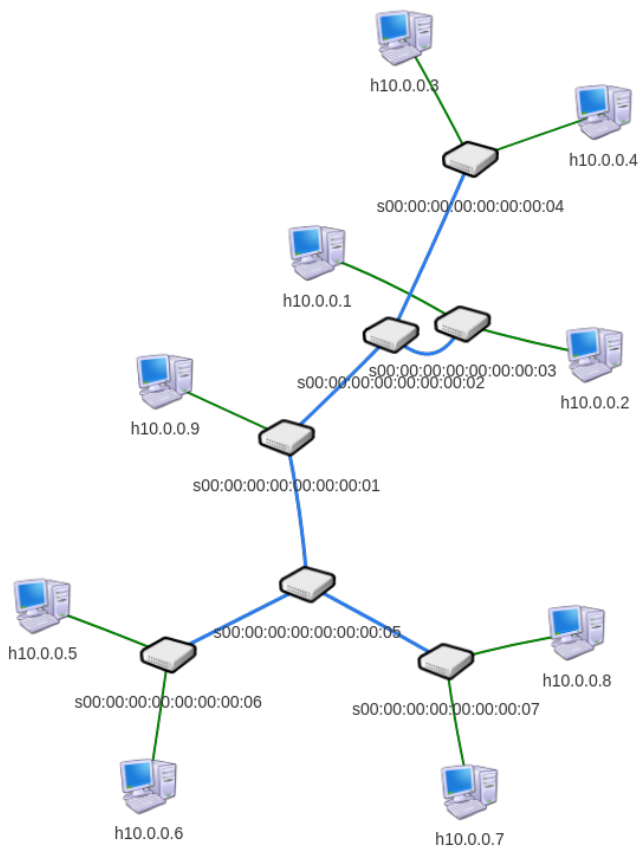

Figure 3. The three-tier topology deployed using Mininet.

3.2. Simulation Test Bed

The work was simulated using Mininet and floodlight as an external controller. The SDN Mininet simulator software was used in creating the three-tier data center topology (shown in Figure 3). The floodlight controller and OpenFlow switches were deployed using a virtual machine running Ubuntu. After setting up the network, a specialized tool known as hping3 [21] was used to generate data traffic. Using the *hping3* tool, we simulated a normal TCP, UDP, and ICMP traffic between two endpoints in the network. Afterwards, we simulated a DoS for TCP, UDP, and ICMP flood attacks. The statistics of the various switches were collected; these include:

- Number of hosts connected to each OpenFlow switch
- Number of packets (transmit and receive) of each OpenFlow switch
- Delay in millisecond (Round-Trip Time)
- Type of Transmission Protocol (TCP, UDP, or ICMP)
- Throughput
- Source IP
- Destination IP

The three different kinds of traffic generated by the *hping3* tool were UDP, TCP, and ICMP. The initial regular data generated were labeled as normal traffic. Afterwards, malicious traffic was generated using *hping3* for the various UDP, TCP, and ICMP floods, which was labeled as malicious traffic. In total, 10,031 data collected, 4270 being malicious traffic (approximately 43%) and 5761 (approximately 57%) normal traffic.

3.3. Scenarios Considered

The data gathered were used to build binary classification models using the following ML models: K-neighbor nearest (KNN), Logistic Regression, Linear SVC, SVC, Decision Tree, Random Forest, Gradient Boosting, and Naïve Bayes classifiers such as Gaussian, Bernoulli, and multinomial, as well as the main algorithms being investigated for the purpose of this work, namely RNN LSTM and CNN. The model summary for LSTM and CNN is shown in Figures 4 and 5, respectively.

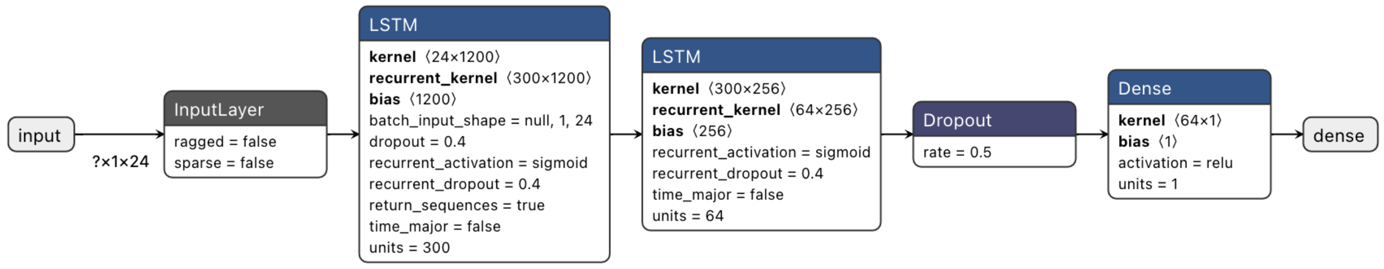

Figure 4. LSTM model summary.

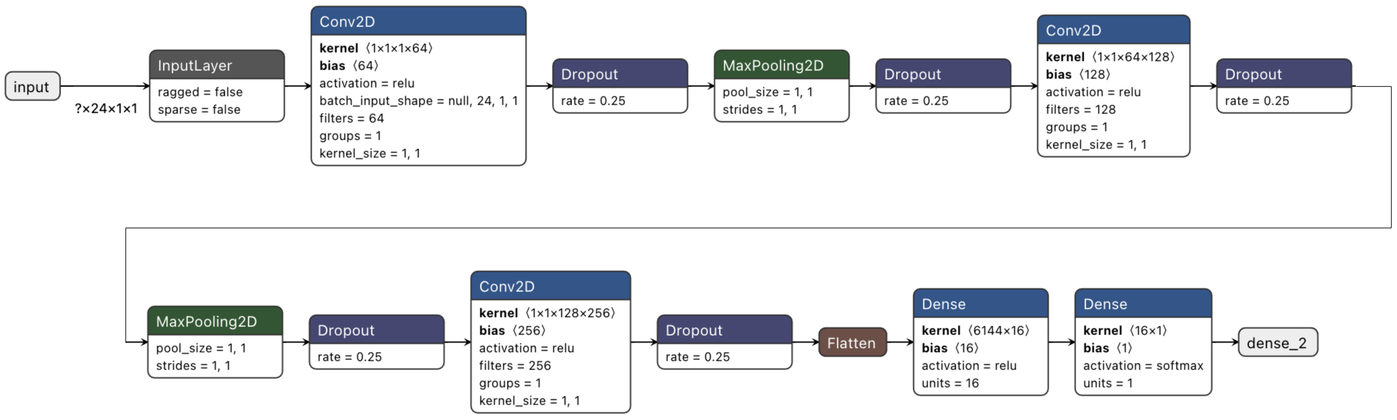

Figure 5. CNN model summary.

The performance of each model was evaluated based on the following key performance indicators: recall, accuracy, true negative rate, and the time used for identifying and mitigating the DDoS attacks. Accuracy is the proportion of correct predictions from the total dataset given. Recall is the percentage of predicted normal data against the total amount of normal data presented. The true-negative rate, on the other hand, measures the sum of true-negative against the sum of the condition of negative. It relates to the test's ability to detect malicious data against the total amount of malicious data presented.

Three scenarios were considered in the research. In the first scenario, 80% of the data were used in training and the remaining 20% for testing. The second scenario utilized 70% of the data for training and 30% for testing. In the third scenario, 60% of the data were used for training, while 40% were used for testing.

3.4. Detection and Defence Mechanism

The model was exported after evaluations were made and used in an application that runs on the controller. The application was used to measure the detection time when the controller was subjected to a DDoS attack. If the model detects a DDoS attack, its output is fed into the defence engine. The defence/mitigation engine is built on top of NetFilterQueue [22], a Linux system implementation that matches packets as accepted, dropped, altered, or given a mark. The rules to match packets have to be manually set, which does not make it scalable. Leveraging this, we implemented the defence mechanism by automatically matching packets based on the output of the detection algorithm.

