Hindawi
Security and Communication Networks
Volume 2019, Article ID 8939041, 9 pages
https://doi.org/10.1155/2019/8939041

Research Article

Recurrent Neural Network Model Based on a New Regularization Technique for Real-Time Intrusion Detection in SDN Environments

Marwan Ali Albahar

Umm Al Qura University, College of Computers in Al-Leith, Mecca, Saudi Arabia

Correspondence should be addressed to Marwan Ali Albahar; marwanalialbahar@gmail.com

Received 26 May 2019; Revised 10 September 2019; Accepted 21 October 2019; Published 18 November 2019

Academic Editor: Huaizhi Li

Software-defined networking (SDN) is a promising approach to networking that provides an abstraction layer for the physical network. This technology has the potential to decrease the networking costs and complexity within huge data centers. Although SDN offers flexibility, it has design flaws with regard to network security. To support the ongoing use of SDN, these flaws must be fixed using an integrated approach to improve overall network security. Therefore, in this paper, we propose a recurrent neural network (RNN) model based on a new regularization technique (RNN-SDR). This technique supports intrusion detection within SDNs. The purpose of regularization is to generalize the machine learning model enough for it to be performed optimally. Experiments on the KDD Cup 1999, NSL-KDD, and UNSW-NB15 datasets achieved accuracies of 99.5%, 97.39%, and 99.9%, respectively. The proposed RNN-SDR employs a minimum number of features when compared with other models. In addition, the experiments also validated that the RNN-SDR model does not significantly affect network performance in comparison with other options. Based on the analysis of the results of our experiments, we conclude that the RNN-SDR model is a promising approach for intrusion detection in SDN environments.

1. Introduction

The current Internet architecture has existed for almost three decades and is now becoming a progressively complicated system. The Internet lacks the capacity to accommodate continually changing requirements and the demanding nature of present day applications. Software-defined networking (SDN) [1] was introduced as an architecture permitting unparalleled compliance and scalability in the implementation and configuration of network services. The segregation of the data and control planes affords better control over traffic flow and flexibility. Real-time information acquisition via the OpenFlow protocol [2] is made possible due to the flow-based nature of SDNs. However, the SDN architecture also contains numerous security challenges concerned with the control application interface, control plane, and control data interface [3]. Consequently, SDN security has become a major issue and has gained critical significance [4, 5].

A significant network security tool is an intrusion detection system (IDS). An anomaly based IDS attempts to recognize deviations from a baseline model. Much research has been performed in the context of detecting anomalies in an SDN environment. While these researches showed great results, they are limited in their applicability. Techniques proposed to detect anomalies have included Bayesian networks, support vector machines (SVMs), and artificial neural networks (ANN), but these proposals have suffered from excessive computational cost and high false alarm rate (FAR) [6]. Lately, traditional machine learning methods have been replaced by a new approach, called deep learning (DL), that gives better accuracy when compared with traditional machine learning. DL can extract deep features in order to obtain high-level features. In SDN environments

(constrained resource network), DL has potential due to its adaptability.

Drawing on cutting-edge research in the field of anomaly detection, the recurrent neural network (RNN) is the most popular method of performing classification and other analysis on sequences of data. In addition, it is a powerful technique that can show remarkable results in sequence learning and improving the anomaly detection rate in an SDN environment. In this paper, we propose an RNN model based on a new regularization technique RNN-SDR. RNN-SDR is tested within an SDN controller against the KDD Cup 1999, NSL-KDD, and UNSW-NB15 datasets.

The major contributions of this paper are as follows:

(1) We introduce the design and implementation of an IDS in an SDN environment using an RNN based on a new regularizer that decays the weights according to the standard deviation of the weight matrices and compares the results against its parents. To the best of our knowledge, this is the first model that has achieved a high accuracy for IDS in an SDN environment in terms of throughput and latency. Noteworthy, however, is that it is slightly slower than the Beacon controller.

(2) The RNN-SDR model achieves a detection rate of 99.5% using the KDD Cup 1999 dataset, 97.43% using the NSL-KDD dataset, and 99.9% using the UNSW-NB15 dataset. This indicates that the RNN-SDR model outperforms other state-of-the-art models.

(3) We also evaluate the RNN-SDR model performance in the SDN controller. The test outcomes indicate that the RNN-SDR model has significant potential for real-time detection.

The rest of this paper is organized as follows. First, the applicability of deep learning in the domain of intrusion detection is established, followed by a review of related work in this field. The system developed in this research is described, and the datasets used to evaluate the system are presented. The system's intrusion detection and network performance is presented, analyzed, and compared with the state of the art. Finally, the paper is concluded.

2. Deep Learning for Intrusion Detection

Deep learning is an advanced field of machine learning (ML) that allows the creation of models with discriminative powers that exceed other statistical ML methods. The basic algorithms of deep learning are deep neural networks (DNNs) that operate across several connected layers. The layers are linked in a way that sees each forward layer taking inputs from the previous layer and modifying those inputs in a hidden way. These algorithms have the advantage of being able to extract discriminative features from data in a hierarchical fashion in a way that best represents the data without resorting to handcrafting.

