future internet

MDPI

Article

An Improved Deep Learning Model for DDoS Detection Based on Hybrid Stacked Autoencoder and Checkpoint Network

Amthal K. Mousa * and Mohammed Najm Abdullah

Computer Engineering Department, University of Technology-Iraq, Baghdad P.O. Box 10071, Iraq; mohammed.n.abdullah@uotechnology.edu.iq
* Correspondence: amthal.k.mousa@uotechnology.edu.iq

Abstract: The software defined network (SDN) collects network traffic data and proactively manages networks. SDN's programmability makes it excellent for developing distributed applications, cybersecurity, and decentralized network control in multitenant data centers. This exceptional architecture is vulnerable to security concerns, such as distributed denial of service (DDoS) attacks. DDoS attacks can be very serious due to the fact that they prevent authentic users from accessing, temporarily or indefinitely, resources they would normally expect to have. Moreover, there are continuous efforts from attackers to produce new techniques to avoid detection. Furthermore, many existing DDoS detection methods now in use have a high potential for producing false positives. This motivates us to provide an overview of the research studies that have already been conducted in this area and point out the strengths and weaknesses of each of those approaches. Hence, adopting an optimal detection method is necessary to overcome these issues. Thus, it is crucial to accurately detect abnormal flows to maintain the availability and security of the network. In this work, we propose hybrid deep learning algorithms, which are the long short-term memory network (LSTM) and convolutional neural network (CNN) with a stack autoencoder for DDoS attack detection and checkpoint network, which is a fault tolerance strategy for long-running processes. The proposed approach is trained and tested with the aid of two DDoS attack datasets in the SDN environment: the DDoS attack SDN dataset and Botnet dataset. The results show that the proposed model achieves a very high accuracy, reaching 99.99% in training, 99.92% in validation, and 100% in precision, recall, and F1 score with the DDoS attack SDN dataset. Also, it achieves 100% in all metrics with the Botnet dataset. Experimental results reveal that our proposed model has a high feature extraction ability and high performance in detecting attacks. All performance metrics indicate that the proposed approach is appropriate for a real-world flow detection environment.

Keywords: DDoS detection; distributed denial of service; software defined networking; SDN; network security

Citation: Mousa, A.K.; Abdullah, M.N. An Improved Deep Learning Model for DDoS Detection Based on Hybrid Stacked Autoencoder and Checkpoint Network. *Future Internet* **2023**, *15*, 278. https://doi.org/10.3390/fi15080278

Academic Editor: Izzat Alsmadi

Received: 20 July 2023
Revised: 11 August 2023
Accepted: 17 August 2023
Published: 19 August 2023

1. Introduction

Software defined networking, also known as SDN, is a novel approach to the networking paradigm which separates control decisions from the forwarding hardware. The primary objective is to make it as simple as possible for software developers to rely on the resources provided by the network for storage and computation [1]. The SDN comprises switches that support open-flow, a controller, and a secure channel for the controller and the switches [2]. SDN focuses on four main features [3]:

- Separation of the data plane from the control plane.
- A centralized management system and network perspective.
- Open connections between the devices in the control plane and the data plane.
- The network can be programmed by an outside administration.

SDN has two main assets, which are the centralization of control and the ability to control the whole network through software. Those two assets are attractive features for

attackers. Thus, several security challenges affect the SDN, including the distributed denial of service attack (DDoS), man-in-the-middle attack, side channel attack, application manipulation, diversion of traffic, application exploitation, traffic sniffing, password guessing or brute force, and network manipulation [4]. Recently, the DDoS attack has become one of the most serious attacks due to the inability to access the controller. The process and communication capacity of the controller are overloaded when DDoS attacks occur against the SDN controller because of the unnecessary flow produced by the controller for the attack packets. The capacity of the switch flow table becomes full, leading the network performance to decline to a critical threshold [5]. Machine learning (ML) and powerful deep learning (DL) are two of the most common techniques to protect any network from DoS/DDoS attacks.

This work proposes a novel model of DL-based DDoS attack detection algorithms in SDN, evaluates those efforts, and then compares those findings to the recent related papers. The motivation of using a proposed model to find and stop DDoS attacks on SDN is to give an overview of the research studies that have already been conducted in this area and point out the strengths and weaknesses of each of those approaches. Also, DDoS attacks are a big problem for SDN networks. Traditional methods of defense may not be able to find and stop these attacks, because attackers now use new methods to flood SDN using different types of traffic (high and low rates), that slow down the SDN controller and make it inaccessible to legitimate users. Additionally, many recent DDoS detection methods have a high potential for producing false alarms, which can be time-consuming to analyze and cause alert fatigue. Consequently, techniques that can lessen false positives as well as increase the accuracy of DDoS detection are required.

This study proposes a model-based CNN-LSTM as a stacked autoencoder with a checkpoint network for DDoS detection to achieve high accuracy DDoS detection. We have demonstrated how this particular structure can enhance performance, accurately estimate attacks, and remarkably suppress false alarms. Additionally, we provide details of the dataset and hyperparameter values. Furthermore, we produce a comparative analysis of the proposed approach against some recently published work. The main contributions of this work are as follows:

- Propose a deep-stacked autoencoder-based CNN-LSTM for detecting DDoS attacks on a network. This model can extract features effectively in an unsupervised learning approach.
- Utilize a checkpoint network model: a fault tolerance strategy for long-running processes that permits the definition of checkpoints for the model weights at certain locations and improves inference accuracy in real time.

After this introductory section of the paper, Section 2 provides a background of DDoS attacks and the detection mechanism in the SDN. Section 3 presents an overview of the most recent related studies for DDoS detection. The proposed system structure appears in Section 4, and the experimental results of the proposed model classifiers for DDoS attack detection in SDN appear in Section 5, along with a comparison to some relevant research. The last topic of discussion in the paper is the conclusion in Section 6.

2. Concept of SDN and the Detection Mechanism of DoS/DDoS Attacks

The development of technology to detect and mitigate distributed denial of service attacks in SDN environments [6] provides a significant obstacle to these attacks. A distributed denial of service attack sends many packets to the target network. Unmatched flows are considered new if the target and source IP addresses of the forwarded packets are fake, and switches cannot locate these packets in their flow table entries. Next, the switch will forward the packet directly to the SDN controller or send the mismatched packet to the SDN controller [7]. Finding the appropriate routes for these packets lies within the purview of the SDN controller. Many disguised DDoS flows are in legitimate traffic. These flows continually consume the controller's resources, and as a result, those resources eventually become unavailable for use by incoming packets. As a direct consequence of this attack, the SDN controller goes offline, which causes the entire network to enter a downstate. Even if

a backup controller is available, this security flaw still exists [8]. The characteristics of a DDoS attack in a software-defined networking system are subtly distinct from those of an attack on a traditional network. The following is a conclusion reached after researching the DDoS attack techniques utilized against the SDN controller [9]:

- In traditional networks, there are one or more network links, and DDoS attackers go after servers that are the endpoints. In SDN, the controller is hit with a DDoS attack. In SDN, the main goal of a DDoS attack is to make the controller's resources unavailable by failing at a single point.
- The IP addresses of packets in traditional networks are real. As a result, DDoS attackers typically target the terminal server. To conduct a DDoS assault in SDN, the attacker attempts to counterfeit the IP addresses of the destination, involving the controller in constant processing with fresh flows. The controller's resources are made unavailable.
- In traditional networks, when a DDoS attack occurs, the server stops providing services to actual users. But in SDN, if the DDoS attack occurs, the controller in the SDN loses contact with the data plane and cannot provide services for moving data packets.