4. Results and Discussion

The performance of the models in terms of precision and recall is shown in Figures 6–11. In terms of precision, the linear regression models (GradientBoosting, KNN, DecisionTree, GaussianNB, MultinomialNB, SVC, and LinearSVC) outperformed the LSTM model (a marginal difference of 1.4%) for the variants of split-ratios considered. Nonetheless, the LSTM model achieved a good tradeoff between a high recall and precision, as shown in Figures 12–14, which makes it suitable for DDoS classification.

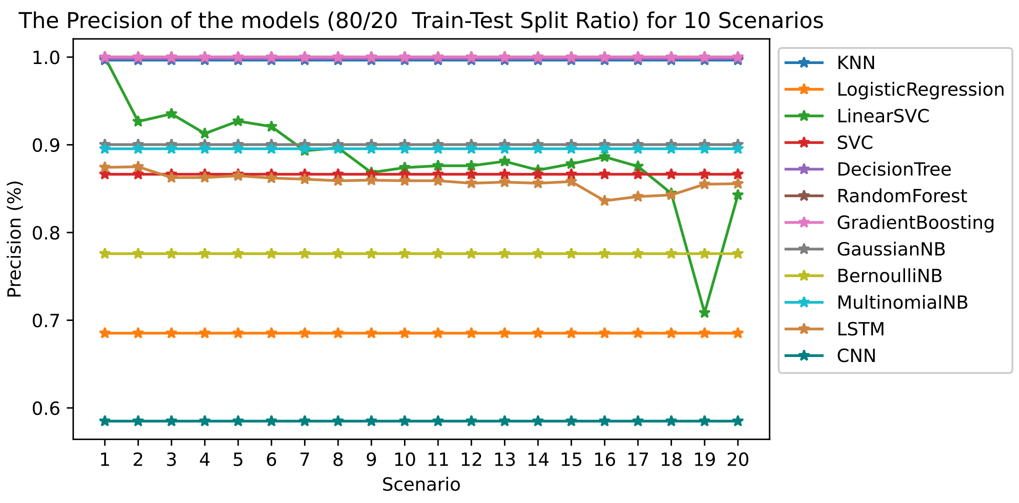

Figure 6. A comparison of the precision of the models for an 80/20 train–test split ratio.

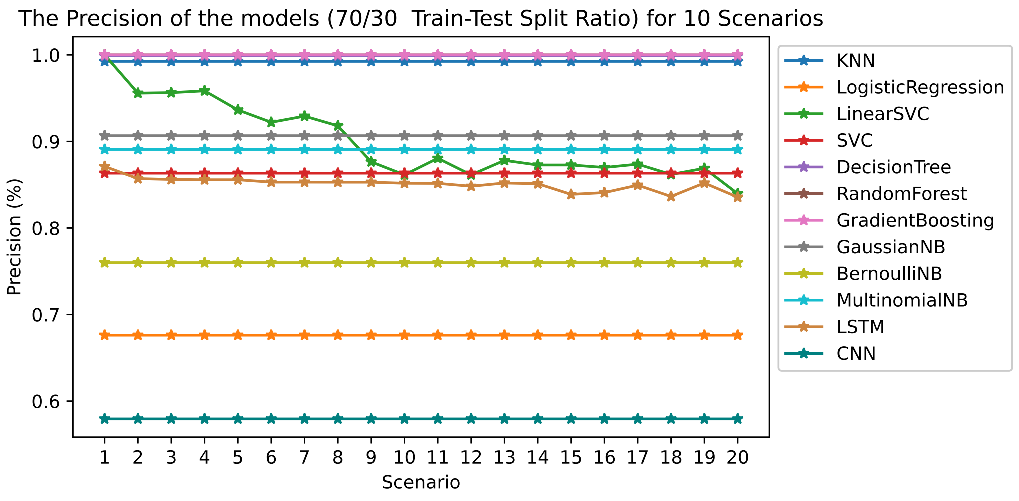

Figure 7. A comparison of the precision of the models for a 70/30 train–test split ratio.

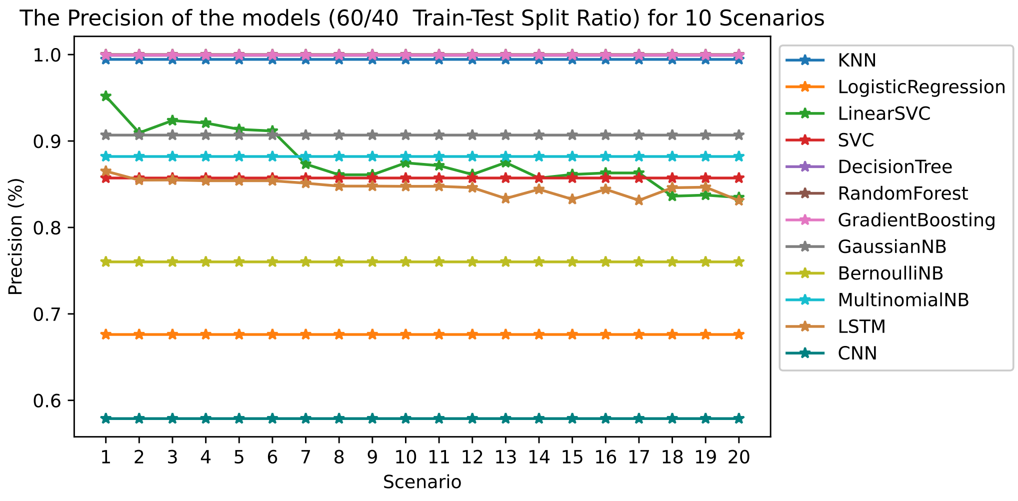

Figure 8. A comparison of the precision of the models for a 60/40 train–test split ratio.

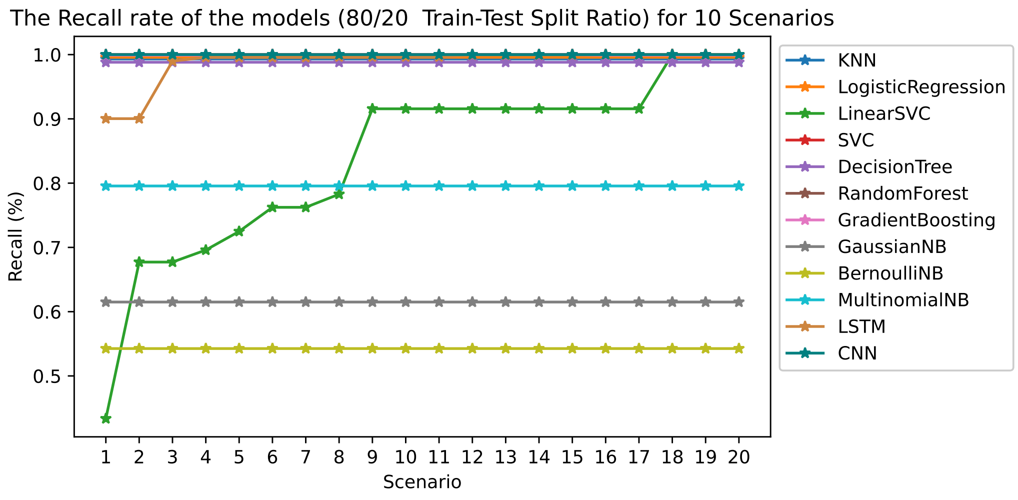

Figure 9. A comparison of the recall of the models for a 80/20 train–test split ratio.