For intrusion detection, features such as *protocol_type*, *duration*, *service*, and *flag* are fed to the neural network and pass through several layers. Every layer in the neural network works as a transformation of features. Each feature becomes more discriminative after passing through a hidden layer. The features pass through several hidden layers and, in the last output layer, the outcome of the neural network is compared with labels attached to the original data to determine whether the network has detected attack types such as *DoS*, *Probe*, and *U2R*. Due to its discriminative power, deep learning approaches have been used by many authors [7, 8] for network intrusion detection, and still the area is open for quality research.

3. Related Work

Before the development of DNN variants, classical ML algorithms, such as random forest (RF), SVM, ANN, and k-nearest neighbors (KNN) were used by various researchers to develop IDSs [9–12]. However, these methods have inherent limitations. In particular, these focus on a large set of features of traditional networks that cannot be applied to SDNs.

Work on anomaly detection in SDN using flow-based IDS was employed in [13, 14]. Self-organizing map (SOM), used by Braga et al. [13], is considered to be a light weight approach for detection of distributed denial of service (DDoS) attacks in SDNs. The accuracy of this approach was found to be very high based on six traffic flow features. Four traffic anomaly detection algorithms NETAD, maximum entropy, threshold random walk with credit-based (TRW-CB) rate limiting, and rate limiting were used in [14] in an SDN environment for anomaly detection. Their simulations showed that these algorithms produced promising results with low overhead in small networks. Other algorithms to detect DDoS attacks, such as SVM, were used in [15, 16] and produced better results. An ensemble of graph theory algorithms based on KNN was used by ALEroud and Alsmadi to detect anomalous flow in SDNs [17]. An algorithm based on the variation of the entropy of the destination IP addresses of the flow in an SDN was proposed by Mousavi et al. [18] to detect early DDoS attacks. The sensitivity was about 96% for 250 packets at the start. Similarly, in [19], a DL-based approach using a stacked autoencoder was used. Their algorithm performance, evaluated on their own dataset, was quite satisfactory having high accuracy and low FAR. In [20, 21], the authors applied DNN and a gated recurrent unit recurrent neural network (GRU-RNN) in an SDN environment. They achieved an average accuracy of 75.75% for DNN and 89% for GRU-RNN using six basic features using the NSL-KDD dataset.

4. Methodology and System Description

In this section, we review the RNN and the new regularizer. Then, we describe, in detail, the architecture of the SDN-based IDS. Finally, we discuss the KDD Cup 1999, NSL-KDD, and UNSW-NB15 datasets.

4.1. Recurrent Neural Networks. The RNN architecture is the addition of sequential information to the feedforward neural

network. The RNN performs the same task for each part. This is why it is called a recurrent network; the output is dependent upon the previous computation. The hidden computation of RNN is computed as given below:

$$H_t = \sigma\left(Wx_t + VH_{t-1} + b_H\right),\ t = T, \ldots, 1, \tag{1}$$

where H_t denotes the hidden state vector at time t; σ is the activation function, also known as the nonlinearity function; W is the hidden weight matrix; V is the hidden to hidden weight matrix; x_t is the input vector at time t; and b_H is the bias term.

4.2. A New Regularization Technique. The regularization technique used in this paper is based on taking the standard deviation of the weight matrix and multiplying that by λ to yield the regularization term. The motivation behind this is to create an adaptive form of weight decay. The formalization is given in equations (2) and (3):

$$\lambda = \sum_{i=1}^{k} \sigma\left(w_i\right), \tag{2}$$

where k is the number of rows in the weight matrix, i is the i^{th} row of the weight matrix, and σ represents the standard deviation as given below:

$$\sigma(w) = \sqrt{\frac{1}{n}\left\{\sum_{j=1}^{n} w_j^2 - \left(\sum_{j=1}^{n} w_j\right)^2\right\}}, \tag{3}$$

where λ is the regularization parameter that acts as a penalty to prevent weights from reaching high values during the training process and n is the number of columns in each i^{th} row of the weight matrix.

Training models were trained using the Nesterov ADAM optimizer, with tanh activation functions. The model was trained over 100 epochs with a batch size of 32. The labeled data were classified with a feedforward network.

4.3. System Architecture. The proposed IDS algorithm is embedded in the SDN controller as an application since the SDN controller is not designed to analyze network traffic in depth like an IDS. This paper focuses on the use of the SDN paradigm as a network infrastructure for the IDS. The architecture of SDN is given in Figure 1. This architecture has three parts:

(i) *Flow Collector*. This module collects the packet from the flow and extracts all the information required for the trained IDS algorithm such as the protocol used, the IP addresses, and the port. All these details are passed to the controller.

(ii) *Anomaly Detector*. This module collects the data from the flow collector and loads the proposed IDS algorithm to check the packet for anomalies. The proposed IDS based on the new regularizer is the heart of this module.