Traditional ways to find DDoS attacks use a stochastic analysis and the randomness of network traffic to find unusual intrusions. When the detection software finds an attack event, traffic rate-limiting and filtering are used to lessen the damage. But if it uses mitigation strategies carelessly, they will affect legitimate traffic. Even though the victim is not receiving a lot of traffic, a poor response like this can make it difficult for regular users to get online. So, the detection technique must be capable of determining when a DDoS attack occurs and distinguishing between attack traffic and normal traffic.

The current trend in DDoS detection is to use machine learning to classify and detect malicious traffic. These techniques can learn the attributes of the underlying data smartly without needing to be told what is normal and what is dangerous. Even though machine learning-based techniques show promise, most focus on offline traffic analysis and have trouble staying current with how DDoS attacks change over time [10]. Lastly, the detection method should try to reduce false alarms, which can hurt sources that are not doing anything wrong. So, the defense system stops attack traffic and ensures that legitimate traffic gets to the end users reliably [11].

3. Related Works

Recent DDoS detection research utilizing machine learning approaches has achieved promising results. These systems can intelligently understand the underlying data properties without explicitly specifying normal and harmful behaviors, bypassing the limits of conventional detection schemes. The DDoS detection problem is a binary classification problem in which the observed traffic is either normal or attack traffic. Moreover, detection techniques have used deep learning more often in recent years to find DDoS attacks and presented several approaches. Of various recent notable works in this field, some utilized convolutional networks, some utilized recurrent neural networks, especially LSTM and bidirectional LSTM, and some used autoencoder, an unsupervised learning approach, to discover non-linear characterizations from input data, and would then perform a classification algorithm to differentiate malicious traffic from genuine traffic.

In 2017, Yuan, Li, and Li [12] developed a deep learning algorithm named "DeepDefense", a model that uses a deep learning model to detect DDoS attacks. To carry out their research, they used CNN, as well as several distinct variants of RNN (such as LSTM and the gated recurrent unit neural network (GRUNN)), and the random forest (RF) method. The study included a comparison analysis between several deep learning methodologies and between deep learning and machine learning algorithms (it selected RF). DeepDefense put into action four deep learning models: LSTM, CNN-LSTM, GRU, and 3-LSTIM. We compared the results of these models with one another. With an accuracy of 98.410% and an area under the curve (AUC) score of 99.450%, the top deep learning model, 3LSTM, was able to identify DDoS attacks. Shone et al. [13] found that stacking two autoencoders allowed for the learning of more complex feature-based correlations. For intrusion detec-

tion, they combined the stacked autoencoder with a random forest classifier. They asserted that the soft-max layer was less effective than traditional classifiers. In 2019, Pektaş and Acarman [14] presented a model-based deep learning method that utilized CNN and LSTM to train the spatial-temporal characteristics of network flows. It used two datasets for training and testing: the ISCX2012 dataset [15] and CICIDS2017 [16]. The results show that the model achieved 0.9669 in precision, 0.9649 in recall, 0.9657 in F1-score, and 0.9666 in accuracy. The model also returned good results when using CI-CIDS2017, where it achieved 0.9797 in precision, 0.9765 in recall, 0.9780 in F1-score, and 0.9772 in accuracy.

Some studies developed hybrid deep learning models. Gadze et al., 2021 [17] proposed a model that combined two types of deep learning, LSTM and CNN, to detect an attack. Mininet generated the dataset dynamically and utilized OpenFlow switches and Floodlight as an external controller. Based on the findings, RNN LSTM outperformed linear-based models like SVM (86.85%) and Naive Bayes (82.61%), achieving an accuracy of 89.63% compared to their respective scores. Their model had an accuracy of 99.4%, while the KNN technique, based on linear models, had an even higher accuracy. In addition, the model functioned most effectively when it split the data in a 70/30 train/test split ratio. Singh and Jang-Jaccard (2022) [18] created a hybrid autoencoder model dubbed MSCNN-LSTM-AE. This model found anomalies in network traffic by utilizing a combination of a multi-scale convolutional neural network (MSCNN) and LSTM. The MSCNN autoencoder was employed initially to evaluate the spatial characteristics of the dataset. Next, it used an LSTM-based autoencoder network to identify the temporal features of the latent space features learned from the MSCNN-AE. The authors analyzed their work with the UNSW-NB15 [19], NSL-KDD [20], and CICDDoS2019 tests. The accuracy score for their model (MSCNN-LSTM-AE) came in at 93.76%, while the recall score was 92.26%. Elubeyd and Yiltas-Kaplan [21] presented a hybrid deep learning approach for detecting and countering DoS/DDoS attacks in SDNs. The selection of a hybrid model that included a 1D CNN, a dense neural network (DNN), and a gated recurrent unit (GRU) took advantage of their individual strengths that synergistically addressed the intricacies of the problem. The model achieved good results when using CICDDoS 2019, where it achieved 0.9981 in accuracy, 0.9996 in precision, 0.999 in recall, and 0.9993 in F1-score.

Some recent studies used a stacked autoencoder to improve DDoS detection accuracy. Yaser et al., 2022 [22] proposed a novel approach for detecting DDoS attacks, which involved integrating deep learning with feedforward neural networks in the form of autoencoders. The training and evaluation of the model were analyzed using two datasets, initially through a static approach and subsequently through an iterative technique. They developed the autoencoding model through a layer-by-layer stacking of the input layer and the hidden layer of self-encoding models, wherein each self-encoding model employed a hidden layer. They assessed the performance of their model by employing a three-fold data partitioning strategy comprising training, testing, and validating subsets. The test result showed that the model yielded superior accuracy for the static dataset. Specifically, for the ISCXIDS-2012 dataset, the model attained a maximum accuracy of 99.35% during training, 99.3% during validation, 99.78% for precision, 99.99% for recall, and 99.87 for F1-score. The UNSW-2018 dataset exhibited high levels of accuracy during training, with values of 99.95% for training and 99.94 for validation, and 99.99 for recall, precision, and F1-score. Jiang et al., 2018 [23], presented a new method (DLGraph) for detecting malware based on deep learning along with graph embedding. Their architecture for deep learning was comprised of two stacked denoising autoencoders (SDA). One SDA was able to learn the latent structure of function-call graphs in programs. The other SDA was capable of learning a latent representation of Windows API calls made by programs. They utilized the node2vec technique when incorporating a function-call graph in a feature space. The experimental results on three distinct datasets demonstrated that the proposed DLGraph method obtained high levels of accuracy and exceeded the closely related DL4MD method, where it gained 99.14% in accuracy for dataset 1, 99.36% for dataset 2, and 99.31 for dataset 3. Table 1 shows the comparison between these related works.

Table 1. Comparison of related works in terms of methods, performance measures, and achievement.

