Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 191 (2021) 254–263

Procedia Computer Science

www.elsevier.com/locate/procedia

The 16th International Conference on Future Networks and Communications (FNC)
August 9-12, 2021, Leuven, Belgium

Deep Learning Algorithms for Detecting Denial of Service Attacks in Software-Defined Networks

Abdullah Soliman Alshra'a*, Ahmad Farhat,, Jochen Seitz

Communications Networks Group, Technische Universität Ilmenau, Germany,
[Abdullah.Alshraa, Ahmad-Hussein.Farhat, Jochen.Seitz]@tu-ilmenau.de

Abstract

In Software-Defined Networking (SDN) the controller is the only entity that has the complete view on the network, and it acts as the brain, which is responsible for traffic management based on its global knowledge of the network. Therefore, an attacker attempts to direct malicious traffic towards the controller, which could lead to paralyze the entire network. In this work, Deep Learning algorithms are used to protect the controller by applying high-security measures, which is essential for the continuous availability and connectivity in the network. Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are proposed to recognize and prevent the intrusion attacks. We evaluate our models using a recently released dataset (InSDN dataset). Finally, our experiments manifest that our models achieve very high accuracy for the detection of Denial of Service (DoS) attacks. Thus, a significant improvement in attack detection can be shown compared to one of the benchmarking state of the art approaches.

Keywords: Software-Defined Networking (SDN); Denial of Service (DoS); Deep Learning (DL); Recurrent Neural Networks (RNN); Long Short-Term Memory (LSTM); Gated Recurrent Unit (GRU);

1. Introduction

Current communication networks architecture has rapidly been developed for the last three decades, and the amount of the exchanged data has increased with the number of the connected devices, especially through the Internet.

Moreover, the traditional network is rigid and not flexible to accommodate all the new requirements. Therefore, Software-Defined Networking (SDN) was defined to provide the ability to respond to all the requests and dynamic nature of modern day applications. SDN presents an optimal solution by offering a new networking architecture that

* Corresponding author. Tel.: +49-3677-69-2698; fax: +49-3677-69-1143.
E-mail address: Abdullah.Alshraa@tu-ilmenau.de

1877-0509
10.1016/j.procs.2021.07.032

decouples the control and the data plane by conceiving two different networking devices in contrast to the legacy networking devices which include both parts in the same network node [5].

Consequentially, SDN gives the network administrators, developers, engineers and vendors the ability to implement their enhancements, ideas and functionalities of the network in a programmable and flexible way. Mainly, the SDN architecture consists of three layers. Firstly, on top, there is the application layer comprising network policies as well as applications related to network automation, network configuration and management, network monitoring, network troubleshooting and network security. Secondly, the control layer comprised of so called SDN controllers is responsible for applying the application layer's policies with programmed algorithms enabling the control of the forwarding nodes in the infrastructure layer, which is the third layer. The connection between the application layer and control layer is done via the northbound API, while the connection between the control layer and the infrastructure layer happens via the southbound API using the OpenFlow protocol [9]. In SDN, once the switch has received a new packet, the switch looks up its flow tables in order to match the packet's header with one of its flow entries. Thereby, the action associated to the flow entry would be executed. But in case the packet header does not match any flow entry, the miss table entry matches the packet and hence the switch sends a `Packet_In` request to the controller. After the controller has received this request, it analyzes it by its functionalities in order to install a new entry into the flow table of the requesting switch with a `Packet_Out`.

Based on the aforementioned way, the controller is considered as the brain of the network, because the OpenFlow switches forward the network traffic based on the rules in their flow table installed by the controller [9]. Nevertheless, the controller could be a single point of failure because if the controller is turned off, the entire network will be down and no more connections will be available. Therefore, the attacker targets the controller and exploits the connection procedure between the switches and the controller by generating a massive number of packets with different destination addresses that do not matches the switch's flow table forcing the switch to send an according number of `Packet_In` messages to the controller [5].