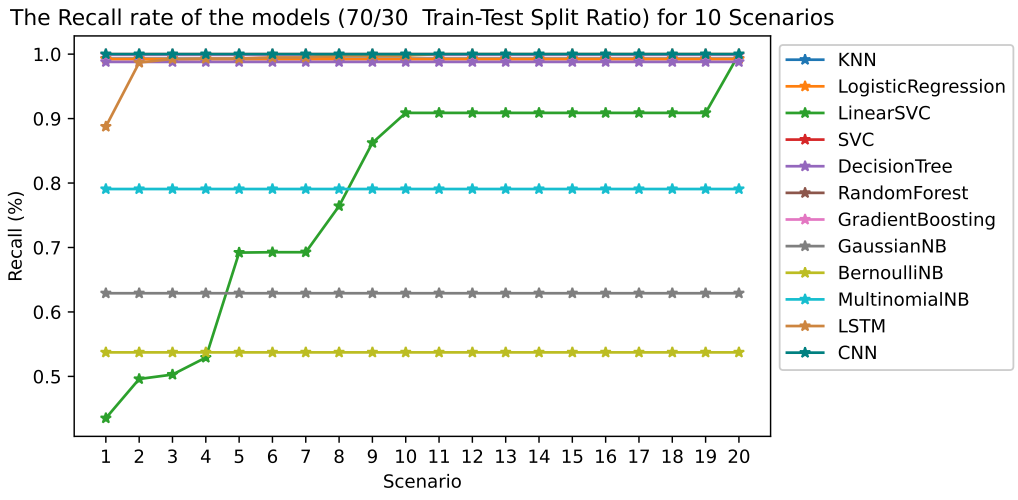

Figure 10. A comparison of the recall of the models for an 70/30 train–test split ratio.

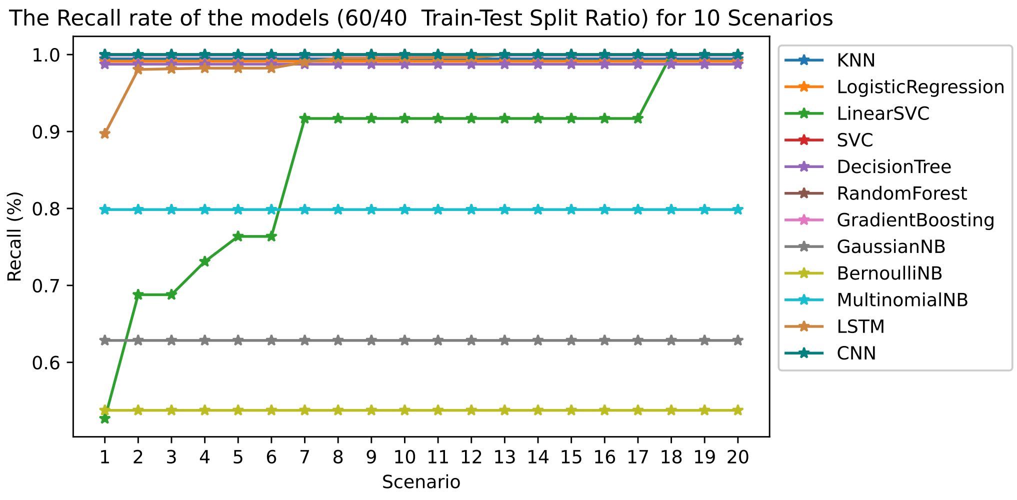

Figure 11. A comparison of the recall of the models for a 60/40 train–test split ratio.

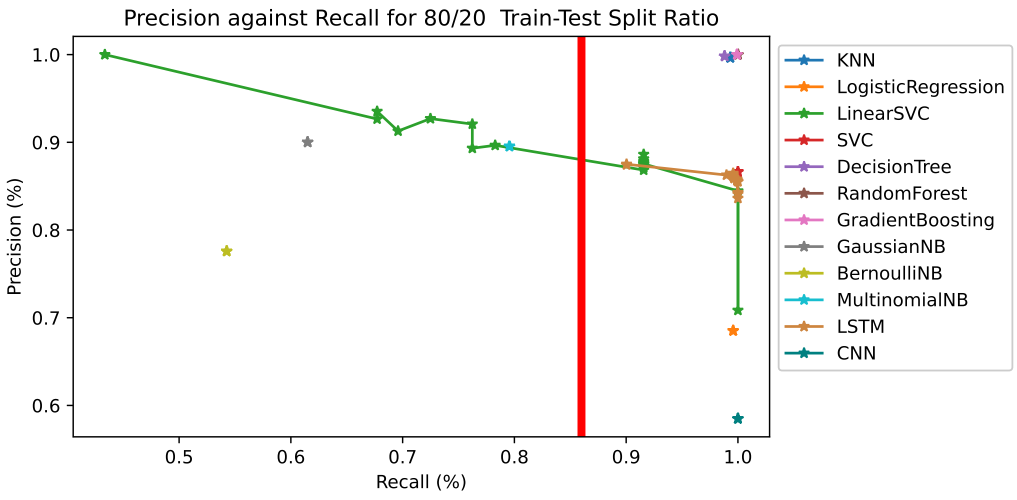

Figure 12. A comparative analysis of the precision and recall of each model using an 80/20 train–test split ratio.

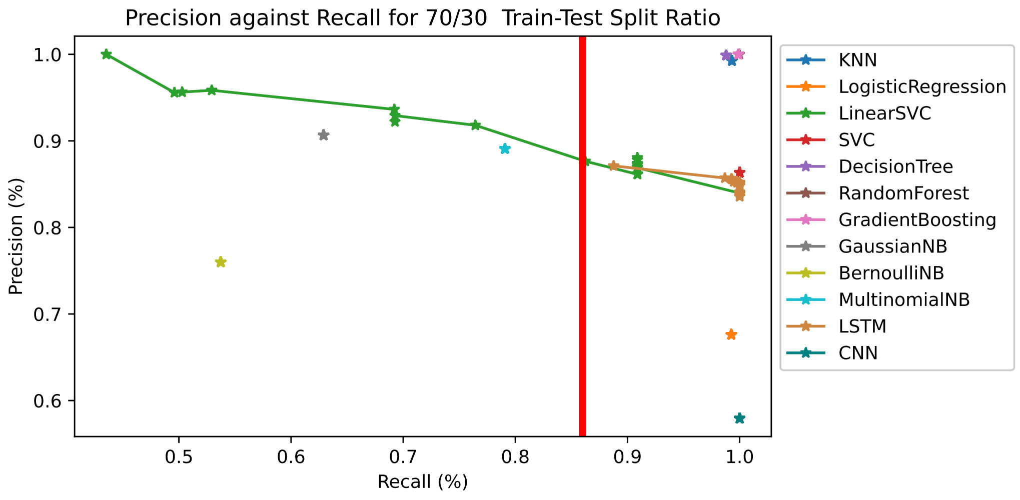

Figure 13. A comparative analysis of the precision and recall of each model using a 70/30 train–test split ratio.

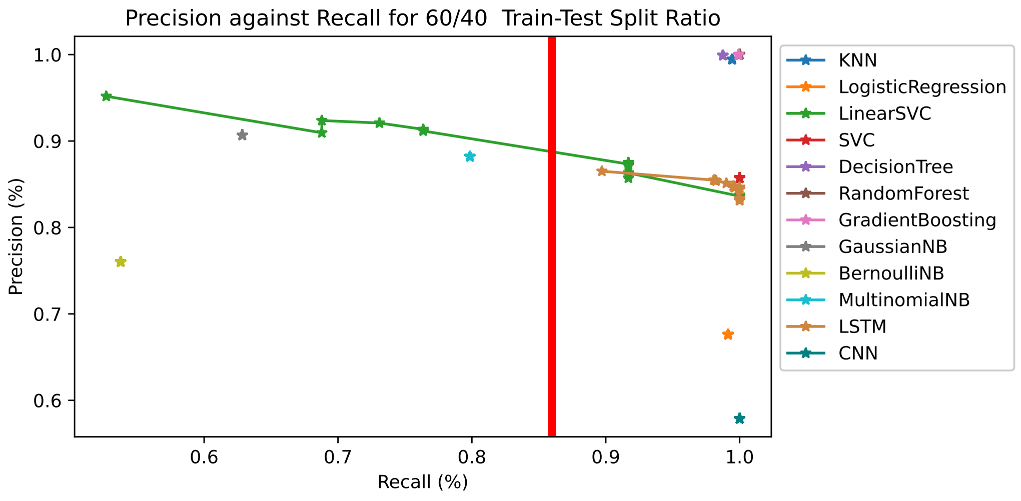

Figure 14. A comparative analysis of the precision and recall of each model using a 60/40 train–test split ratio.

A summary of the performance of the models in shown in Table 2. The values in bold indicate the highest accuracy, recall and true-negative rate of each model.