Ref	Model	Achievement
Yuan, Li [12]	LSTM, GRU, CNN-LSTM, and 3-LSTIM	3-LSTIM outperformed the other models which gained 99.450% in accuracy
Shone et al. [13]	combined the stacked autoencoder with a random forest classifier	They asserted that the soft-max layer was less effective than traditional classifiers
Pektaş and Acarman [14]	LSTM and CNN	For the ISCX2012 dataset, the model achieved 0.9669 in precision, 0.9649 in recall, 0.9657 in F1-score, and 0.9666 in accuracy. For CI-CIDS2017, the model achieved 0.9797 in precision, 0.9765 in recall, 0.9780 in F1-score, and 0.9772 in accuracy
Gadze et al., 2021 [17]	LSTM and CNN	The model outperformed the other ML models, in which it gained 99.4% in accuracy compared with RNN LSTM that archived 89.63%, SVM achieved 86.85%, and Naive Bayes achieved 82.61%
Singh and Jang-Jaccard, 2022 [18]	Hybrid autoencoder model dubbed MSCNN-LSTM-AE	The accuracy score was 93.76% and the recall score was 92.26%
Elubeyd and Yiltas-Kaplan [21]	Hybrid deep learning approaches (1D CNN, a dense neural network (DNN), and a gated recurrent unit (GRU))	The model achieved good results when using CICDDoS 2019, where it achieved 0.9981 in accuracy, 0.9996 in precision, 0.999 in recall, and 0.9993 in F1-score
Yaser et al., 2022 [22]	LSTM-Autoencoder	For the ISCXIDS-2012 dataset, the model attained a maximum accuracy of 99.35% during training, 99.3% during validation, 99.78% for precision, 99.99% for recall, and 99.87 for F1-score. For UNSW-2018, it gained 99.95% for training accuracy and 99.94 for validation accuracy, and 99.99 for recall, precision, and F1-score
Jiang et al., 2018 [23]	DLGraph based on two stacked denoising autoencoders (SDA)	It gained 99.14% in accuracy for dataset 1, 99.36% for dataset 2, and 99.31 for dataset 3

4. Proposed Model Structure

Our approach uses autoencoders, a method that is now popular in deep learning. An autoencoder is an unsupervised neural network-based feature extraction method that learns the best feasible factors to reproduce faithfully its output given some input. One of its many appealing features is its potential to provide a non-linear and more efficient generalization than the principal component analysis (PCA). It achieves this result by backpropagation with input-equivalent target values. To rephrase, it tries to figure out how to predict the occurrence of itself as closely as possible. The typical architecture of an autoencoder consists of three layers: an input layer, an output layer, and a hidden layer. The hidden layer's dimensions are lower than that of the input [24]. Figure 1 shows the traditional (single) autoencoders.

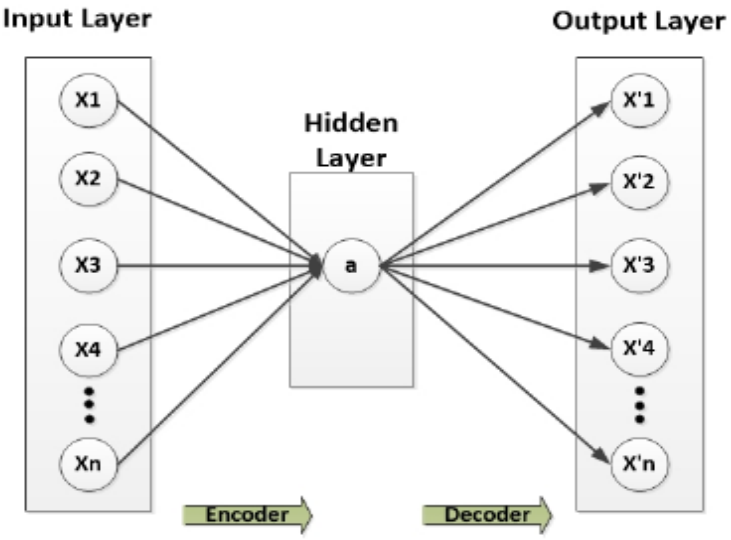


Figure 1. Single autoencoder [24].

In the proposed method, we utilize a deep autoencoder. Unlike traditional autoencoders, deep autoencoders consist of two typical deep-belief networks, one for encoding and one for decoding, with four or five shallow layers each. Deep learning can be applied to autoencoders by a stacked autoencoder, in which many hidden layers build depth, and the hidden layers reflect fundamental concepts. As a result of this increased depth, computing costs will reduce, the amount of instruction data required will decrease, and accuracy will improve. The output of one buried layer serves as the input to a later, more advanced step. First-order features are often learned from unprocessed data by the first layer of a stacked autoencoder. Second-order features based on trends in the presence of first-order traits are typically learned by the second layer. Subsequent layers build our understanding of higher-order characteristics. Figure 2 shows the structure of the proposed deep autoencoder model.

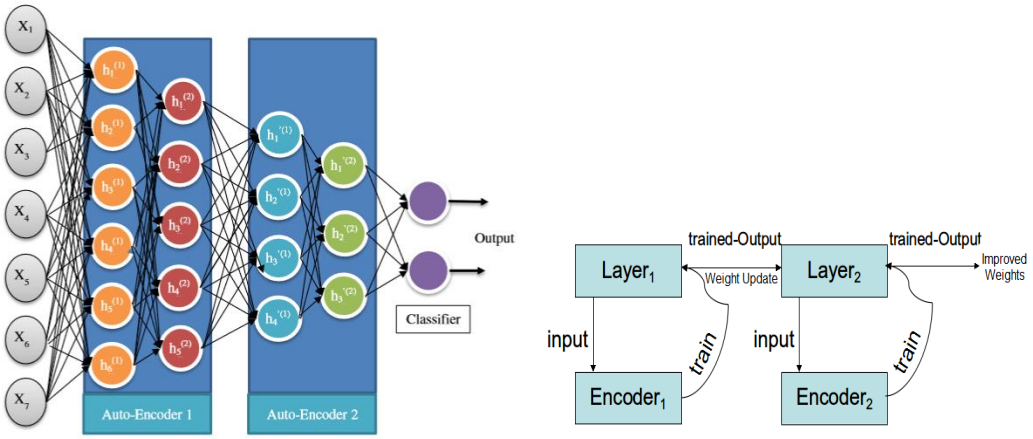


Figure 2. The general structure of the proposed deep autoencoder.

The first layer is the input layer, which receives input Xi and uses numerous hidden layers to encode and decode it (encoder and decoder blocks). The encoding process compresses the attributes to make them smaller than the input data, and the decoding process restores these attributes in reverse order to begin the final output at the deepest layer. When processed, the output feature vector Xi is virtually identical to the input. The convolutional layer and LSTM are combined with an autoencoder to generate a robust DDoS attack classifier. LSTM is excellent at understanding the context of Internet packets, identifying long- and short-term dependencies, and identifying trends in DDoS attack sequences. LSTM is particularly proficient at categorizing processes such as time series and learning from experience. After the encoding is complete, based on the output result of the hidden layer, the output layer is decoded and reconstructed according to Equation (2) to produce an output of the same size as the input layer neuron.