As the controller has limited resources, it would be out of service when the received number of requests exceeds its abilities. However, a benign user sometimes behaves similar to a malicious user when the the number of requests is increasing because of temporary reasons (e.g., free services on some servers), which is known as flash crowd [16]. Thus, implementing solid security measures on the controller is essential to protect the controller against Distributed Denial of Service (DDoS) attacks. The solution has to distinguish between malicious and benign users with a high accuracy that ensures the lowest False Alarm Rate (FAR) and minimizes the associated computational cost as much as possible. Recently, Deep Learning (DL) algorithms have been suggested as a new solution that achieves the required accuracy better than other machine learning algorithms. Therefore, in this paper, the performance of Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) DL algorithms is investigated for an intrusion detection system (IDS) based on flow states.

To the best of our knowledge, this is the first work to use LSTM and RNN based on flow states. Moreover, the paper extends the GRU tested with the InSDN dataset with 48 parameters. The remainder of this paper is organized as follows. Related approaches are introduced in section 2. The method we propose is described in section 3. In section 4, we evaluate our approach with simulation experiments. Finally in section 5, we conclude our paper.

2. Related Work

The SDN architecture allows the network providers and administrators to benefit from the programmability, flexibility, and ease of deployment of this architecture. Also, it includes the implementation of different functionalities and security measures. This section briefly introduces some recent and popular approaches that have been suggested for the detection of DDoS attacks in SDN. The measures to assess the different approaches are accuracy, precision, recall and F1 score as defined in section 4.

The authors of [10] used three different algorithms (Genetic, Ranker, Greedy) with selected features in order to train the Neural Network Blocks (NB) classifier. They obtained 81% precision, while recall and F1 score are 0.77 and 0.77, respectively. Tang et al. [13] presented GRU-RNN for an anomaly-based IDS in an SDN environment.

The authors used the NSL-KDD dataset and selected six features that are related to the SDN nature. Their work showed an improvement over the state of the art algorithms, with a detection rate of 89% and they also introduced a network performance analysis, where they implemented their work for real-time traffic classification. In [15], Support

Vector Machine (SVM) algorithms are used to analyze the behaviour of the traffic based on six features that can be collected by the controller. Prasath et al. [11] proposed an agent program framework which protects the OpenFlow switches against external attacks. Furthermore, they used a meta-heuristic bayesian network classification algorithm to classify the incoming packets into benign or malicious.

Thus, the the proposed approach obtains 82.99% as an overall accuracy, 77% precision, 74% and 75% for recall and F1 score respectively. The authors of [4] proposed InSDN, a novel dataset for the context of software-defined networks. InSDN is the first dataset specific to the SDN environment. It reflects real-world attacks such as DDoS, DoS, probe, web-attack, botnet, brute force, and U2R attacks. Moreover, they evaluated the performance of different machine learning algorithms such as MLP, SVM, Random Forest, Decision Tree, KNN, Nearest Neighbor, and Adaboost. in [12], Santos et al. implemented and analyzed four machine learning algorithms (Multiple Layer Perception, Support Vector Machine, Decision Tree and Random Forest) to classify DDoS attacks in an SDN environment. They adopted the Scapy tool to produce traffic in their simulation. As an outcome, they claim that the Random Forest algorithm showed the best accuracy and the Decision Tree algorithm had the best processing time. The security of an Internet of Things (IoT) SDN network infrastructure is taken in account by the authors of [14]. They suggested to use co-detection modal of deep learning with snort for controlling high terrific flows, where each OpenFlow switch has an agent for sampling the distributed traffic using sFlow- and adaptive polling-based sampling techniques. Thereby, the SnortIDS and stacked autoencoders are applied to achieve high accuracy and low false prediction rate. Dekhordi et al. [2] suggested an approuch to detect DDoS attacks in an SDN environment that depends on three parts collector, entropy-based and classification sections based on machine learning algorithms (BayesNet, J48, Random Tree, logistic regression, REPTree classification algorithms). The results achieved a good outcome in term of accuracy in detecting DDoS attacks in SDN. They assessed their results by applying the UNB-ISCX, CTU-13 and ISOT datasets.