(iii) *Anomaly Mitigator*. This module is dependent upon the decision of the anomaly detector. The anomaly mitigator will either drop or forward the packet based on the results conveyed by the detector.

4.4. Datasets. In this paper, we targeted three benchmark datasets to test the proposed model. A brief description of these datasets is given below:

(1) *KDD Cup 1999 Dataset.* This was the first dataset used in the third international KDD tools competition that was held with the collaboration of KDD-99. The data consist of "normal" data and anomalous data broken up into four different classes based on the attack types given in Table 1.

(2) *NSL-KDD Dataset*. This dataset is a refined version of the KDD Cup 1999 dataset and has the following advantages over the KDD Cup 1999 dataset:

(i) There were no redundant data in the training set, so the classifier does not get biased toward duplicate records

(ii) There were no redundant data in the testing set, so the trained classifier performance is not disturbed by the duplicate records

(iii) Records selected from each difficulty level are inversely proportional to that of the KDD dataset. Therefore, the performance of different machine learning methods varies widely making it a more complex dataset

(3) *UNSW-NB15 dataset.* The UNSW-NB15 dataset includes nine different modern attack types and a wide variety of normal activities with 49 features, inclusive of the class label, over more than 2.5 million records. There are six categories of features: flow features, basic features, content features, time features, additional generated features, and labeled features. These features are further divided into subcategories. For example, additional generated features have two subgroups called general purpose features and connection features.

In the dataset, features from the 36th column to the 40th column represent general purpose features, while the 41st feature to 47th feature are connection features [22].

The attacks, which are known by labels, are classified into the nine types given below:

(i) Reconnaissance
(ii) Shellcode
(iii) Exploit
(iv) Fuzzers
(v) Worm
(vi) DoS
(vii) Backdoor
(viii) Analysis
(ix) Generic

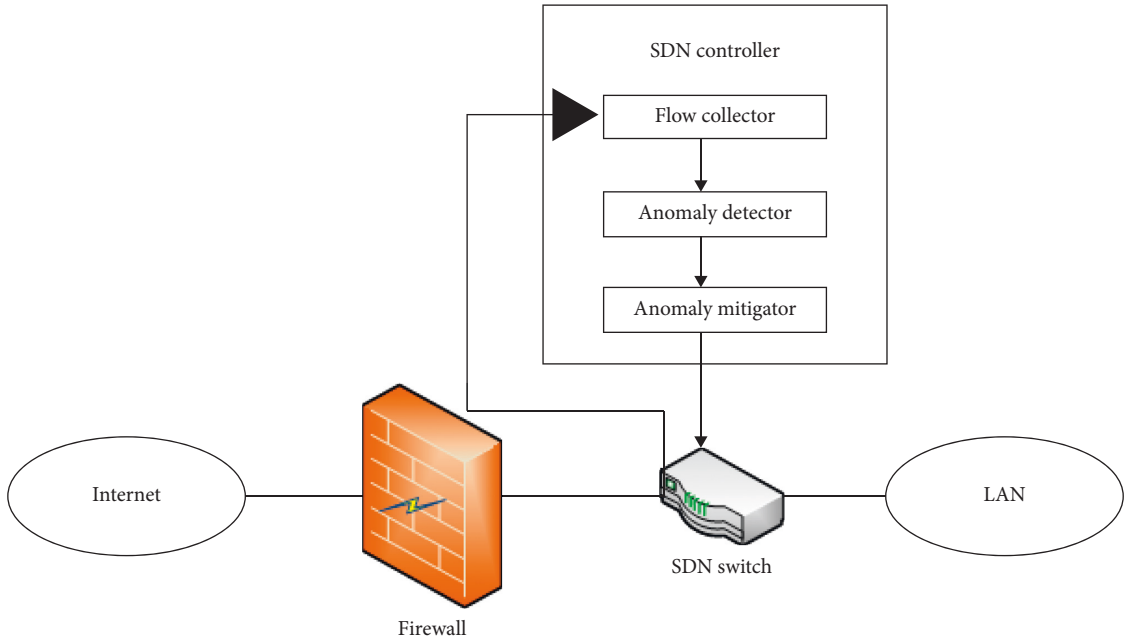

Figure 1: A high-level view of the IDS architecture for the SDN environment.

Table 1: Attack classes based on different attack types.

Attack class	Attack types
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop
Probe	Satan, Nmap, IPsweep, Portsweep
U2R	Loadmodule, Perl, RootKit, Bufferoverflow
R2L	Ftpwrite, GuessPasswd, Imap, Warezclient, Warezmaster, multihop, Phf, Spy

4.5. Data Preprocessing. The RNN only takes numerical data for training and testing. Therefore, the first step is to convert textual and nominal data into numerical data. For this purpose, the following steps were performed:

(1) For the NSL-KDD dataset, 38 nonnumerical features were converted to numerical features. Three features (*protocol type*, *service*, and *flag*) were assigned unique integer values. The four distinct attack types (*R2L*, *DoS*, *U2R*, and *Probe*) were assigned integer values 1, 2, 3, and 4, respectively. The normal category was assigned the value 0.