The purpose of the autoencoder section is to map input $x \in [0,1]^d$ to a latent representation $y \in [0,1]d'$, where the mapping is performed by the function

$$yi = s(Wx_i + b) \tag{1}$$

Through

$$z_i = s(W'yi + b') \tag{2}$$

This concealed representation is mapped back into a reconstruction of the same shape as input x. Here, s represents a non-linear function, such as the sigmoid function. The first component is the encoder, while the second is the decoder. This model's parameters minimize the average reconstruction error.

The model consists of one input layer, one convolutional layer (Conv1D), two LSTM layers, one max pooling layer, and one dense layer in output. Figure 3 shows the training model with the proposed deep autoencoder scheme.

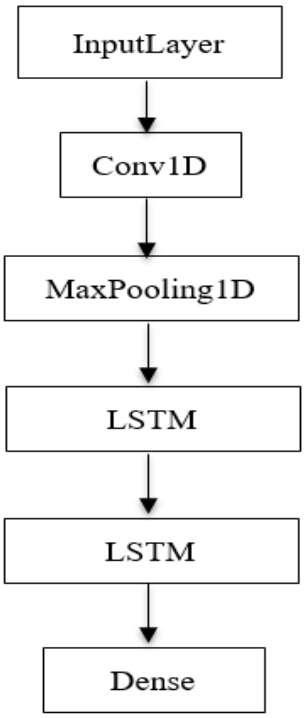


Figure 3. The training model of the proposed deep autoencoder.

In addition, the checkpoint network improves the weights. Checkpointing is a crucial functionality that expedites failure recovery, reducing the total training time and ensuring continuous progress. Checkpoints are periodic captures of the current state of a running process, which are then stored in a durable storage medium. The individual loads the most recent checkpoint to recover from a setback and recommence training. In addition to the imperative of failure recovery, the utilization of checkpoints is necessary for transferring training processes across various nodes or clusters. This transition may be necessary for server maintenance (for instance, urgent security updates that cannot be delayed), hardware malfunctions, network complications, and the optimization or reallocation of resources. Another significant application of checkpoints involves the real-time publication of snaps of trained models to enhance the accuracy of inference, commonly referred to as online training. For example, we can employ an interim model obtained by checkpointing for prediction serving. This method allows the model to continue training on more recent datasets, ensuring the freshness of the inference model. We can also utilize checkpoints for transfer learning, a technique where an intermediate structure state is an initial point for training toward a distinct objective [10]. Figure 4 shows the training loop with a checkpoint network.

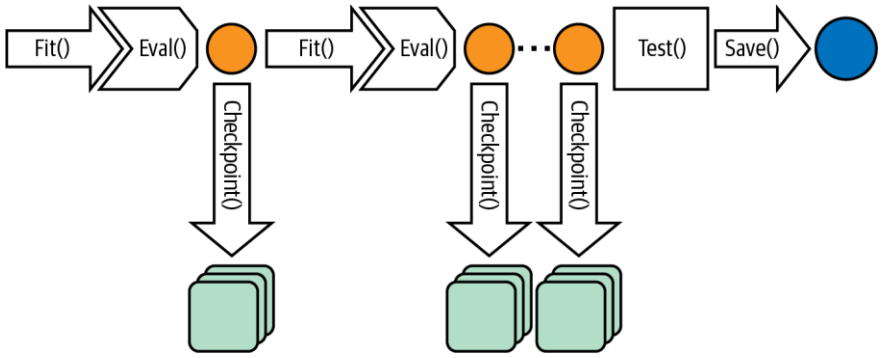


Figure 4. The training looping with checkpoint network [10].

The evaluation of the model occurs at the conclusion of each epoch, and the weights corresponding to the highest accuracy and lowest loss during that specific epoch are retained and saved. In the event that the weights in the model during a specific epoch fail to yield the optimal accuracy or loss, as determined by the user-defined criteria, the weights will not be preserved. However, the training process will persist, commencing from the aforementioned condition.

5. Results and Discussion

This section discusses the experimental results of our proposed method. We use several performance metrics for evaluation. We use two datasets to test the performance of the proposed model in detecting DDoS attacks. Then, we compare these results with some recent related works using the same datasets and with some other machine learning algorithms.

5.1. Datasets

Most datasets are imperfect, and the row samples employed to cover the application manners are insufficient in these cases. The most common DDoS datasets involve CI-CIDS2019, CICIDS2017, KDDCUP99, ISCX2012, Kyoto 2006+, and NSL-KDD, which researchers have significantly utilized for intrusion detection. In this work, we chose two datasets to validate our proposed DDoS classifier, which are:

1- DDoS attack SDN dataset (Mendeley Data): This set is an SDN-specific dataset created by the Mininet emulator and utilized by machine learning and deep learning algorithms for traffic classification. The project begins by constructing 10 Mininet topologies with switches connected to a single Ryu controller. It simulates a network for benign TCP, UDP, and ICMP traffic and malicious traffic, which consists of a TCP Syn attack, UDP Flood attack, and ICMP assault. The data collection contains 23 features, of which some are from the switches, and others are calculated, such as the Packet count, Switch-id, duration sec, byte count, Destination IP, Source IP, Port number, etc.

2- Botnet dataset (UNSW_2018_IoT_Botnet): Even though several datasets have been proposed for detecting intrusions, most datasets are not updated and do not reflect actual data. The Canadian Institute for Cybersecurity addressed these issues by developing the Intrusion Detection Evaluation Dataset, ISCX-IDS 2012, and [25] generated by monitoring network activity for seven days. The labeled dataset consists of approximately 1,512,000,000 packets with 20 features. The primary characteristics of this dataset are discussed in [25] and include real, normal, and malicious streams comprising FTP, HTTP, IMAP, POP3, SMTP, and SSH protocols collected using real devices. All data are categorized and marked. The collected datasets contain a variety of intrusion kinds (Infiltrating, DoS, DDoS, and Brute Force SSH).

5.2. Performance Evaluation Metrics

A performance evaluation is the process of measuring the of a classification model after assigning cases to their various predetermined labels. The performance evaluation considers measures including accuracy, recall, precision, F1-score, and confusion matrix [26]. These metrics are described as follows:

1. Accuracy: The ratio of all correct predictions over the total number of packets in the dataset [24]:

$$\text{Accuracy(Ac)} = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \quad (3)$$

where T_p is the "True Positive," which describes the rate where the actual instance of a certain label is categorized as that label; T_n is the "True Negative," which is the attack instance's value that is classified as an attack; F_p is the "False Positive," which is the number incorrectly classified for a certain class label, i.e., the instance categorized

value as an additional class label for a given dataset; F_n is the "False Negative," which is the value of normal traffic which is classified as an attack.

2. Recall: Recall is the number of correctly predicted positive records over all the positive records: a metric that can detect DDoS attack traffic compared to normal traffic [24].

$$\text{Recall} = \frac{T_p}{T_p + F_n} \tag{4}$$

3. Precision: Precision is the proportion of actual positive instances that were correctly predicted, i.e., it is a metric that can detect DDoS attack traffic among normal traffic [24].

$$\text{Precision} = \frac{T_p}{T_p + F_p} \tag{5}$$

4. F1-score: The F1-score is the balance between the recall and the precision [24].

$$\text{F1} - \text{score} \ = 2 \times \frac{\text{Recall} \times \text{precision}}{\text{Recall} + \text{precision}} \tag{6}$$

5. Confusion Matrix: The data classification results appear in a table format. The accuracy of a classification model is evaluated by applying it to test data for which the results have already been determined. The study uses it to show the distribution of the expected outcomes, despite its poor suitability for anything beyond binary classification [24].