Table 2. A Summary of the performance of the models in terms of accuracy, recall and true-negative rate.

	ACCURACY			**RECALL**			**TRUE-NEGATIVE RATE**		
MODEL	**80/20**	**60/40**	**70/30**	**80/20**	**60/40**	**70/30**	**80/20**	**60/40**	**70/30**
KNN	**99.40**	99.35	99.17	**99.50**	99.24	98.98	99.30	**99.40**	99.30
LOGISTIC REGRESSION	**90.33**	72.00	72.00	**77.30**	34.73	34.52	**99.60**	99.13	99.26
LINEAR SVC	86.85	**87.49**	87.04	80.20	81.70	**81.76**	91.60	**91.69**	90.88
SVC	**70.85**	68.95	69.53	**29.80**	26.26	27.56	**100.00**	**100.00**	**100.00**
DECISION TREE	99.20	99.23	**99.24**	99.80	**99.88**	**99.88**	**98.80**	98.75	98.79
RANDOM FOREST	**100.00**	**100.00**	99.97	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	99.94
GRADIENT BOOSTING	**99.95**	99.93	99.93	**100.00**	99.95	100.00	**99.90**	**99.90**	99.88
G. NAÏVE BAYES	73.49	**74.76**	74.75	90.30	**91.12**	91.08	61.50	62.80	**62.90**
B. NAÏVE BAYES	**64.08**	63.42	63.36	**77.90**	76.68	76.62	**54.30**	53.76	53.73
M. NAÏVE BAYES	**82.61**	82.16	82.26	**86.90**	85.32	86.66	79.60	**79.85**	79.06
RNN LSTM	86.60	89.51	**89.63**	**81.75**	75.08	75.34	90.0	**100.00**	**100.00**
CNN	**66.00**	**66.00**	**66.00**	0.00	0.00	0.00	**100.00**	**100.00**	**100.00**

4.1. Detection of DDoS Attack Using LSTM Model

This section of the experiment was aimed at creating scenarios of DDoS in the form of ICMP, UDP, and TCP flood attacks on the controller. It also determined how long it took the trained LSTM model to detect the attacks. Ten scenarios were considered, and Figure 15 shows the time it took the trained model to detect each flood attack, using a 60/40 train–test ratio.

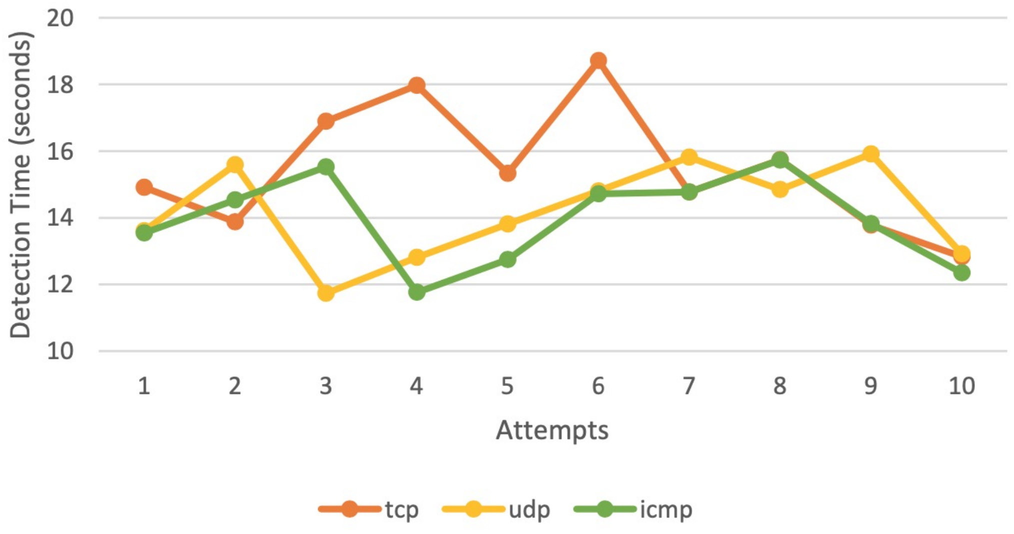

Figure 15. LSTM model detection time using a 60/40 train–test split ratio for TCP, UDP, and ICMP flood attacks.

In Figure 15, the highest time it took the LSTM model to detect TCP DDoS flood among the 10 attempts was 18.70 s and the least time was 12.83 s. It took 11.73 and 15.90 s, respectively, as the lowest and highest times to detect the UDP DDoS flood attack. In addition, it took 11.76 and 15.73 s, respectively, as the lowest and highest times to detect ICMP flood attack among the 10 scenarios considered.

Figure 16 shows the detection time of the LSTM model when a 70/30 train–test ratio was used.

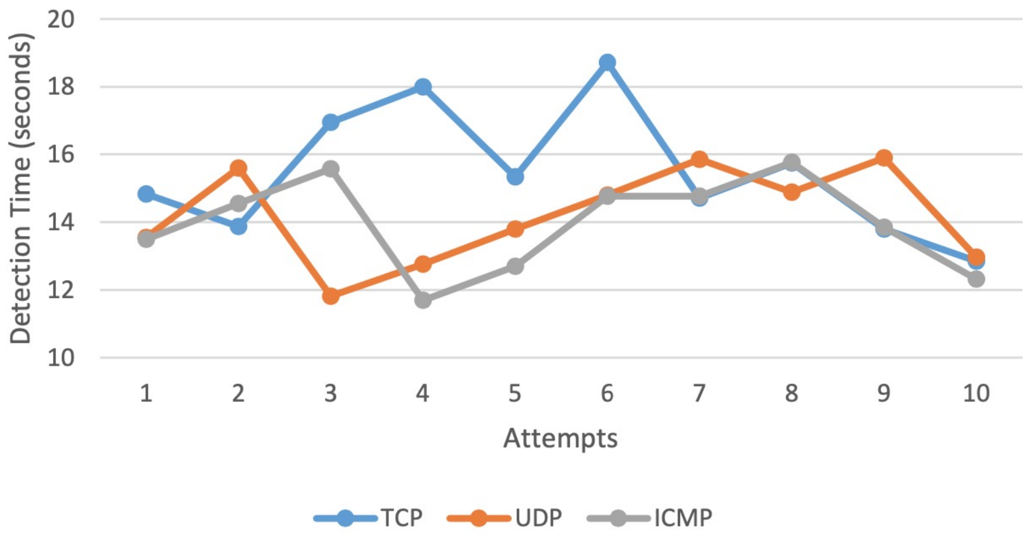

Figure 16. LSTM model detection time using a 70/30 train–test split ratio for TCP, UDP, and ICMP flood attacks.

In Figure 16, the highest time it took the LSTM model to detect the TCP flood among the 10 scenarios was 18.71 s and the least time was 12.84 s. It took 11.81 and 15.89 s, respectively, as the lowest and highest times to detect UDP flood attack. In addition, it took 11.68 and 15.76 s, respectively, as the lowest and highest times to detect ICMP flood attack.

Figure 17 shows the detection time of the LSTM model when a 80/20 train–test ratio was used.

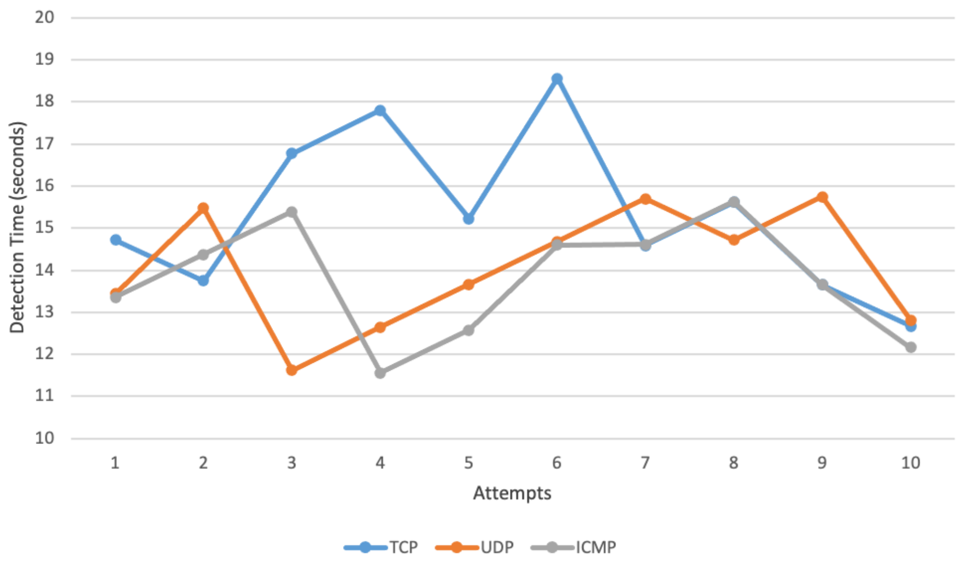

Figure 17. LSTM model detection time using an 80/20 train–test split ratio for TCP, UDP, and ICMP flood attacks.