(2) For the UNSW-NB15 dataset, there are already nine specific categories defined in the dataset as explained in the previous section. We only performed scaling/normalizing of the values for this dataset.

4.6. Data Scaling. All the datasets have different attributes of numerical and nonnumerical types. Attributes are encoded to numerical values and are scaled according to the following mathematical equation:

$$X_i = \frac{X_i - \min(X_i)}{\max(X_i) - \min(X_i)}, \quad \text{for } i = 1, \ldots, n, \tag{4}$$

where n represents the number of records and X represents a specific column in the dataset. Duplicate records were removed from the dataset to prevent the classifier from producing biased results.

5. Intrusion Detection Performance Analysis

5.1. Evaluation Metrics. For all datasets, the AUC-ROC curve, true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and average validation accuracy were computed. Apart from these measures, the F-Score values were also calculated. All these measures can easily be calculated from the confusion matrix that represents the true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The mathematical representation of each measure is given below:

(i) *Accuracy.* It is the ratio between true intrusions detected vs overall detection:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{5}$$

(ii) *True Positive Rate.* It is the percentage of intrusions that were actually intrusions and correctly detected:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{6}$$

(iii) *True Negative Rate.* It is the percentage of packets that were nonintrusions and correctly classified as nonintrusions:

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}. \quad (7)$$

(iv) *Precision.* It is the ratio between the true anomalous packets vs total packets that were marked as intrusions:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \quad (8)$$

(v) *F-Score.* It is the harmonic mean of true positive rate and precision:

$$\text{F} - \text{Score} = \frac{2}{(1/\text{precision}) + (1/\text{TPR})}. \quad (9)$$

5.1.1. KDD Cup 1999 Dataset. For the KDD Cup 1999 dataset, the average validation accuracy achieved is 99.52% with 100% AUC-ROC curve. Our proposed model has produced 99.5% and 99.4% TPR and TNR, respectively. The validation accuracy and AUC-ROC are shown in Figures 2 and 3, respectively.

5.1.2. NSL-KDD Dataset. For the NSL-KDD dataset, the RNN model embedded with the new regularizer was trained for 100 epochs. The TPR and TNR for this dataset were computed to be 97.43% and 97.35%, respectively. The average validation accuracy achieved was 97.39%, and the AUC-ROC curve had the value 99.7%, as shown in Figures 4 and 5, respectively.

5.1.3. UNSW-NB15 Dataset. For the UNSW-NB15 dataset, the RNN model with the novel regularizer was trained for 100 epochs. The TPR and TNR for this dataset were computed as 98.4% and 98.13%, respectively. The average validation accuracy achieved was 99.9%. The AUC-ROC curve had a value of 1.0, as shown in Figures 6 and 7, respectively.

5.2. Detection Performance Measurement. We compared the performance results of our proposed model with results from the literature. From Table 2, we can clearly see that our proposed method outperforms other models in all the evaluation metrics for all classes and has the potential to be used for anomaly detection in an SDN environment. In Table 2, we compare precision (Pr), true positive rate (TPR), and F-Score for legitimate and anomaly classes. We computed all these measures for the three datasets we used. For legitimate classes, precision was 94.3%, 92.2%, and 93.5% for the KDD Cup 1999, NSL-KDD, and UNSW-NB15 datasets, respectively. Precision tells us the positive predictive values. In other words, it shows the ratio of the total positive values in the dataset, and the positive instances predicted by the classifier. Therefore, we can say that our classifier is good enough to detect nonanomalous

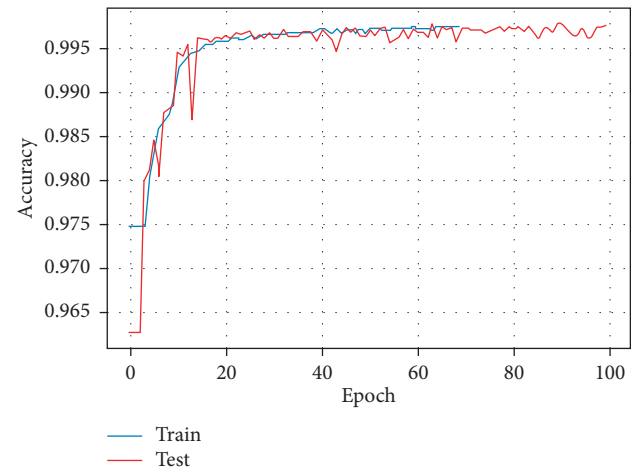

Figure 2: Train and test accuracy for the KDD Cup 1999 dataset while training the RNN embedded with the proposed regularizer.