3. Methodology and System Description

Deep learning algorithms are considered as a part of machine learning algorithms. They are based on special nodes known as neurons that receive data as input and apply some operations like multiplying them by weights or adding them to other values in order to obtain the output which could be used as input for the next neuron and so on. The neurons are arranged next to each other and in layers according to the algorithm. In this section, RNN, LSTM and GRU are briefly explained. Moreover, the InSDN dataset is discussed.

3.1. Recurrent Neural Networks (RNN)

The RNN model is broadly used in different research and development areas such as image processing, reading handwriting and speech recognition. Also, it is considered as a type of neural network algorithms designed to capture information from sequence points. Therefore, in contrast to feed forward neural networks, RNN can solve a problem that requires processing a sequence of data. Furthermore, RNN has a recurrent structure because the output of one step is fed as an input to the next step [7]. For a better understanding, assume the input to the model over time t given by the vector $X = (x_1, x_2, x_3, ..x_t)$. Moreover, the neuron nodes in the layers between the input and output nodes (hidden nodes) can be computed as the vector sequence $H = (h_1, h_2, h_3, ..h_t)$. The output nodes are considered as the vector sequence $Y = (y_1, y_2, y_3, ..y_t)$. The RNN model as depicted in Figure 1.(a) calculates the hidden node h_t at time t according to the following equation:

$$h_t = \sigma(W_{xh}x_t + W_{hh}h_{t-1} + b_h); \tag{1}$$

where σ is the activation function, W are different weights, b_h is the bias and h_{t-1} is the hidden node at time $t-1$. Also, in the end, the output y_t at time t is defined in the next equation:

$$y_t = W_{hy}h_t + b_y \tag{2}$$

Finally, an RNN can learn and enhance the output with massive amounts of data and an iterative approach known as backpropagation. Every time, an RNN receives data, it produces an output, which is compared to that the desired result by a loss function.

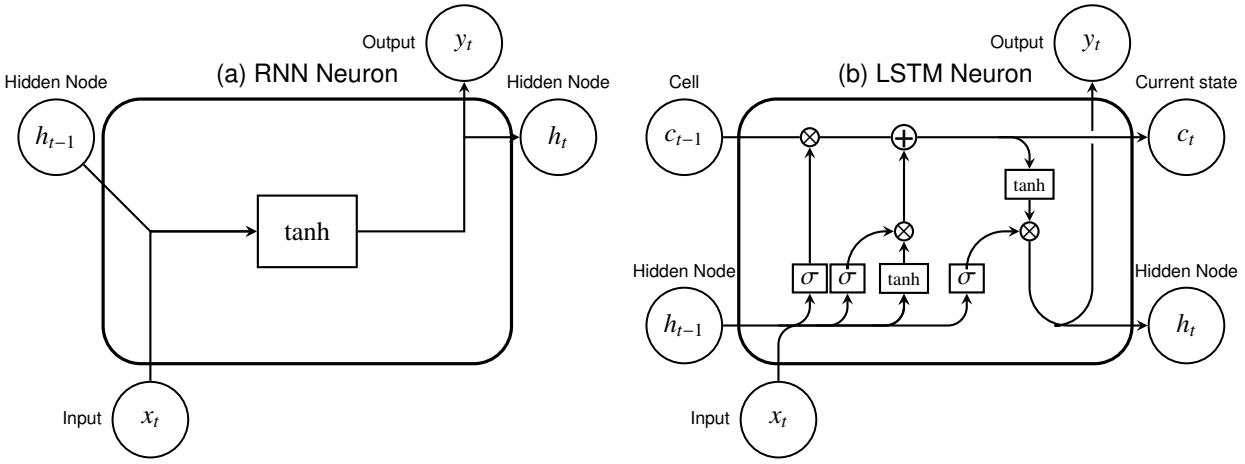

Fig. 1: Different Neuron Nodes

Repeatedly, the RNN readjusts the weights based on the difference between the results and the target result. However, initial weights are assigned at the beginning with the random values, which are close to zero. When the RNN starts with backpropogation by multiplying a tiny number by the weight values, the result becomes less and less which called the vanishing gradient problem. Therefore, Long Short Term Memory (LSTM) and Gated Recurrent Units (GRUs) were proposed to solve this problem

3.2. Long Short Term Memory (LSTM)

LSTM is an enhanced architecture of RNN, which is used for deep learning applications, such as, intrusion detection systems, speech recognition, and connected handwritten applications. Also, the RNN problem of vanishing gradient has been solved by the LSTM architecture. So, RNN is able to benefit better from the feedback connections to process the sequence of data. This means that LSTM fits to learn from experience to classify a time series [13]. In general, an LSTM has a similar control flow as an RNN. But the differences are the operations inside the LSTM's cells which are depicted in as depicted in Figure 1.(b).