In Figure 17, the highest time it took the LSTM model to detect the TCP flood was 18.55 s and the least time was 12.67 s. It took 11.62 and 15.70 s, respectively, as the lowest and highest times to detect the UDP flood attack. In addition, it took 11.56 and 15.63 s, respectively, as the lowest and highest times to detect the ICMP flood attack.

Table 3 gives a summary of the detection time of the LSTM model.

Table 3. A summary of LSTM model's detection time for TCP, UDP, and ICMP flood attacks.

Split Ratio		TCP	UDP	ICMP
60/40	Highest time	18.70	15.91	15.73
	Lowest time	12.83	11.73	11.76
70/30	Highest time	18.71	15.89	15.76
	Lowest time	12.84	11.81	11.68
80/20	Highest time	18.55	15.70	15.63
	Lowest time	12.67	11.62	11.56

From Table 3, It is observed that the 80/20 split had the lowest values (indicated in bold) for the detection of ICMP, TCP and UDP flood attacks .

4.2. Mitigation of DDoS Attack Using LSTM Model

This section of the experiment was aimed at creating scenarios of DDoS in the form of ICMP, UDP, and TCP flood attacks on the controller. It also determined how long it took for the trained LSTM model to mitigate the attack after detection. After the controller detects the DDoS attack, the IP, protocol type, and destination port are sent to the mitigation engine, which instantly drops all packets from that particular source IP. Ten scenarios (attempts) were considered. The train–test ratios used were 60/40, 70/30, and 80/20.

Figure 18 shows the mitigation time of the LSTM model when a 60/40 train–test ratio was used.

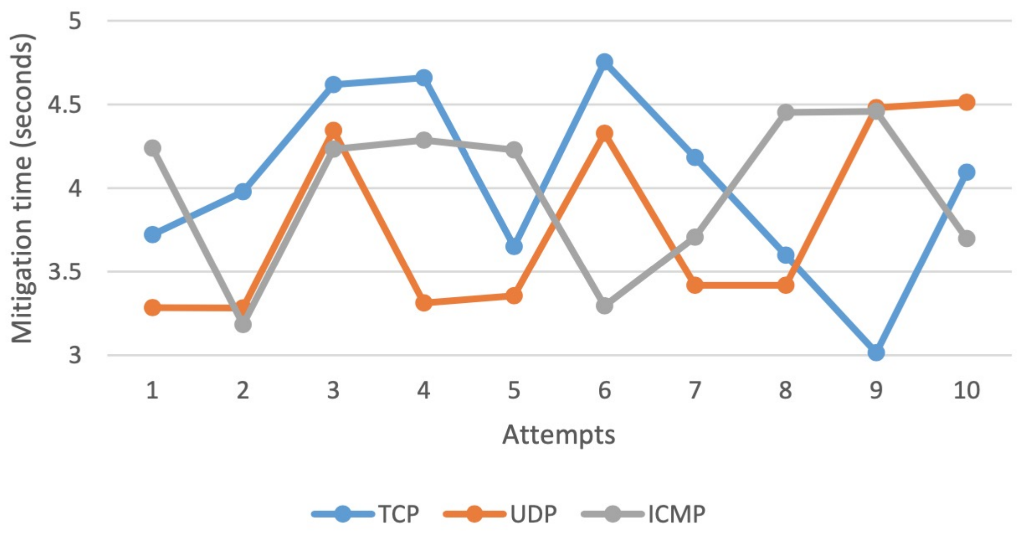

Figure 18. LSTM model mitigation time using a 60/40 train–test split ratio for TCP, UDP, and ICMP flood attacks.

In Figure 18, the highest time it took the LSTM model to mitigate the TCP DDoS flood attack was 4.75 s and the least time was 3.01 s. It took 3.28 and 4.51 s, respectively, as the lowest and highest times to mitigate the UDP flood attack. In addition, it took 3.18 and 4.45 s, respectively, as the lowest and highest times to mitigate the ICMP flood attack.

Figure 19 shows the mitigation time of the LSTM model when a 70/30 train–test ratio was used.

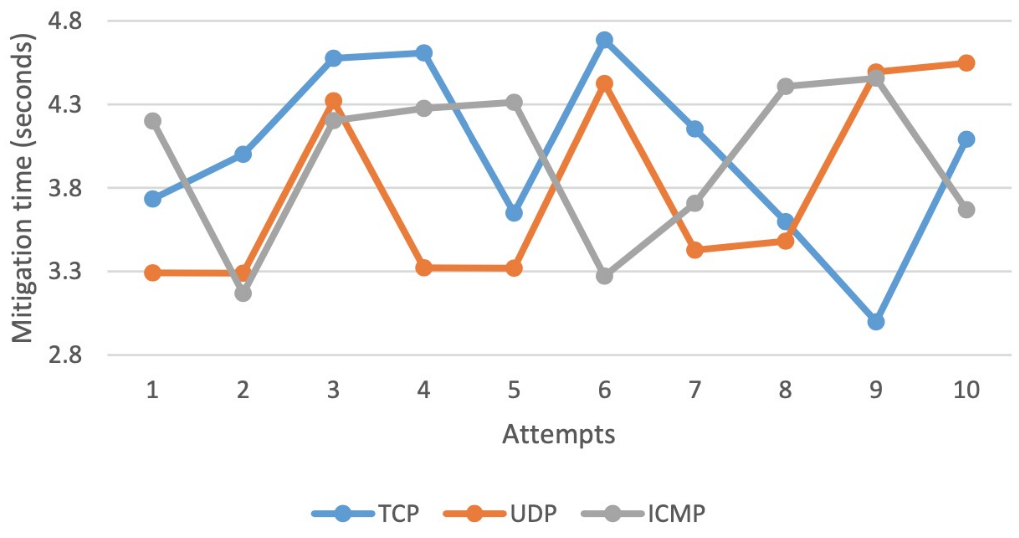

Figure 19. LSTM model mitigation time using a 70/30 train–test split ratio for TCP, UDP, and ICMP flood attacks.

In Figure 19, the highest time it took the LSTM model to mitigate the TCP flood attack was 4.68 s and the least time was 2.99 s. It took 3.28 and 4.54 s, respectively, as the lowest and highest times to mitigate the UDP flood attack. In addition, it took 3.16 and 4.45 s, respectively, as the lowest and highest times to mitigate the ICMP flood attack.

Figure 20 shows the mitigation time of the LSTM model when a 80/20 train–test ratio was used.