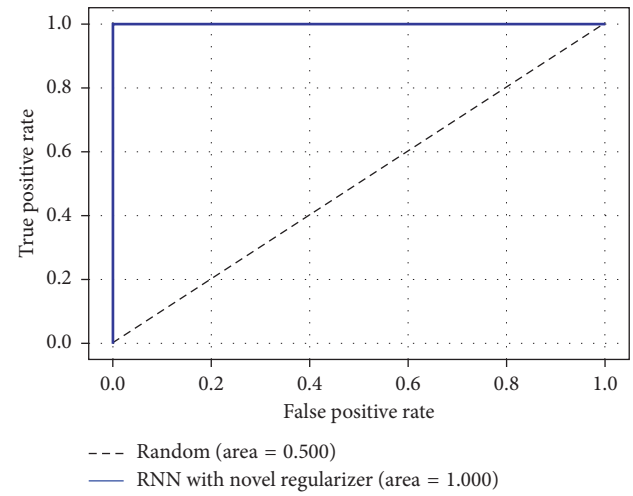

Figure 3: AUC-ROC curve for the KDD Cup 1999 dataset after training the RNN with the proposed regularizer.

classes. Similar results were achieved for anomalous classes. The TPR measure represents the sensitivity of the classifier. The values indicate that the model is very accurate. Sometimes, TPR and precision are not considered good measures. F-Score provides an alternate view representing the harmonic mean of precision and sensitivity. The data in Table 2 indicate that our model achieved higher values for F-Score and, therefore, can be considered efficient. Hence, our model is more accurate and precise than other methods.

In addition, we compared our results with classical and neural network based methods used by different researchers. In terms of accuracy, our model outperforms all listed methods used for anomaly detection in an SDN environment (Table 3). For the three datasets, we achieved satisfactory accuracy. This accuracy is high for the cost of overhead and latency as can be seen from Figures 8 and 9. The throughput is low and latency is high, but the intrusion detection accuracy is also high.

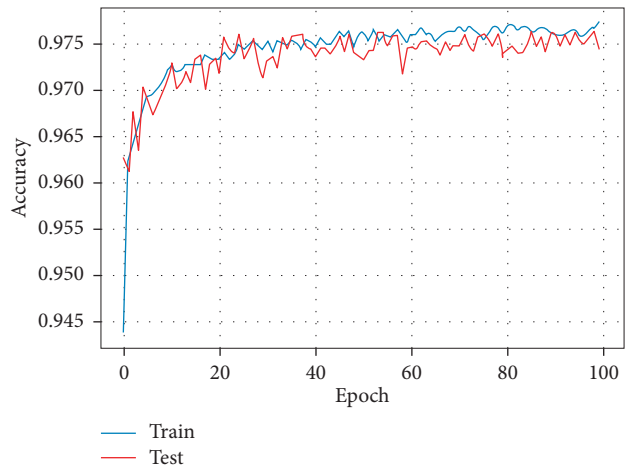

Figure 4: Train and test accuracy for the NSL-KDD dataset while training the RNN embedded with the proposed regularizer.

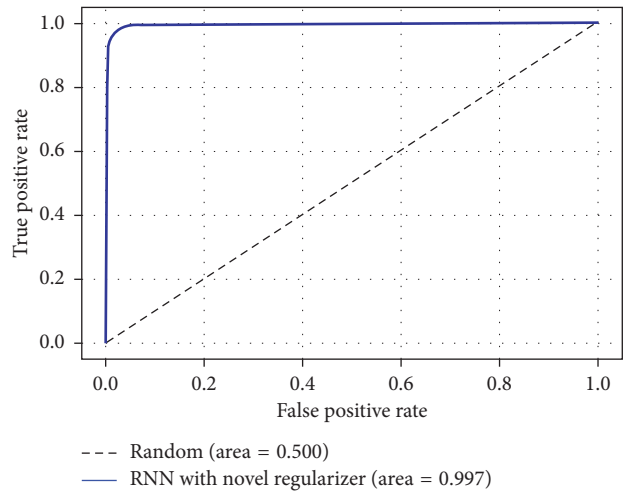

Figure 5: AUC-ROC curve for the NSL-KDD dataset after training the RNN with the proposed regularizer.

6. Network Performance Analysis

In this section, we evaluate the effect of our RNN on network performance. The evaluation testbed is described in the first part, and then the network performance evaluation is presented.

6.1. Experimental Setup. The mininet emulator was used to test the learning model in the network as an intrusion detection system based on the new regularization technique. Mininet is an open source Python-based network emulator that is used to create a virtual networking topology connecting virtual hosts via various devices such as switches, links, and controllers. It runs Linux network software and can support OpenFlow for custom routing and SDN.