The values over arbitrary time intervals are memorized by the cell and the flow of information into and out of the cell is regulated by the three gates: an input gate, an output gate, and a forget gate. In general, the most important functionality in the LSTM cell is to decide which information should be deleted or kept. The forget gate is responsible for information coming from a prior hidden state (h_{t-1}) and from the currant input (x_t). It combines them and applies the σ function to get values between 0 and 1. If the value closer to 0, the information would be removed, otherwise the value would be passed to next step in order to process with the previous cell state (c_{t-1}). Moreover, the result of ($h_{t-1} + x_t$) is passed through another σ and tanh function respectively. Thereafter, the result would be multiplied. The current state (c_t) is then calculated by the next equation:

$$c_t = c_{t-1} \otimes f_t \oplus I_t \tag{3}$$

where, f_t is the forget gate

$$f_t = \sigma(W_f(x_t + h_{t-1})) \tag{4}$$

and I_t is the input gate, which consists of two equations:

$$i_{t1} = \sigma(W_{i2}(x_t + h_{t-1})) \tag{5}$$

$$i_{t2} = \tanh(W_{i1}(x_t + h_{t-1})) \tag{6}$$

With the previous two equations, we obtain the input gate I_t value as

$$I_t = i_{t1} \otimes i_{t2} \tag{7}$$

The output gate decides which information in the cell state is sent to the network as input in the following time step. The output gate's activation is given by.

$$h_t = \sigma(W_o(x_t + h_{t-1})) \otimes \tanh(I_t) \tag{8}$$

LSTM solved the problem of the vanishing ingredient, but it gets more complex because LSTM needs a lot of resources and time to be trained and ready for real-world applications.

3.3. Gated Recurrent Units (GRU)

GRU is another advanced type of RNN and similar to LSTM, but GRU needs less time to get trained because of the simplicity of the structure of its gates, as it lacks an output gate.

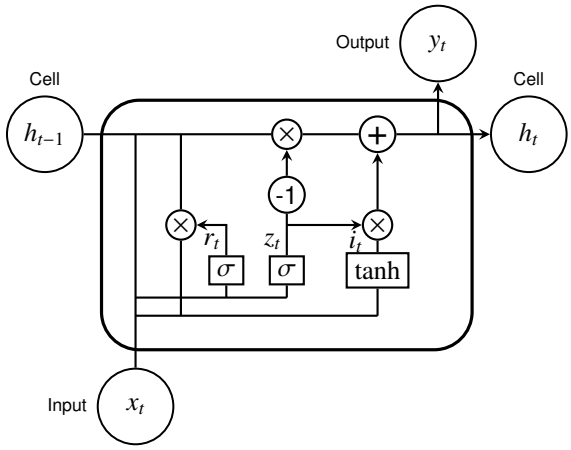

Fig. 2: Gated Recurrent Unit Cell.

GRU contains two sigmoid gates and one hidden state. The computation can be summarized as:

$$h_t = (1 - z_t) \otimes h_{t-1} \oplus z_t \otimes i_t \tag{9}$$

where z_t , r_t and i_t are obtained as the result of the next equations respectively according to the depicted gates in Figure 2:

$$z_t = \sigma(W_z(h_{t-1} + x_t)) \tag{10}$$

$$r_t = \sigma(W_r(h_{t-1} + x_t)) \tag{11}$$

$$i_t = \tanh(W_i(r_t \otimes (h_{t-1} + x_t))) \tag{12}$$

Figure 2 shows the architectural details of a single GRU cell, which replaces the forget and input gates with an update gate i_t, adds a reset gate r_t in order to modify h_{t-1} and removes the internal memory of LSTM cell c_t. In other words, the main differences between GRU and LSTM are the number of gates and the maintenance of cell states. In contrast to GRU, LSTM has three gates (input, forget, output) and maintains an internal memory cell state, which makes it more flexible, but less efficient memory and time wise. Although GRU and LSTM are perfect to resolve the vanishing gradient problem, both of them need to track long term dependencies. Moreover, it is recommended to extensively train an LSTM in the beginning, because it has more parameters and is a bit more flexible. Nevertheless, in case there are no quantifiable differences in their performance, then GRU would be better because it is simpler and more efficient [6].

3.4. Dataset

The InSDN dataset [4] is one of the state-of-the-art datasets for intrusion detection system evaluation in the context of software-defined networks. This dataset has 83 features. Details of these features are explained in the given reference.

The dataset includes benign traffic and various attack categories that can occur in the different elements of the SDN platform.

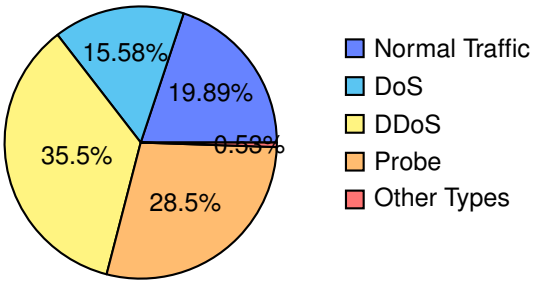

Fig. 3: InSDN Dataset Distribution.

Figure 3 explains the dataset distribution that describe different types of attacks and includes seven attack classes:

1. *DoS attack*:
   (a) DoS attack towards the SDN controller,
   (b) DoS attack to affect benign users, network bandwidth or any network device using protocols such as UDP, TCP or ICMP,
   (c) DoS attacks that overflow a server or the application layer on the victim device using the HTTP protocol.
2. *DDoS Attack*: TCP-SYN Flood, UDP Flood, and ICMP Flood attacks from different sources towards the same victim in parallel.
3. *Probe:* The main objective for this type of attack is to obtain detailed information about the victim by scanning the operation system, discovering the open ports etc.
4. *Botnet:* The attacker controls some benign users by malware in order to run different malicious activities such as stealing information, fraud attack, launching DDoS attacks against victim servers or web applications server.
5. *Web Attack*: Once the user accesses a web site, a malicious code is installed by the attacker. If the malicious code is executed, the attacker obtains information from the client machine, such as session tokens, cookies etc. Moreover, the attacker could access the database of a server by injecting malicious SQL code into the web application server. Thereby, the attacker has the power to change, delete, misuse and steal information stored in the database.
6. *Password Brute-Force Attack (BFA)*: The attacker creates a dictionary for all user names and password credentials and tries all of them.
7. *Exploitation User-to-Root (U2R)*: A normal user illegally accesses either root's or super user's privileges (such as admin) in the network.

As a result, it is clear that despite the numerous benefits of the SDN technology, SDN is more sensitive to new malicious threats that can be used by malicious users to launch different attack types.

4. Detection Performance Analysis

In this section, detection evaluation metrics are explained firstly. Afterwards, the experimental setup is described. Finally, our work is compared to another existing approach.

4.1. Evaluation Metrics

Firstly, for our evaluation, 48 features are selected as suggested in [4] for the used algorithms, which are then compared with the suggested six features in [13] who only applied the GRU Model. However, in this experiment, the previous approach with GRU is extended to work with RNN and LSTM as well. Further, all the considered features can be retrieved from the basic flow statistics features of the OpenFlow Switches. Secondly, for the evaluation of the aforementioned methodologies, four measures are considered, where True Positive (TP) is considered as the number of malicious records classified accurately, True Negative (TN) is the number of normal records classified accurately, False Positive (FP) is the number of normal records incorrectly classified as attack and False Negative (FN) is the number of malicious records incorrectly classified as normal traffic. Based on this, the metrics are calculated according to Table 1.