5.3. Results and Discussion of Base Classifier Models

5.3.1. Mendeley DDoS Attack SDN Dataset

The dataset utilized in this study is a software defined networking (SDN) dataset generated by implementing ten distinct topologies within the Mininet framework, wherein switches are interconnected to a singular Ryu controller. The network simulation encompasses benign traffic, including TCP, UDP, and ICMP, as well as malicious traffic, which comprises TCP Syn, UDP Flood, and ICMP attacks. The dataset contains 23 features, including extracted data from switches and calculated variables. At first, we extract packet fields from the DDoS attack SDN dataset. The number of samples used is 100,000. The extracted feature appears in Figure 5.

```
dt              byteperflow
switch          pktrate
src             Pairflow
dst             Protocol
pktcount        port_no
bytecount       tx_bytes
dur             rx_bytes
dur_nsec        tx_kbps
tot_dur         rx_kbps
flows           tot_kbps
packetins       label
pktperflow      dtype: int64
```

Figure 5. Extracted feature from the original packets.

The extracted features are Packet_count, Switch-id, byte_count, duration_sec (representing the duration in seconds), duration_nsec (representing the duration in nanoseconds), and the overall duration obtained by summing duration_sec and duration_nsec. Additionally, the characteristics contain Source IP and Destination IP. The numerical identifier assigned to a specific communication endpoint within a computer network is commonly referred to as a port number. The variable "tx_bytes" represents the quantity of bytes that have been moved via the switch port, whereas "rx_bytes" denotes the quantity of bytes that have been received on the switch port. The "dt" field represents the numerical

representation of both the date and time. This field is utilized to monitor the flow of a particular process at regular intervals of 30 s. The calculated features encompass the term "packet per flow", which refers to the count of packets transmitted during a single flow. Similarly, "byte per flow" represents the count of bytes transmitted during a single flow. "Packet Rate" denotes the number of packets sent per second and may be computed by splitting the packet for each flow by the monitoring interval. Additionally, the number of "Packet_ins" messages and the total flow inputs in the switch are relevant factors in this context. The variables tx_kbps and rx_kbps represent the rates at which data are transferred and received, respectively. Port bandwidth refers to the cumulative value of both the transmitted kilobits per second (tx_kbps) and received kilobits per second (rx_kbps).

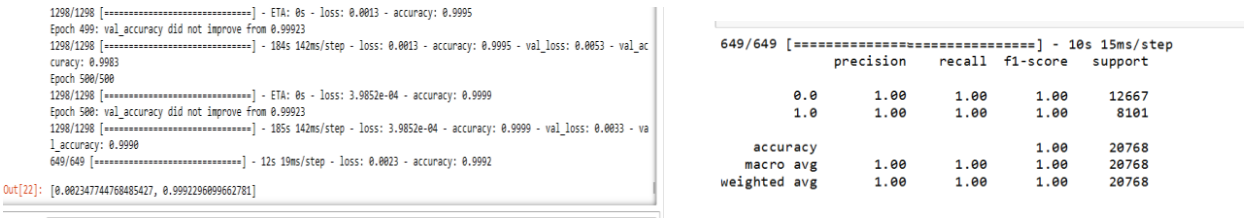
The final column denotes the class label, which serves as an indicator to determine if the traffic class is normal or malicious. In the classification scheme utilized, benign traffic is assigned a label of 0, whereas malicious traffic is assigned a label of 1. A network simulation is conducted over a duration of 250 min, resulting in the collection of 104,345 rows of data. The simulation is executed repeatedly within a specified time frame, allowing for the accumulation of further data.

We split the dataset into 80% for training and 20% for testing. Attack traffic is labeled with 1, whereas the normal traffic is labeled with 0, and we train the model for 500 epochs to study the effect of the checkpoint strategy on improving classification accuracy. The classification result appears in Table 2 and Figures 6 and 7.

Table 2. Results of tests using the Mendeley dataset of the proposed model.

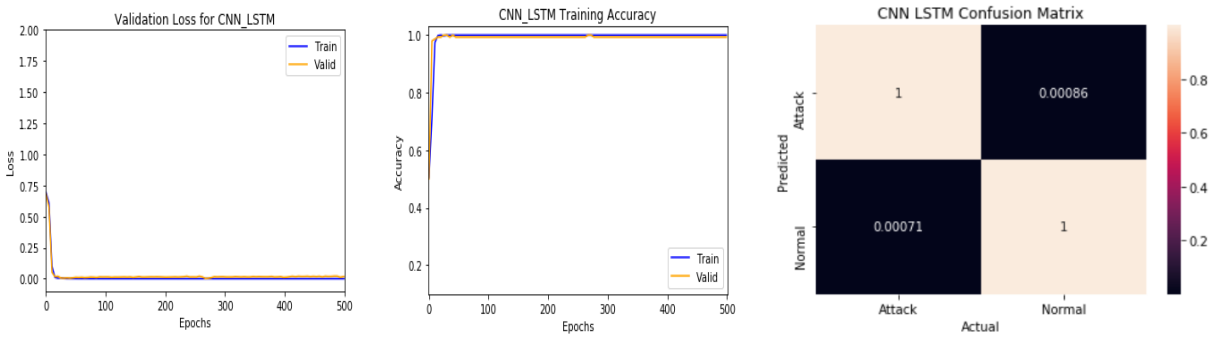
Accuracy (%)	Validation Accuracy (%)	Precision (%)		Recall (%)		F1-Score (%)	
		Normal	Attack	Normal	Attack	Normal	Attack
99.99	99.923	100	100	100	100	100	100

where Normal is the normal traffic, and Attack is DDoS attack traffic.



(**a**) (**b**)

Figure 6. A screenshot of the output terminal (**a**) training and validation accuracy results; (**b**) metrics results.



(**a**) (**b**) (**c**)

Figure 7. Classification results of the CNN-LSTM-autoencoder model. (**a**) Accuracy results of training and validation per epoch, (**b**) loss per epoch, and (**c**) confusion matrix.

Table 2 and Figure 6 show the final results of training and Figure 7a, b represent the training/test loss and training/test accuracy after 500 epochs, respectively. Figure 7c shows the confusion matrix of the proposed model. These results prove that the model achieves very high classification results and stability with close results between training and validation, where it gains 99.99% in training accuracy, 99.923% in validation accuracy, and gain 100% in precision, recall, and F1-score. From Figure 7a, the model training and validation loss are very low, about 3.98×10^{-4} for training and 3.3×10^{-3} for validation. From Figure 7b, the model reaches the best train/validation accuracy at epoch (19). From Figure 7c, the model achieves a significant degree of prediction accuracy. It reaches an accuracy of about 100% in correctly detecting attacks and normal traffic flows. Moreover, the proposed model has fewer false alarms since it shows a False Positive Rate (FPR) of 0.00086 and a False Negative Rate (FNR) of 0.00071. These results show the importance of using a checkpoint network and many epochs, which can highly affect accuracy, as shown in Figure 6.