As mininet needs to be installed on a Linux server, we chose Oracle VM VirtualBox to carry out our simulations. The simulation was conducted on a system with 64-bit Ubuntu 18.04 LTS on a Core-i7 with 16 GB of RAM. The

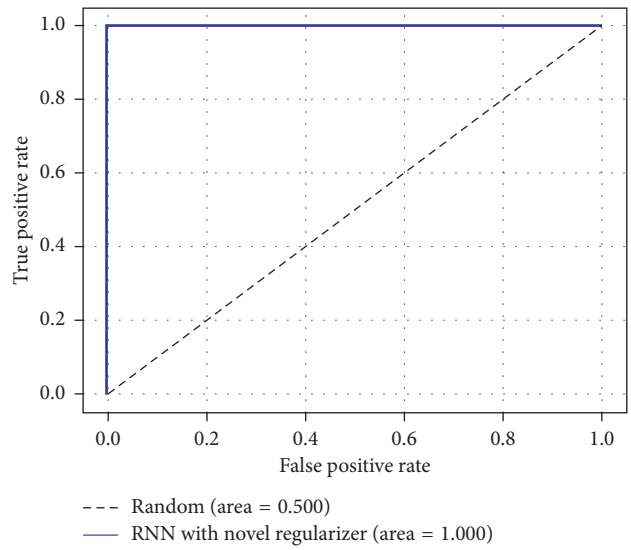

Figure 6: AUC-ROC curve for the UNSW-NB15 dataset after training the RNN embedded with the proposed regularizer.

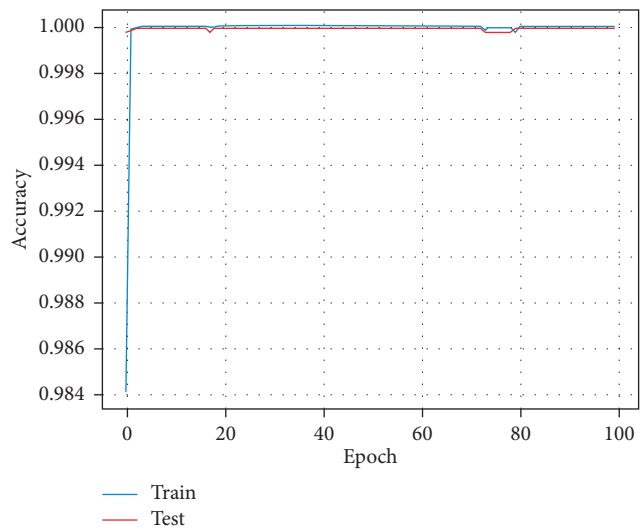

Figure 7: Accuracy graph for the UNSW-NB15 dataset after training the RNN with the proposed regularizer.

performance of the controller embedded with the proposed model was tested on various numbers of OpenFlow switches emulated by Cbench in mininet. The performance of the proposed model was compared with the POX and Beacon OpenFlow controllers after training on the NSL-KDD and UNSW-NB15 datasets with the throughput and latency running modes.

6.2. Analysis of Results. We conducted an analysis of our results in terms of throughput and latency and then compared it with existing POX and Beacon controllers. Figure 8 shows the throughput for POX, Beacon, and our proposed work. Based on our analysis, there is a slight difference in throughput between Beacon and the proposed work when the number of switches is 8, 128, or 256. When compared

Table 2: Comparison of detection performance.

Algorithms	Legitimate class			Anomaly class		
	Pr (%)	TPR (%)	F-Score (%)	Pr (%)	TPR (%)	F-Score (%)
Our work (KDD Cup 1999)	**94.3**	**99.5**	**93.01**	**95**	**96**	**92.21**
Our work (NSL-KDD)	**92.2**	**97.43**	**91**	**94.57**	**93.5**	**91.13**
Our work (UNSW-NB15)	**93.5**	**98.4**	**94.03**	**95.2**	**97.01**	**92.3**
VanilaRNN	43	90	58	57	10	17
SVM	71	32	44	64	90	75
DNN	67	89	76	88	66	76
GRU-DNN	87	89	88	91	90	90

Table 3: Comparison to other methods.

Method	Accuracy (%)
DNN [20]	75.75
SVM [23]	69.53
NB trees [23]	82.02
GRU-RNN [21]	89
Our work (KDD Cup 1999)	**99.5**
Our work (NSL-KDD)	**97.39**
Our work (UNSW)	**99.9**

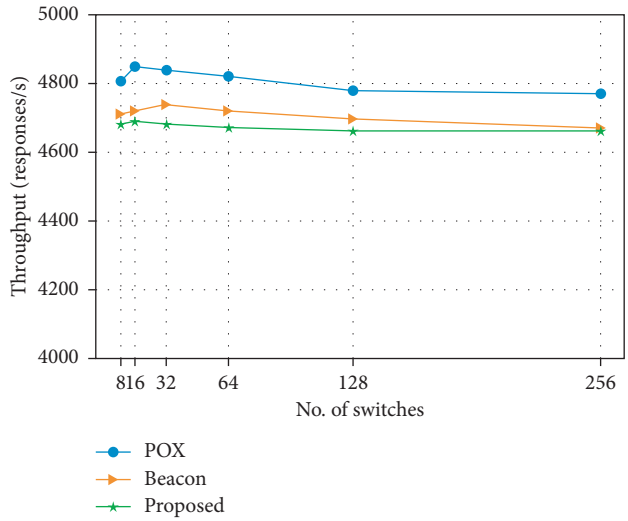

Figure 8: Throughput of existing controllers vs our proposed model on the NSL-KDD dataset.

with the POX controller, the performance decreased by 2.702% for 8 switches. For 128 and 256 switches, the performance dropped 3.32% and 3.51%, respectively, when compared with the POX controller.