Table 1: Evaluation Metrics

Accuracy	**Precision**	**Recall**	**F1 Score**
the percentage of correctly classified flows in relation to the total number of classified flows	the percentage of malicious flows identified correctly from the all the flows identified as attacks	the percentage of correctly classified malicious flows versus all malicious flows present in the dataset	the harmonic mean between Precision and Recall
$\frac{TP+TN}{TP+TN+FP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$

In the experiments, TensorFlow and Keras framework are used [8, 1] to implement RNN, LSTM and GRU in Python programming language. Also, in order to select the best values of hyper-parameters to build our neural network structure, the best practice, trial and error or human knowledge is still considered [4], with testing different values and recording the corresponding results in each test experiment and then identifying the values which provide the highest accuracy in trade off with training time. Sequentially, all our models are built with two hidden layers where include 32 and 16 neural nodes with Rectified Linear Unit (ReLu) activation function respectively. The dense layer has two units as an output layer, two units work with Sigmoid function. Additionally, the experiments use a Nadam optimizer [3] and a mean squared error (MSE) for the model. The hyper-parameter configuration is 100, 10 and 0.001 for the batch size, the epoch and the learning rate, respectively. The details of our models for 48 and 6 features can be seen in Table 2. All the experiments were performed on Ubuntu 18.04.5 LTS 64-bit operating system with Intel®Core™ i5-8400, CPU @ 2.80GHz × 6, 16 GB of RAM and AMD® Rv635 Graphics.

Table 2: Neural Network Model Structure

	48 Features			**6 Features**		
Algorithm	*Input Layers*	*Hidden Layers*	*Output Layers*	*Input Layers*	*Hidden Layers*	*Output Layers*
RNN, LSTM, GRU	48	32, 16	2	6	6, 4, 2	2

4.2. Experimental Results

4.2.1. Performance Comparison

Firstly, the detection performance of RNN, LSTM, and GRU with 48 features are presented in terms of accuracy, precision, recall, f1 score. As shown in Table 3, all models achieve good results with around 10% enhancements in comparison to the models with 6 features. Moreover, LSTM shows the best results in term of precision and accuracy, although the training time is always faster for RNN and GRU according to Table 4. Secondly, the Receiver Operating Characteristic (ROC) curve is introduced for both numbers of features in order to estimate how correctly the models

work. The ROC curve depicts the relation between false positive rate and true positive rate which provides the Area Underneath the ROC Curve (AUC) that is useful to define which classifier predicts the classes best. Therefore, Figure 4 show that our models with 48 features grant 13% AUC enhancement for LSTM, and around of 15% for both of RNN and GRU compared to the models with 6 features. Considering 48 features, LSTM is able to classify 98% of positive and negative classes successfully, while RNN and GRU are able to correctly classify 96.2% and 96.4% respectively.

Table 3: Detection Performance Comparison for All Attacks

	with 48 Features				**with 6 Features**			
Algorithm	*Precision*	*Recall*	*F1 Score*	*Accuracy*	*Precision*	*Recall*	*F1 Score*	*Accuracy*
RNN	0.97899	0.99658	0.98771	0.980946	0.89941	0.99577	0.94514	0.91117
LSTM	0.98842	0.99702	0.99270	0.98874	0.92130	0.98771	0.95335	0.92573
GRU	0.97948	0.99757	0.98844	0.98208	0.90175	0.99544	0.94628	0.91315

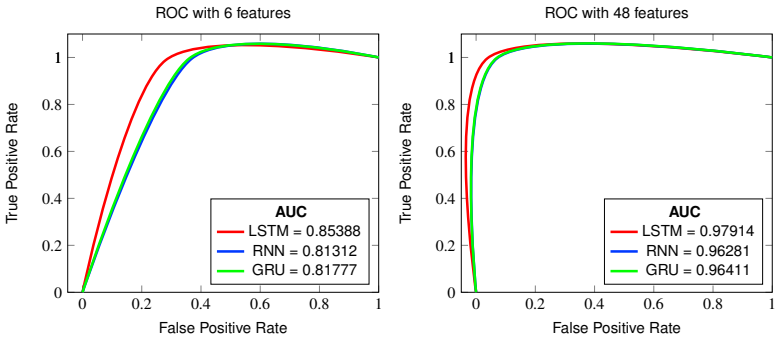

Fig. 4: Receiver operating characteristic.