5.3.2. UNSW_2018_IoT_Botnet Dataset

The Bot-IoT dataset was developed in 2018 and published in 2019 by the New South Wales University (UNSW). It is a contemporary and authentic dataset for training machine learning models to effectively identify and mitigate Botnet attacks within Internet of Things (IoT) networks. The dataset comprises 72 million instances, consisting of three dependent and forty-three independent features. The dataset encompasses various cyber-attacks, such as OS and Service Scan, DoS, DDoS, Data exfiltration, and Keylogging. Additionally, the DoS and DDoS attacks are further categorized based on the specific protocol. At first, we extract packet fields from the DDoS attack SDN dataset. The number of samples used is 100000. The extracted feature appears in Figure 8.

```
pkSeqID
proto
saddr
sport
daddr
dport
seq
stddev
N_IN_Conn_P_SrcIP
min
state_number
mean
N_IN_Conn_P_DstIP
drate
srate
max
attack
category
subcategory
dtype: int64
UDP                  1584650
TCP                  1274843
Service_Scan           58626
OS_Fingerprint         14293
HTTP                    1970
Normal                   370
Keylogging                59
Data_Exfiltration          6
Name: subcategory, dtype: int64
```

Figure 8. Extracted feature from the original UNSW 2018 dataset packets.

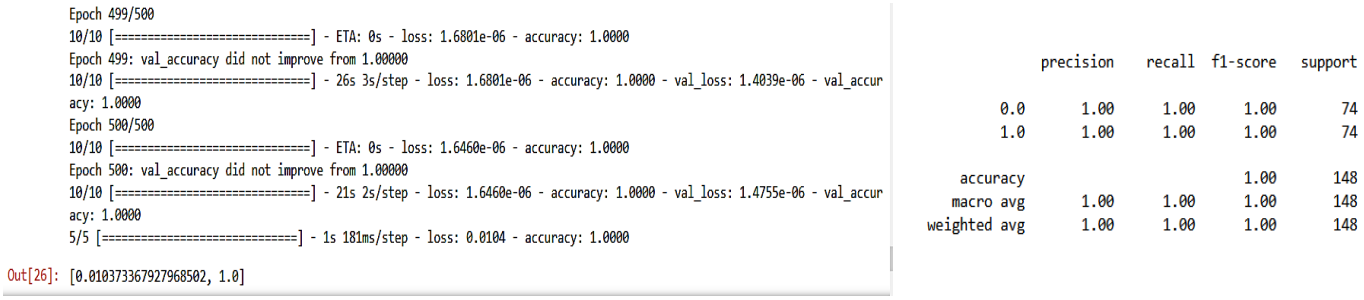
The UNSW 2018 dataset involves pkSeqID to represent the row identifier, Proto is the representation of textual for transaction protocols that are resent in the network flow, saddr is the IP address of the source, sport is the port number of the source, daddr is the IP address of the destination, dport is the port number of the destination, seq is the argus sequence number, stddev is the aggregated records of the standard deviation, min is the minimum duration of the standard deviation, state number represents the feature state numerical representation, mean is the aggregated records of the average deviation, drate is the packets per second of destination-to-source, srate is the packets per second of source-to-destination, max is the aggregated records' maximum duration, attack is the class label where 0 represents normal traffic and 1 represents the attack traffic, category is the category of traffic, subcategory is the subcategory of traffic, and dbytes is the byte count of destination-to-source.

Hence, like the previous dataset, we split the data into 80% for training and 20% for testing. Attack traffic is labeled 1, the normal traffic is labeled 0, and we train the model for 500 epochs. The classification results appear in Table 3 and Figures 9 and 10.

Table 3. Results of tests using the UNSW 2018 dataset of the proposed model.

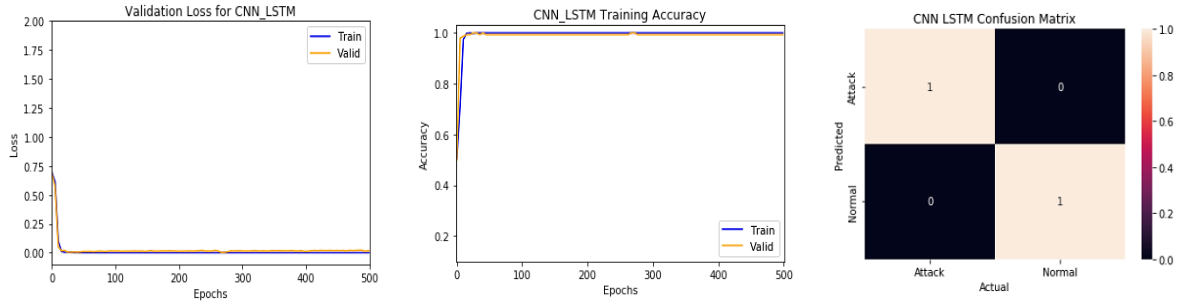
Accuracy (%)	Validation Accuracy (%)	Precision (%)		Recall (%)		F1-Score (%)	
		Normal	Attack	Normal	Attack	Normal	Attack
100	100	100	100	100	100	100	100

where Normal is the normal traffic, and Attack is DDoS attack traffic.



(**a**) (**b**)

Figure 9. Screenshot of the output terminal showing (**a**) training and validation accuracy results; (**b**) metrics results.



(**a**) (**b**) (**c**)

Figure 10. Classification results of the CNN-LSTM-autoencoder model. (**a**) Accuracy results of training and validation per epoch, (**b**) loss per epoch, and (**c**) confusion matrix.

Table 3 and Figure 9 show the final results of training and Figure 10a,b represent the training/test loss and training/test accuracy after 500 epochs, respectively. Figure 10c shows the confusion matrix of the proposed model. These results prove that the model achieves very high classification results and stability with close results between training and validation, where it gains 100% in all metrics. From Figure 10a, the model training and validation loss are very low, about 3.98×10^{-4} for training and 3.3×10^{-3} for validation. From Figure 10b, the model reaches the best train/validation accuracy at epoch 19, which shows the importance of using a checkpoint network and many epochs, which can significantly affect accuracy, as shown in Figure 11.

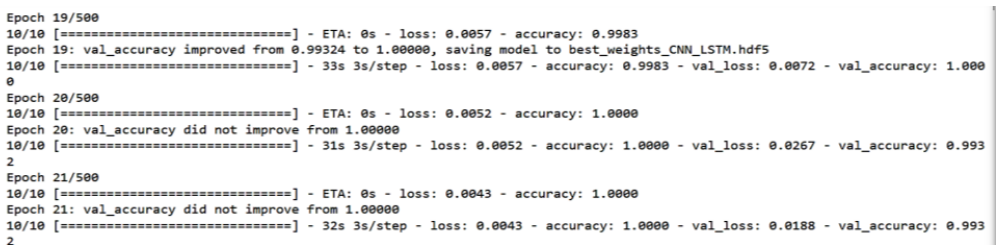
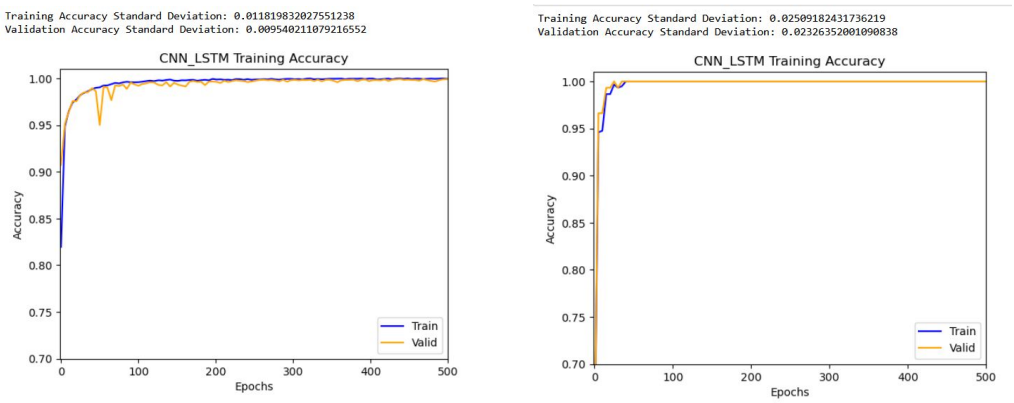


Figure 11. Best accuracy results of training and validation at epoch 19.