We observed minimal difference in the performance of our proposed algorithm when compared with the Beacon controller. The average decreases in throughput of 0.63%, 1.22%, and 2.33% are seen, respectively, for 8, 128, and 256 switches. Hence, our proposed model gave satisfactory results in terms of throughput.

Figure 9 illustrates latency on the NSL-KDD dataset. In this figure, it can be seen that performance decreases regularly as the number of switches increases. This decrease can be ascribed to the extra responsibility of analyzing and checking packets.

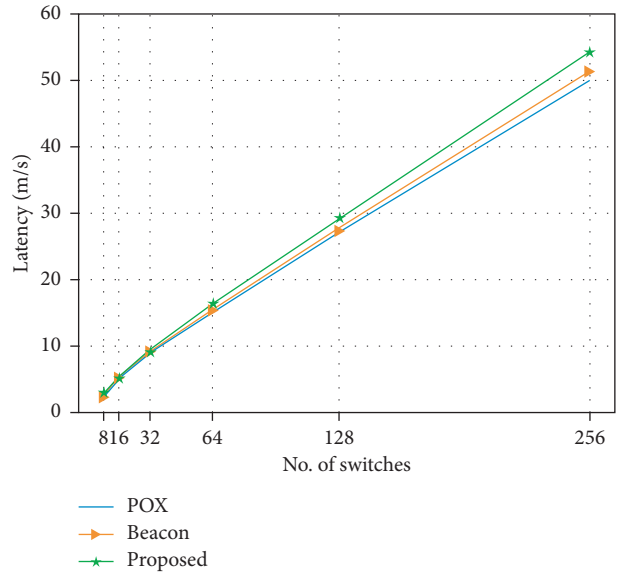

Figure 9: Latency of existing controllers vs our proposed model on the NSL-KDD dataset.

Similarly, the simulations were carried out for the model trained on the UNSW-NB15 dataset, and the results were analyzed in terms of throughput and latency. The results were compared with existing POX and Beacon controllers. Figure 10 shows the throughput for POX, Beacon, and our proposed work.

As can be seen, there is a significant difference in throughput of the Beacon controller and the proposed work when the number of switches is 32, 64, or 256. When compared with the POX controller, the performance decreased by 1.4% for 32 switches. As for 128 and 256 switches, the performance dropped 3.4% and 3.6%, respectively, when compared with the POX controller. When compared with the Beacon controller, there is minimal difference in the performance of our proposed algorithm. Average decreases in throughput of 1.6%, 1.3%, and 2.4% are seen, respectively, for 32, 128, and 256 switches. Hence, from these results, it is seen that our proposed model, trained on the UNSW-NB15 dataset, slightly lags in throughput but with a higher accuracy than the other models (Table 3).

For the latency presented in Figure 11, we observed the same findings as above. The performance decreases as the number of switches increases, just like for the other models. This decrease is due to the extra responsibility for checking and

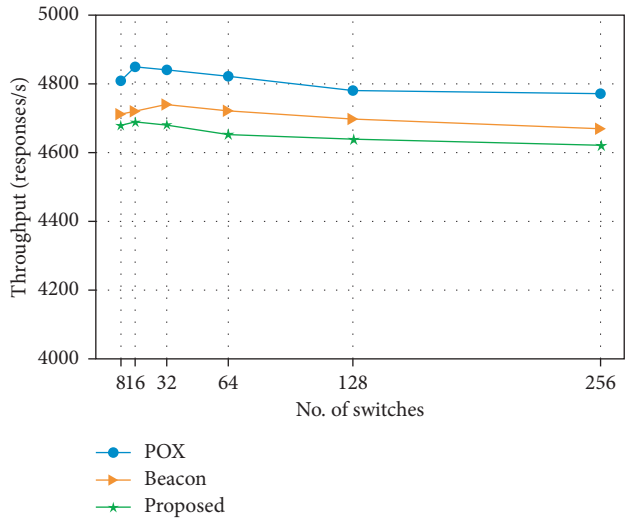

Figure 10: Throughput of existing controllers vs our proposed model on the UNSW-NB15 dataset.

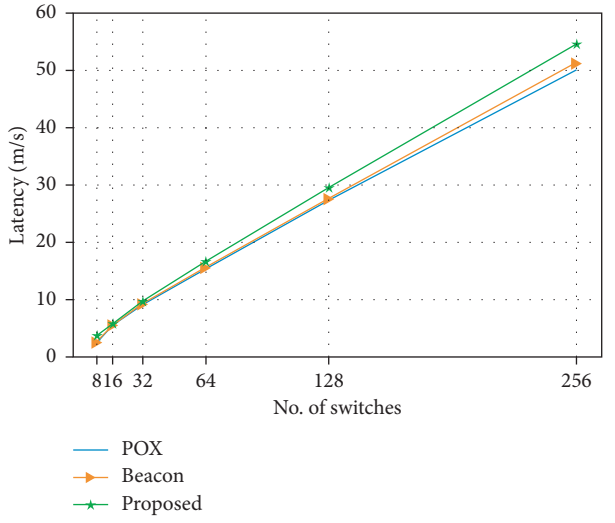

Figure 11: Latency of existing controllers vs our proposed model on the UNSW-NB15 dataset.

analyzing packets for intrusions. After analyzing the results, we can say positively that the proposed model has the potential for real-time anomaly detection in an SDN environment.