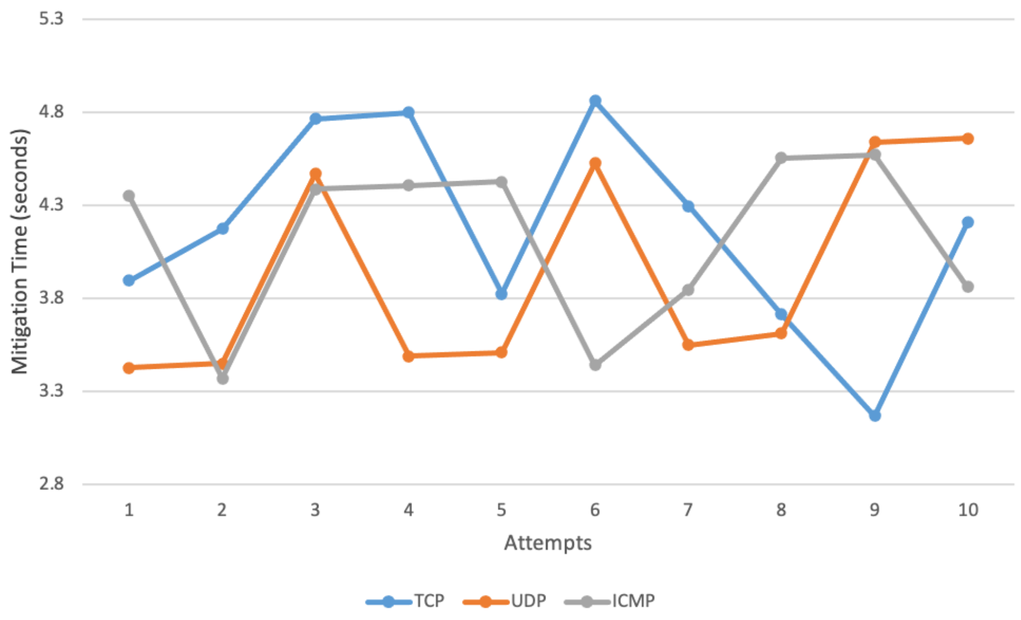

Figure 20. LSTM model mitigation time using an 80/20 train–test split ratio for TCP, UDP, and ICMP flood attacks.

In Figure 20, the highest time it took the LSTM model to mitigate the TCP flood attack was 4.86 s and the least time was 3.16 s. It took 3.42 and 4.65 s, respectively, as the lowest and highest times to mitigate the UDP flood attack. In addition, it took 3.36 and 4.57 s, respectively, as the lowest and highest times to mitigate the ICMP flood attack.

Table 4 gives a summary of the mitigation time of the LSTM model.

Table 4. A summary of LSTM model's mitigation time for TCP, UDP, and ICMP flood attacks.

Split Ratio		TCP	UDP	ICMP
60/40	Highest time	4.75	4.51	4.45
	Lowest time	3.01	3.28	3.18
70/30	Highest time	**4.68**	**4.54**	**4.45**
	Lowest time	**2.99**	**3.28**	**3.16**
80/20	Highest time	4.86	4.65	4.57
	Lowest time	3.16	3.42	3.36

It can be observed that the 70/30 split had the lowest values (indicated in bold) in time for mitigating ICMP, TCP, and UDP flood attacks.

4.3. Comparison of the LSTM Model with the Best Performing Linear-Based ML Models

We compared the best performing linear-based ML models with the LSTM model in terms of detection and mitigation time. According to the observed detection times for all the three different ratio splits, the 80/20 split had the best time values for detecting DDoS attacks. Hence, these values were compared with that of the best performing linear models in the sample split ratio. This is shown in Figures 21–24.

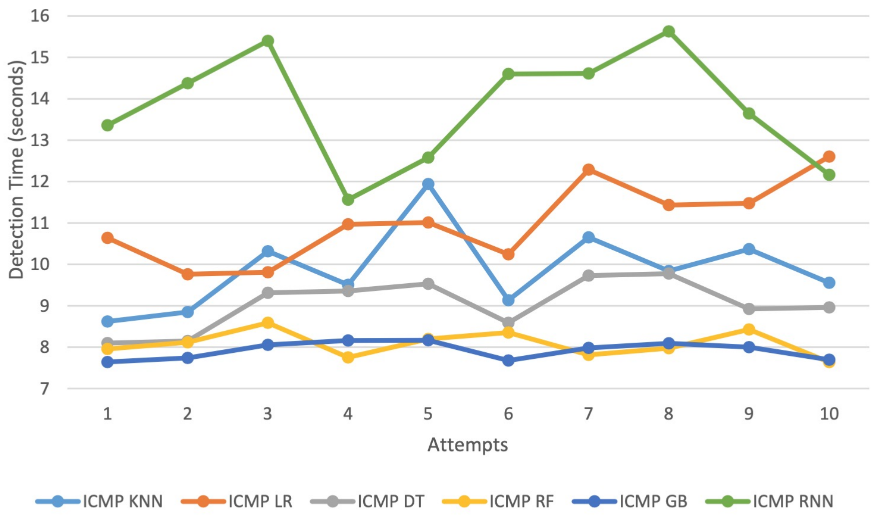

Figure 21. Comparison of the LSTM model with the best performing linear-based ML models: detection time for ICMP flood attack.

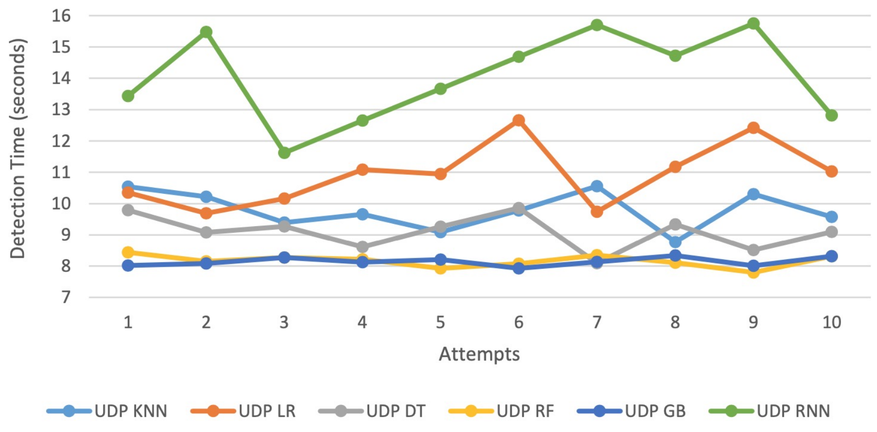

Figure 22. Comparison of the LSTM model with the best performing linear-based ML models: detection time for UDP flood attack.

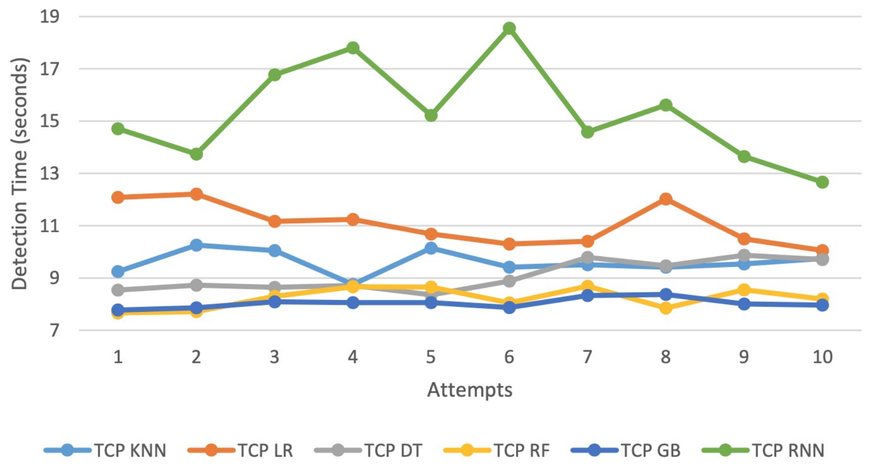

Figure 23. Comparison of the LSTM model with the best performing linear-based ML models: detection time for TCP flood attack.