For further checking, we utilize the standard deviation metrics to quantify the extent to which the attribute value of a feature deviates from its mean value. The standard deviation categorization aids in the identification of features that deviate from the average value by highlighting values that are both above and below the mean. Figure 12 shows the standard deviation result for the proposed model for two datasets.



(**a**) (**b**)

Figure 12. The standard deviation metrics for proposed model using (**a**) Mendeley dataset and (**b**) UNSW 2018 dataset.

As shown in Figure 12 and Table 4, the proposed model has a lower variance. As a result, the proposed approach outperforms in terms of accuracy and reliability, and the learning curves are smoother, indicating that the proposed model is consistent. As a result, it is not only more accurate, but it is also more robust and consistent.

Table 4. Results of tests using the UNSW 2018 dataset of the proposed model.

Dataset	Accuracy (%)	Standard Deviation (%)	Validation Accuracy (%)	Standard Deviation (%)
Mendeley	99.99	0.0118198	99.923	0.000954
UNSW 2018	100	0.9250918	100	0.023263

5.3.3. Comparison of Results with Some Machine Learning and Deep Learning Algorithms

This section compares the proposed CNN-LSTM-autoencoder model with the LSTM model and three other ML algorithms includes K-nearest neighbors algorithm (KNN), SVM, and XGBoost. We use the same datasets and number of epochs to determine the difference in performance between the proposed model and these two ML models. The results appear in Table 5.

Table 5. Results of tests using the Mendeley dataset for ML models.

Model	Accuracy (%)	Val. Accuracy (%)	Precision (%)		Recall (%)		F1-Score (%)	
			Normal	Attack	Normal	Attack	Normal	Attack
LSTM	92.6	79.433	80	80	6	62	11	70
KNN	-	-	97	95	97	95	97	95
SVM	-	95	98	91	94	96	96	94
XGBoost	99.55	99.54	-	-	-	-	-	-

where Normal is the normal traffic, and Attack is DDoS attack traffic.

From Table 5, the results show the lower performance of the LSTM model with the DDoS attack SDN dataset when it gains 92.6% in training accuracy and 79.433% in validation accuracy. It also achieves very low results in recall, 6% in normal and 62% in attack, and the precision is the same in both normal and attack (about 80%). So, the model

gains poor results in the F1-score, 11% for normal, and 70% in attack. Table 5 shows good results for the KNN model with the DDoS attack SDN dataset when it gains 97% for normal and 95% for attack for precision, and for recall, it gains 97% in normal and 95% in attack. The F1-score is then good at 97% for normal and 95% in attack. However, these results are less than the proposed CNN-LSTM-autoencoder model. The SVM also gains good results which are 98% for normal and 91% for attack for precision, and for recall, it gains 94% in normal and 96% in attack. The F1-score is then good at 96% for normal and 94% in attack. The XGBoost achieves higher accuracy and reaches up to 99.54%.

By comparing the deep learning model (LSTM) with the proposed CNN-LSTM-autoencoder model, the proposed model is more accurate and stable than the LSTM model and achieves higher accuracy in lower epochs. Compared with the machine learning model (KNN), the proposed CNN-LSTM-autoencoder model is also more accurate than all ML for the DDoS attack SDN dataset.

5.3.4. Comparison of Results with Published (Base)

We compared the result of the proposed system with some recent related works using the DDoS attack SDN dataset and UNSW2018 BoTIoT (Table 6). The obtained results showed that the model achieves very high classification results. For the UNSW2018 dataset, the proposed model achieves 100% in all metrics and a very low loss, about 3.98×10^{-4} for training and 3.3×10^{-3} for validation. Table 6 proves that our model outperforms Yaser et al. [22] in all metrics using the same dataset (UNSW 2018). Ivanova et al. [27] and Prasad et al. [28] had models that achieved an accuracy of 99.99%, whereas our model achieves 100% accuracy. Our model outperforms their models in all metrics. Although it had high accuracy, they showed lower precision, recall, and F1-score for normal flows. So, our model can accurately detect and recognize normal and abnormal flows since it shows 100% in all metrics.

Table 6. Comparison results between the proposed CNN-LSTM-autoencoder model and some recent works.

Ref.	Dataset	Algorithm	Accuracy (%)	Val. Accuracy (%)	Precision (%)		Recall (%)		F1-Score (%)	
					Normal	Attack	Normal	Attack	Normal	Attack
Proposed model	UNSW2018	CNN-LSTM-autoencoder	100	100	100	100	100	100	100	100
Yaser et al. [22]		LSTM-autoencoder	99.95	99.94	95	99	94	99	95	99
Ivanova et al. [27]		optimized feed-forward neural network	99.99	99.99	82.55	99.99	66.35	99.99	73.57	99.87
Prasad et al. [28]		VMFCVD	99.99	99.99	87.72	99.99	82.55	99.99	81.97	99.99
Proposed model	DDOS attack SDN Dataset (Mendeley dataset)	CNN-LSTM-autoencoder	99.99	99.923	100	100	100	100	100	100
Ahuja et al. [29]		CNN	98.74	-	98.75	98.73	98.9	98.55	98.83	98.64
		LSTM	95.60	-	96.20	94.90	95.64	95.56	95.92	95.23
		CNN-LSTM	99.48	-	99.43	99.55	99.66	99.26	99.54	99.40
		SVC-SOM	95.45	-	96.71	93.75	95.40	95.51	96.05	94.62
		SAE-MLP	99.75	-	99.96	99.69	99.77	99.94	99.87	99.82
Yaser et al. [22]	Generated SDN dataset	LSTM-autoencoder	97.62	97.68	98	88	92	97	95	93

where Normal is the normal traffic, and Attack is DDoS attack traffic.

Meanwhile, for the DDoS attack SDN dataset, our system gains an accuracy of up to 99.99% in training and 99.923% in validation and achieves 100% in precision, recall, and F1-score. Also, the model training and validation losses are very low, about 3.98×10^{-4} for training and 3.3×10^{-3} for validation. Our model outperforms all proposed models by Ahuja et al. [29] in all factors. Experimental results reveal that our proposed model has a high feature extraction ability and high performance in detecting attacks. All performance metrics indicate that the proposed approach is the most appropriate choice to apply to a real-world flow detection environment.