Based on the analysis of the results above, we can see that there is a trade-off between the throughput or latency and security. When one of them increases, the other will decrease. Hence, it is the responsibility of the network administrators to tune the network according to its requirements, either to make it more secure by adding the algorithm for analyzing the packet or make it fast.

7. Conclusion

In this paper, we present an anomaly based IDS in an SDN environment using an RNN with a new regularization algorithm. We train the RNN-SDR on the KDD Cup 1999, NSL-KDD, and UNSW-NB15 datasets and show that our model outperforms other state-of-the-art algorithms with an accuracy of 99.5%, 97.39%, and 99.9% for the KDD Cup 1999, NSL-KDD, and UNSW-NB15 datasets, respectively. Our scheme uses a minimum number of features compared with other state-of-the-art approaches. This makes the model more computationally efficient for real-time detection. In addition, the network performance evaluation shows that our proposed approach slightly affects the controller performance. This implies a trade-off in either selecting security or speed. Nevertheless, our model is practical for implementation in the context of an SDN.

Data Availability

All relevant data are included within the article.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] "Software defined networking definition," 2017, https://www.opennetworking.org/sdn-definition/.

[2] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pp. 55–60, ACM, New York, NY, USA, August 2013.

[4] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[5] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[6] R. Sommer and V. Paxson, "Outside the closed world: on using machine learning for network intrusion detection," in *Proceedings of the 2010 IEEE symposium on security and privacy*, pp. 305–316, IEEE, Berkeley/Oakland, CA, USA, May 2010.

[7] G. Karatas, O. Demir, and O. Koray Sahingoz, "Deep learning in intrusion detection systems," in *Proceedings of the 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*, Ankara, Turkey, December 2018.

[8] E. Steinbrecher, "Intrusion detection using MDL clustering," US. Patent No. 9,106,689, 2015.

[9] M. S. Parwez, D. Rawat, and M. Garuba, "Big data analytics for user activity analysis and user anomaly detection in mobile wireless network," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, 2017.

[10] L. Nie, D. Jiang, and Z. Lv, "Modeling network traffic for traffic matrix estimation and anomaly detection based on Bayesian network in cloud computing networks," *Annals of Telecommunications*, vol. 72, no. 5-6, pp. 297–305, 2017.

[11] J.-h. Bang, Y.-J. Cho, and K. Kang, "Anomaly detection of network-initiated LTE signaling traffic in wireless sensor and

actuator networks based on a Hidden semi-Markov model," *Computers & Security*, vol. 65, pp. 108–120, 2017.

[12] S. T. Ikram and A. K. Cherukuri, "Improving accuracy of intrusion detection model using pca and optimized svm," *Journal of Computing and Information Technology*, vol. 24, no. 2, pp. 133–148, 2016.

[13] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks (LCN)*, pp. 408–415, IEEE, Denver, CO, USA, October 2010.

[14] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," Lecture Notes in Computer Science, in *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*, pp. 161–180, Springer, Menlo Park, CA, USA, September 2011.

[15] R. Kokila, S. T. Selvi, and K. Govindarajan, "Ddos detection and analysis in sdn-based environment using support vector machine classifier," in *Proceedings of the 2014 Sixth International Conference on Advanced Computing (ICoAC)*, pp. 205–210, IEEE, Chennai, India, December 2014.

[16] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, and N. H. Thanh, "Openflowsia: an optimized protection scheme for software-defined networks from flooding attacks," in *Proceedings of the 2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pp. 13–18, IEEE, Ha Long, Vietnam, July 2016.

[17] A. AlEroud and I. Alsmadi, "Identifying cyber-attacks on software defined networks: an inference-based intrusion detection approach," *Journal of Network and Computer Applications*, vol. 80, pp. 152–164, 2017.

[18] S. M. Mousavi and M. St-Hilaire, "Early detection of ddos attacks against sdn controllers," in *Proceedings of the 2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 77–81, IEEE, Garden Grove, CA, USA, February 2015.

[19] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," 2016, https://arxiv.org/abs/1611.07400.

[20] T. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (WINCOM'16)*, Fez, Morocco, October 2016.

[21] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in SDN-based networks," in *Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Montreal, QC, Canada, June 2018.

[22] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection," in *Proceedings of the MilCIS-IEEE Stream, Military Communications and Information Systems Conference*, IEEE publication, Canberra, Australia, November 2015.

[23] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*, Ottawa, ON, Canada, July 2009.