4.2.2. Analysis of the Proposed Models

In addition to the foreseen outcome, the experiments provide more analysis for the detection of different attacks and compare our models' performance with RNN, LSTM and GRU. All models show outstanding performance in term of accuracy. The precision is also perfect for the detection of DDoS, DoS, and Probe. Besides, LSTM provides the best result to detect Botnet, web-attack, brute force attacks, and U2R attacks compared to RNN and GRU as can be seen in Table 4. Generally, all models consistently have good scores on all metrics for DDoS, DoS, and probe classes, while their performance are notably declined on botnet, web attack, and U2R classes due to the low number of samples of these attacks in the dataset. Notably, the models are able to detect these attacks better when they work with the entire dataset, because the records of the attack classes often have mutual features. Besides, recall and f1 score for RNN and GRU are relativity low on web attack and U2R attacks. Finally, we compared the performance of LSTM, RNN, and RNN in terms of training time. Table 4 shows that both GRU and RNN needed less training time than LSTM, where RNN is always the fastest model due to use one gate, but it is still vulnerable to the problem of vanishing gradients. Moreover, GRU is faster than LSTM, but its results are comparable to the LSTM results or even better when the number of samples get lower. However, with an increasing number of samples, LSTM is performing better although it needs more time than the others for training or to respond because LSTM has three gates.

5. Conclusions

Applying SDN and the network virtualization concepts offers the possible malicious attacker more vulnerable aspects to threaten the network architecture in addition to the threats that are already existing in the traditional network. Therefore, this research work sueggests to use Deep Learning algorithms, which achieved great success in many

Table 4: Evaluation of LSTM, RNN, and GRU with 48 Features

Learning Model	*DDoS*	*DoS*	*Probe*	*Botnet*	*Web-Attack*	*BFA*	*U2R*	
Accuracy Score.								
LSTM	0.99940	0.98131	0.98192	0.99897	0.96633	0.99584	0.99802	
RNN	0.99936	0.98975	0.98333	0.99876	0.96320	0.99126	0.9989	
GRU	0.99943	0.98332	0.98189	0.99584	0.96342	0.99334	0.99948	
Precision Score.								
LSTM	0.99925	0.97793	0.97403	0.70212	0.07444	0.99476	0.074074	
RNN	0.99911	0.99536	0.97395	0.66000	0.068518	0.85972	0.13333	
GRU	0.99918	0.97565	0.97356	0.36666	0.06890	0.98958	0.28571	
Recall Score.								
LSTM	0.99959	0.97957	0.99628	1.00000	0.94871	0.67615	0.50000	
RNN	0.99966	0.98125	0.99842	1.00000	0.94871	0.67615	0.50000	
GRU	0.99972	0.98666	0.99587	1.00000	0.94871	0.67615	0.50000	
F1 score.								
LSTM	0.99942	0.97875	0.98189	0.8250	0.13805	0.80508	0.12903	
RNN	0.99938	0.98826	0.98603	0.79518	0.12780	0.75697	0.21052	
GRU	0.99945	0.98113	0.98483	0.53658	0.12847	.80338	0.36363	
Training Times.								All Types
LSTM	21.2022	18.53204	24.26479	19.42383	19.30129	11.52263	19.59553	41.39506
RNN	14.70538	12.73099	16.70782	13.21762	13.37479	8.08346	13.3889	28.74049
GRU	21.13238	18.36072	23.72825	18.99140	19.18356	11.55337	11.51159	40.94993

different applications. They can extract raw features from measured data without any human intervention. They are able to analyse both training and test data and use any ongoing data to correct the understanding for the upcoming data. This research paper extends work on the state of the art which uses GRU only to use RNN and LSTM. Thus, RNN, LSTM and GRU outperform the existing work by changing the structure and the number of the input features. In our work, we evaluated the models' performance with 48 features related to the SDN environment which achieved a high detection rate. The experiments achieve around 10% enhancements in comparison to the models with 6 features as well as around of 15% for AUC. Also, the work shows that LSTM requires more training time compared to RNN and GRU, while both models achieved almost the same performance in terms of detection accuracy, and both models surpassed RNN. In our future work, the performance of other machine learning algorithms will be investigated. We will carefully select better combinations of features by aggregating existing features, and choose the best combination of algorithms and features. Finally, we will apply our models for real-time classification to evaluate the network performance.