6. Conclusions

Network virtualization imposes new risks and exploitable attacks in addition to those currently on traditional networks. The DDoS attack group is one of the most aggressive attack types in recent years, devastating the entire network infrastructure. To defend against the DDoS attack, within the scope of this project, we developed and deployed a DDoS detection system based on deep learning to detect multi-vector attacks within an SDN environment. The proposed approach has a success rate of 99.99% in train and 99.923% in validation and 100% for all metrics (precision, recall, and F1-score) for identifying individual DDoS attacks in all DDoS datasets. It does so with an accuracy of 100% and an extremely low False-Positive Rate compared to other efforts, and it categorizes the traffic into normal and attack groups. One of our future goals is to test the proposed model as a real-time classifier in an SDN environment under real-time DDoS traffic and normal traffic to address its accuracy and time of detection using an emulator such as Mininet or in a real SDN environment. In addition, our goal is to lessen the strain placed on the controller by putting in place a network intrusion detection system that can identify not only DDoS attacks but also others.

Author Contributions: Conceptualization, A.K.M. and M.N.A.; methodology, A.K.M.; formal analysis, A.K.M. and M.N.A.; investigation, A.K.M.; writing—original draft preparation, A.K.M.; supervision, M.N.A. All authors have read and agreed to the published version of the manuscript.All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data derived from public domain resources.

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

References

1. Urrea, C.; Benítez, D. Software-Defined Networking Solutions, Architecture and Controllers for the Industrial Internet of Things: A Review. *Sensors* **2021**, *21*, 6585. [CrossRef] [PubMed]
2. Nadeau, T.D.; Gray, K. *SDN: Software Defined Networks*; O'Reilly Media: Newton, MA, USA, 2013.
3. Feamster, N.; Rexford, J.; Zegura, E.; Tech, G. The Road to SDN: An Intellectual History of Programmable Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–98. [CrossRef]
4. Pradhan, A.; Mathew, R. Solutions to Vulnerabilities and Threats in Software Defined Networking (SDN). *Procedia Comput. Sci.* **2020**, *171*, 2581–2589. [CrossRef]
5. Silva, F.S.D.; Silva, E.; Neto, E.P.; Lemos, M.; Neto, A.J.V.; Esposito, F. A Taxonomy of DDoS Attack Mitigation Approaches Featured by SDN Technologies in IoT Scenarios. *Sensors* **2020**, *20*, 3078. [CrossRef] [PubMed]
6. Abdulkarem, H.S.; Alethawy, A.D. DDoS attack detection and mitigation at SDN enviroment. *Iraqi J. Inf. Commun. Technol.* **2021**, *4*, 1–9. [CrossRef]
7. Tan, L.; Pan, Y.; Wu, J.; Zhou, J.; Jiang, H.; Deng, Y. A New Framework for DDoS Attack Detection and Defense in SDN Environment. *IEEE Access* **2020**, *8*, 161908–161919. [CrossRef]
8. Choudhary, A.R.; Associates, C. OpenFlow switch controller as a policy-based system. *Issues Inf. Syst.* **2021**, *22*, 320–334. [CrossRef]
9. Wang, T.; Chen, H.; Cheng, G.; Lu, Y. SDNManager: A Safeguard Architecture for SDN DoS Attacks Based on Bandwidth Prediction. *Secur. Commun. Netw.* **2018**, *2018*, 7545079. [CrossRef]
10. Lakshmanan, V.; Robinson, S.; Munn, M. *Machine Learning Design Patterns*; O'Reilly Media, Inc.: Newton, MA, USA, 2020; Chapter 4; ISBN 9781098115784.
11. Doshi, K.; Yilmaz, Y.; Uludag, S. Timely Detection and Mitigation of Stealthy DDoS Attacks via IoT Networks. *arXiv* **2020**, arXiv:abs/2006.08064. Available online: http://arxiv.org/abs/2006.08064 (accessed on 1 May 2023). [CrossRef]
12. Yuan, X.; Li, C.; Li, X. DeepDefense: Identifying DDoS Attack via Deep Learning. In Proceedings of the 2017 IEEE International Conference on Smart Computing, SMARTCOMP, Hong Kong, China, 29–31 May 2017; pp. 1–8. [CrossRef]
13. Shone, N.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A Deep Learning Approach to Network Intrusion Detection. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *2*, 41–50. [CrossRef]
14. Pektaş, A.; Acarman, T. A deep learning method to detect network intrusion through flow-based features. *Int. J. Netw. Manag.* **2018**, *29*, e2050. [CrossRef]
15. IDS 2012 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids.html (accessed on 11 December 2022).

16. IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 11 December 2022).
17. 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. [CrossRef]
18. Singh, A.; Jang-Jaccard, J. Autoencoder-based Unsupervised Intrusion Detection using Multi-Scale Convolutional Recur-rent Networks. *arXiv* **2022**, arXiv:2204.03779.
19. The UNSW-NB15 Dataset | UNSW Research. Available online: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 12 December 2022).
20. NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/nsl.html (accessed on 6 December 2019).
21. Elubeyd, H.; Yiltas-Kaplan, D. Hybrid Deep Learning Approach for Automatic DoS/DDoS Attacks Detection in Software-Defined Networks. *Appl. Sci.* **2023**, *13*, 3828. [CrossRef]
22. Yaser, A.L.; Mousa, H.M.; Hussein, M. Improved DDoS Detection Utilizing Deep Neural Networks and Feedforward Neural Networks as Autoencoder. *Futur. Internet* **2022**, *14*, 240. [CrossRef]
23. Jiang, H.; Turki, T.; Wang, J.T. DLGraph: Malware detection using deep learning and graph embedding. In Proceedings of the 2018 17th IEEE international conference on machine learning and applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; IEEE: Piscataway, NJ, USA, 2018.
24. Elsayed, M.S.; Le-Khac, N.-A.; Dev, S.; Jurcut, A.D. Network Anomaly Detection Using LSTM Based Autoencoder. In Proceedings of the Q2SWinet'20, Alicante, Spain, 16–20 November 2020. [CrossRef]
25. Shiravi, A.; Shiravi, H.; Tavallaee, M.; Ghorbani, A.A. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Comput. Secur.* **2012**, *31*, 357–374. [CrossRef]
26. Tonkal, Ö.; Polat, H.; Başaran, E.; Cömert, Z.; Kocaoğlu, R. Machine Learning Approach Equipped with Neighbourhood Component Analysis for DDoS Attack Detection in Software-Defined Networking. *Electronics* **2021**, *10*, 1227. [CrossRef]
27. Ivanova, V.; Tashev, T.; Draganov, I. Detection of IoT based DDoS Attacks by Network Traffic Analysis using Feedforward Neural Networks. *Int. J. Circuits Syst. Signal Process.* **2022**, *16*, 653–662. [CrossRef]
28. Prasad, A.; Chandra, S. VMFCVD: An Optimized Framework to Combat Volumetric DDoS Attacks using Machine Learning. *Arab. J. Sci. Eng.* **2022**, *47*, 9965–9983. [CrossRef] [PubMed]
29. Ahuja, N.; Singal, G.; Mukhopadhyay, D.; Kumar, N. Automated DDOS attack detection in software defined networking. *J. Netw. Comput. Appl.* **2021**, *187*, 103108. [CrossRef]