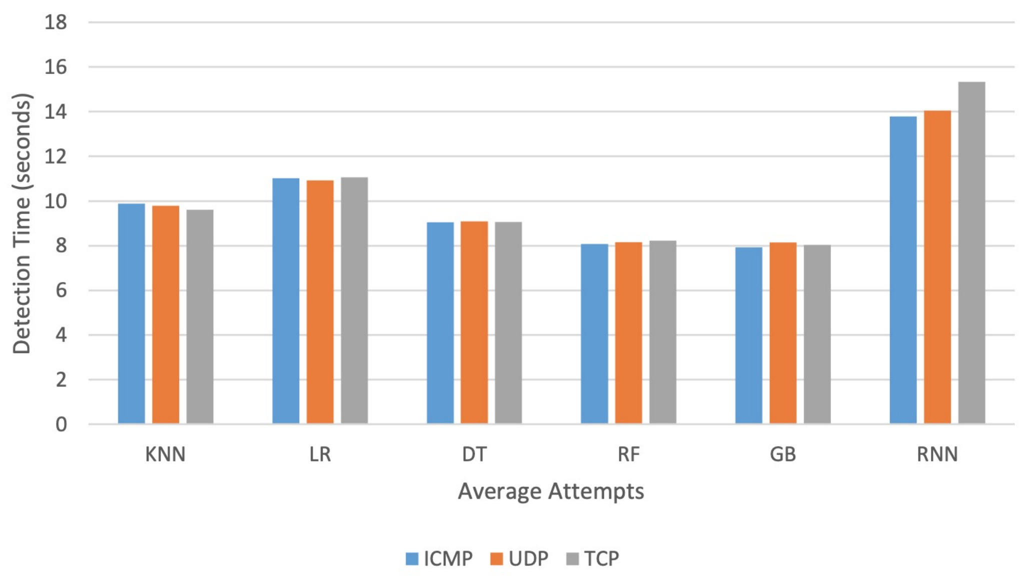

Figure 24. Comparison of the LSTM model with the best performing linear-based ML models: average detection time for ICMP, UDP, and TCP flood attacks.

It was observed that the LSTM model had a higher time in detecting DDoS attacks on the SDN controller. However, it was not more than 4 s from the highest time for the linear-based models, which makes it a good result.

In addition, from the mitigation time for all the three different ratio splits, it was observed that the 70/30 split had the best time values for mitigating the DDoS attacks. Hence, these values were compared to that of the best performing linear models in the sample split ratio. This is shown in Figures 25–28.

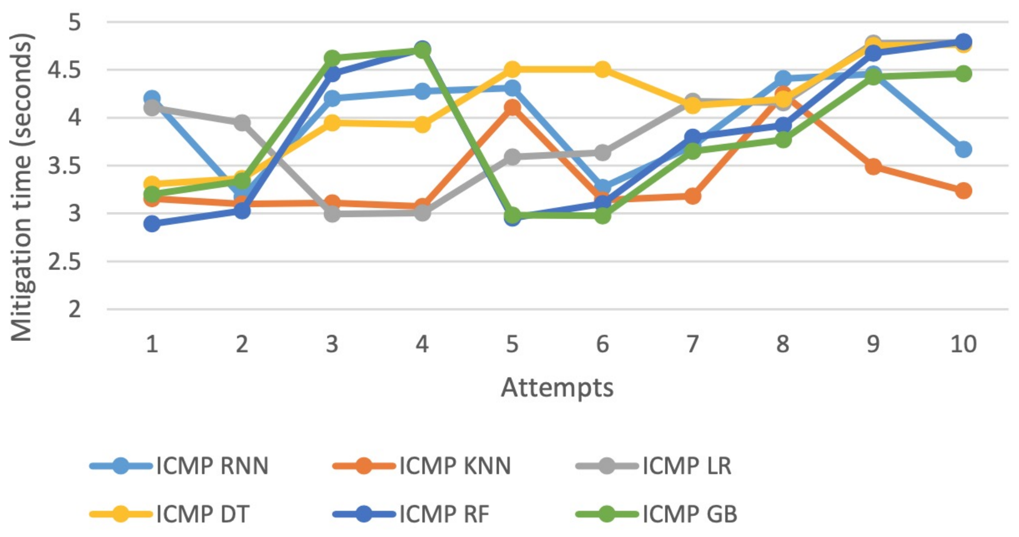

Figure 25. Comparison of the LSTM model with the best performing linear-based ML models: mitigation time for ICMP flood attack.

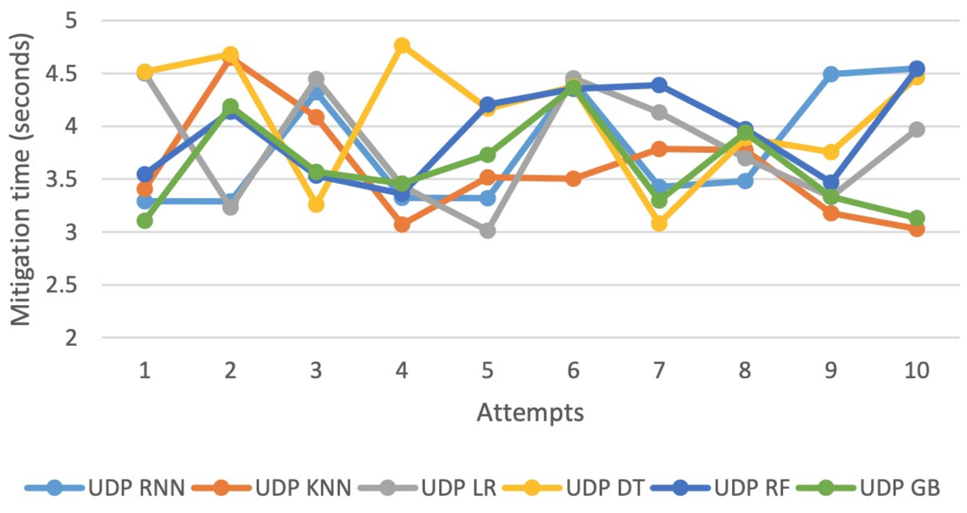

Figure 26. Comparison of the LSTM model with the best performing linear-based ML models: mitigation time for UDP flood attack.

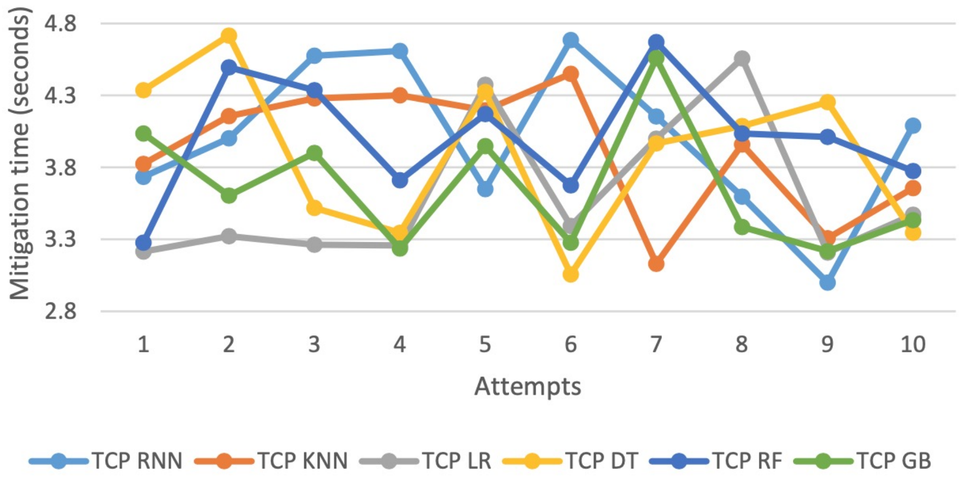

Figure 27. Comparison of the LSTM model with the best performing linear-based ML models: mitigation time for TCP flood attack.

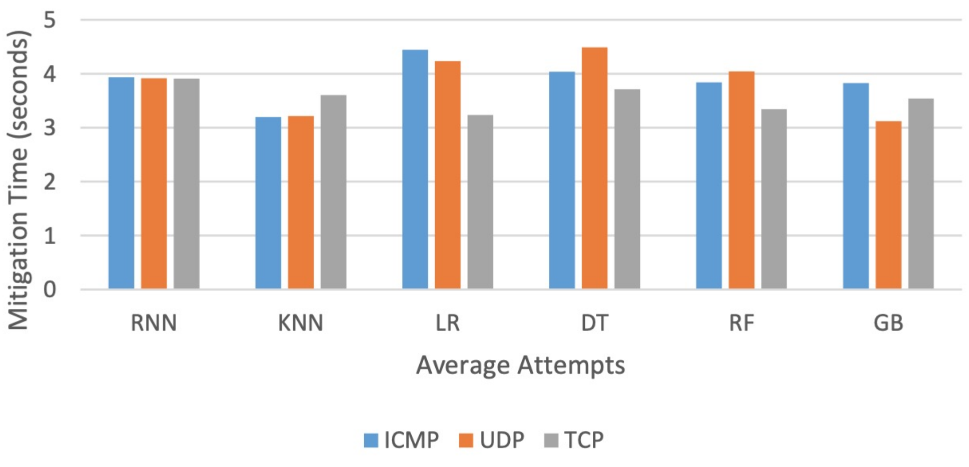

Figure 28. Comparison of the LSTM model with the best performing linear-based ML models: average mitigation time for ICMP, UDP, and TCP flood attacks.