References

[1] Abadi, M., 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://download.tensorflow.org/paper/whitepaper2015.pdf.

[2] Dehkordi, A.B., Soltanaghaei, M.R., Boroujeni, F.Z., 2020. The DDoS Attacks Detection through Machine Learning and Statistical Methods in SDN. The Journal of Supercomputing 77, 2383—-2415. URL: https://doi.org/10.1007/s11227-020-03323-w.

[3] Dozat, T., 2015. Incorporating Nesterov Momentum into Adam http://cs229.stanford.edu/proj2015/054_report.pdf.

[4] Elsayed, M.S., Le-Khac, N.A., Jurcut, A.D., 2020. InSDN: A Novel SDN Intrusion Dataset. IEEE Access 8, 165263–165284.

[5] Gadze, J.D., Bamfo-Asante, A.A., Agyemang, J.O., Nunoo-Mensah, H., Opare, K.A.B., 2021. An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers. Technologies 9, 14.

[6] Hu, C., Ou, T., Chang, H., Zhu, Y., Zhu, L., 2020. Deep GRU Neural Network Prediction and Feed Forward Compensation for Precision Multiaxis Motion Control Systems. IEEE/ASME Transactions on Mechatronics 25, 1377–1388.

[7] Jiang, F., Fu, Y., Gupta, B.B., Liang, Y., Rho, S., Lou, F., Meng, F., Tian, Z., 2018. Deep Learning Based Multi-Channel Intelligent Attack Detection for Data Security. IEEE Transactions on Sustainable Computing 5, 204–212.

[8] Keras, 2021. Keras Framework. https://keras.io/. [Online; accessed 21-Feb-2021].

[9] Li, C., Wu, Y., Yuan, X., Sun, Z., Wang, W., Li, X., Gong, L., 2018. Detection and Defense of DDoS Attack–Based on Deep Learning in OpenFlow-Based SDN. International Journal of Communication Systems 31, e3497.

[10] Mohammed, S.S., Hussain, R., Senko, O., Bimaganbetov, B., Lee, J., Hussain, F., Kerrache, C.A., Barka, E., Bhuiyan, M.Z.A., 2018. A New Machine Learning-based Collaborative DDoS Mitigation Mechanism in Software-Defined Network, in: 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE. pp. 1–8.

[11] Prasath, M.K., Perumal, B., 2019. A Meta-Heuristic Bayesian Network Classification for Intrusion Detection. International Journal of Network Management 29, e2047.

[12] Santos, R., Souza, D., Santo, W., Ribeiro, A., Moreno, E., 2020. Machine Learning Algorithms to Detect DDoS Attacks in SDN. Concurrency and Computation: Practice and Experience 32, e5402.

[13] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M., 2018. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks, in: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), pp. 202–206. doi:10.1109/NETSOFT.2018.8460090.

[14] Ujjan, R.M.A., Pervez, Z., Dahal, K., Bashir, A.K., Mumtaz, R., González, J., 2020. Towards sFlow and adaptive Polling Sampling for Deep Learning Based DDoS Detection in SDN. Future Generation Computer Systems 111, 763–779.

[15] Ye, J., Cheng, X., Zhu, J., Feng, L., Song, L., 2018. A DDoS Attack Detection Method Based on SVM in Software Defined Network. Security and Communication Networks 2018.

[16] Yue, M., Wang, H., Liu, L., Wu, Z., 2020. Detecting DoS Attacks Based on Multi-Features in SDN. IEEE Access 8, 104688–104700.