In Figure 28, all the classification models (both the linear models and the LSTM model) after detecting a DDoS flood attack, took almost the same time to mitigate the attack. Hence, it can be concluded that the LSTM model performs exceptionally just as linear models will in the mitigation of DDoS attacks on the SDN controller. It was also observed that, aside KNN and GB, the RNN LSTM model performed better in some protocols than the three remaining linear models.

Table 5, shows a comparison of the classification models from this research and other related works.

Table 5. Comparison of the classification models with other related works.

	Classification Model	Accuracy from Other Works	Accuracy from This Research
1.	KNN (DDoS, SDN)	97% [6]	99.40%
2.	SVM (DDoS, SDN)	82% [6]	86.85%
3.	Naive Bayes (DDoS, SDN)	83% [6]	82.61%
4.	SVM (DDoS, SDN)	98% [13]	86.85%

In Table 5, it can be observed that the LSTM model (which achieved an accuracy of 89.63%) is quite good in comparison to the linear-based ML models. Furthermore, the LSTM model has a good tradeoff between precision and recall rates which makes it a good classification model for DDoS detection (Figures 12–14).

5. Conclusions and Future Work

In this research, we demonstrated that RNN LSTM is a viable deep learning algorithm that can be applied in the detection and mitigation of DDoS in the SDN controller. In addition, it was observed that the split ratio of the training and testing dataset could give different results in the performance of a deep learning algorithm used in a specific work. Thus, a 70/30 split produces a better model accuracy when compared to 80/20 and 60/40 split ratios. It can be concluded that RNN LSTM is also a good model for the identification and mitigation of DDoS attacks in the SDN architecture. The software-defined network used in this work was designed and tested within a virtual environment which simulates a software running on a set of network devices. Thus, future works may be carried out in a real SDN architecture to test how this application works in real-time. Future works will also explore the gathering of a larger dataset with in depth feature selection analysis and tuning of hyper-parameters to achieve better performance when using the neural network models.

Author Contributions: Conceptualization, J.D.G. and A.A.B.-A.; methodology, A.A.B.-A. and J.O.A.; validation, H.N.-M.; formal analysis, K.A.-B.O.; investigation, H.N.-M. and K.A.-B.O.; resources, J.D.G.; data curation, A.A.B.-A. and J.O.A.; writing—original draft preparation, A.A.B.-A., writing—reviewing and editing, J.O.A. and H.N.-M.; project administration, J.D.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in this study is available at https://github.com/jayluxferro/SDN-DoS/blob/master/README.md.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. McKeown, N.; Anderson, T.; Balakrishnan, H.; Parulkar, G.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* **2008**, *38*, 69–74. [CrossRef]
2. Sangodoyin, A.; Sigwele, T.; Pillai, P.; Hu, Y.F.; Awan, I.; Disso, J. DoS Attack Impact Assessment on Software Defined Networks. In Proceedings of the International Conference on Wireless and Satellite Systems, WiSATS 2017: Wireless and Satellite Systems, Oxford, UK, 14–15 September 2017; pp 11–22.
3. Abhiroop, T.; Babu, S.; Manjo, B.S. A Machine Learning Approach for Detecting DoS Attacks in SDN Switches. In Proceedings of the Twenty Fourth National Conference on Communications (NCC), Hyderabad, India, 25–28 February 2018; pp. 1–6. ISBN 978-1-5386-1224-8.
4. Conti, M.; Gangwal, A.; Gaur, M.S. A comprehensive and effective mechanism for DDoS detection in SDN. In Proceedings of the IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, Italy, 9–11 October 2017; pp. 1–8. [CrossRef]
5. Dotcenko, S.; Vladyko, A.; Letenko, I. A fuzzy logic-based information security management for software-defined networks. In Proceedings of the 16th International Conference on Advanced Communication Technology, Pyeongchang, Korea, 16–19 February 2014; pp. 167–171. [CrossRef]
6. Prakash, A.; Priyadarshini, R. An Intelligent Software defined Network Controller for preventing Distributed Denial of Service Attack. In Proceedings of the Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 585–589. [CrossRef]
7. Phan, T.V.; Toan, T.V.; Tuyen, D.V.; Huong, T.T.; Thanh, N.H. OpenFlowSIA: An optimized protection scheme for software-defined networks from flooding attacks. In Proceedings of the IEEE Sixth International Conference on Communications and Electronics (ICCE), Ha Long, Vietnam, 27–29 July 2016; pp. 13–18. [CrossRef]
8. Ye, J.; Cheng, X.; Zhu, J.; Feng, L.; Song, L. A DDoS Attack Detection Method Based on SVM in Software Defined Network. *Secur. Commun. Netw.* **2018**, *2018*, 1–8. [CrossRef]
9. Sahoo, K.S.; Iqbal, A.; Maiti, P.; Sahoo, B. A Machine Learning Approach for Predicting DDoS Traffic in Software Defined Networks. In Proceedings of the International Conference on Information Technology (ICIT), Bhubaneswar, India, 19–21 December 2018; pp. 199–203. [CrossRef]
10. Mohammed, S.S. A New Machine Learning-based Collaborative DDoS Mitigation Mechanism in Software-Defined Network. In Proceedings of the 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Limassol, Cyprus, 15–17 October 2018; pp. 1–8. [CrossRef]
11. Yang, L.; Zhao, H. DDoS Attack Identification and Defense Using SDN Based on Machine Learning Method. In Proceedings of the 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN), Yichang, China, 16–18 October 2018; pp. 174–178. [CrossRef]
12. Meti, N.; Narayan, D.G.; Baligar, V.P. Detection of distributed denial of service attacks using machine learning algorithms in software defined networks. In Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1366–1371. [CrossRef]
13. He, B.; Zou, F.; Wu, Y. Multi-SDN Based Cooperation Scheme for DDoS Attack Defense. In Proceedings of the Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC), Shanghai, China, 18–19 October 2018; pp. 1–7. [CrossRef]
14. Ujjan, R.M.A.; Pervez, Z.; Dahal, K. Suspicious Traffic Detection in SDN with Collaborative Techniques of Snort and Deep Neural Networks. In Proceedings of the IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Exeter, UK, 28–30 June 2018; pp. 915–920. [CrossRef]
15. Liu, Y.; Dong, M.; Ota, K.; Li, J.; Wu, J. Deep Reinforcement Learning based Smart Mitigation of DDoS Flooding in Software-Defined Networks. In Proceedings of the IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, Spain, 17–19 September 2018; pp. 1–6. [CrossRef]
16. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In Proceedings of the 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), Montreal, QC, Canada, 25–29 June 2018; pp. 202–206. [CrossRef]
17. Dey, S.K.; Md Rahman, M. Flow Based Anomaly Detection in Software Defined Networking: A Deep Learning Approach With Feature Selection Method. In Proceedings of the 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 13–15 September 2018; pp. 630–635. [CrossRef]
18. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. (2016) Deep learning approach for Network Intrusion Detection in Software Defined Networking. In Proceedings of the International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016; pp. 258–263. [CrossRef]

19. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
20. Asad, M.; Asim, M.; Javed, T.; Beg, M.O.; Mujtaba, H.; Abbas, S. DeepDetect: Detection of Distributed Denial of Service Attacks Using Deep Learning. *Comput. J.* **2019**, bxz064. [CrossRef]
21. Hping3. Available online: https://github.com/antirez/hping (accessed on 27 April 2020).
22. NetFilterQueue. Available online: https://github.com/kti/python-netfilterqueue (accessed on 27 April 2020).