
Book Section:
Tang, TA, McLernon, D orcid.org/0000-0002-5163-1975, Mhamdi, L et al. (2 more authors) (2019) Intrusion Detection in SDN-Based Networks: Deep Recurrent Neural Network Approach. In: Alazab, M and Tang, M, (eds.) Deep Learning Applications for Cyber Security. Advanced Sciences and Technologies for Security Applications . Springer , Cham, Switzerland , pp. 175-195. ISBN 978-3-030-13056-5

https://doi.org/10.1007/978-3-030-13057-2_8

Intrusion Detection in SDN-based Networks: Deep Recurrent Neural Network Approach

Tuan Anh Tang, Des Mclernon, Lotfi Mhamdi, Syed Ali Raza Zaidi and Mounir Ghogho

Abstract Software Defined Networking (SDN) is emerging as a key technology for future Internet. SDN provides a global network along with the capability to dynamically control network flow. One key advantage of SDN, as compared to the traditional network, is that by virtue of centralized control it allows better provisioning of network security. Nevertheless, the flexibility provided by SDN architecture manifests several new network security issues that must be addressed to strengthen SDN network security. So, in this paper, we propose a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) enabled intrusion detection system for SDN. The proposed approach was tested using the NSL-KDD and CICIDS2017 dataset, and we achieved an accuracy of 89% and 99% respectively with low dimensional feature sets that can be extracted at the SDN controller. We also evaluated network performance of our proposed approach in terms of throughput and latency. Our test results show that the proposed GRU-RNN model does not deteriorate the network performance. Through extensive experimental evaluation, we conclude that our proposed approach exhibits a strong potential for intrusion detection in the SDN environments.

Tuan Anh Tang, Des Mclernon, Lotfi Mhamdi, Syed Ali Raza Zaidi
University of Leeds, Leeds, LS2 9AN, UK
e-mail: eltat@leeds.ac.uk, d.c.mclernon@leeds.ac.uk, l.mhamdi@leeds.ac.uk, s.a.zaidi@leeds.ac.uk

Mounir Ghogho
International University of Rabat, Morocco
e-mail: m.ghogho@leeds.ac.uk

1 Introduction

1.1 Motivation

The current Internet architecture has been developed for nearly three decades and is now becoming an increasingly complex system. It is largely decentralised, autonomous and built from a large number of network devices such as routers, switches and numerous types of middleboxes (e.g., firewall, load balancing, etc.) with several complex protocols implemented on each of them. These network devices are traditionally developed by several manufacturers. Each manufacturer has their own designs, firmware and other software to operate their own hardware in a proprietary and non co-operative way. Consequently, current Internet architecture lacks the agility to respond to ever changing demands and the dynamic nature of modern day applications. Software Defined Networking (SDN) [21] is introduced as a promising architecture, enabling scalability and unprecedented flexibility in the configuration and deployment of network services. The separation of control and data planes provides more flexibility and greater control over the traffic flows. Nevertheless, as highlighted in [13], the SDN architecture also introduces various security issues pertaining to the control plane, the control-data interface and the control-application interface. Recently, as mentioned in [14] and [26], SDN security has become a serious concern and attracted significant interest.

An intrusion detection system (IDS) is one of the most important network security tools. IDSs can be broadly classified into two categories : Signature-based IDS and Anomaly-based IDS. The signature-based IDS uses the signature database of the previous attacks to detect the new attacks. This system gives a low false alarm rate but it fails to detect zero-day attacks. The operational efficiency of the signature-based IDS is strongly coupled with the integrity and freshness of signature information available in databases. Maintaining such databases incurs huge operational overhead and is difficult if not impossible to realize in real-time. The anomaly-based IDS tries to identify the observation that deviates from the baseline model. Thus, this system can detect zero-day attacks better than the signature-based IDS. Various approaches have been proposed for intrusion detection like artificial neural networks (ANNs), support vector machines (SVMs), and Bayesian approaches. However, these techniques have a quite high false alarm rate and associated computational cost as mentioned in [28]. Recently, Deep Learning (DL) has emerged as a new approach and achieved a huge success in computer vision and language processing areas. DL has the ability to process raw data and learn the high-level features on its own, and so DL delivers higher accuracy than traditional machine learning techniques. The flow-based nature of SDNs enables network information acquisition in real-time via the OpenFlow [16] protocol. Consequently, flow-based intrusion detection systems have recently attracted significant attention in the context of SDNs.

1.2 Contribution

Following the trajectory of current research, we believe that deep recurrent neural networks can potentially offer better solutions for implementation of IDS under the context of SDN. Recurrent Neural Networks (RNNs) have shown great success in language modelling, text generating and speech recognition. We believe that the RNN is a powerful technique that can represent the relationship between a current event and previous events and then enhance the anomaly detection rate. In this chapter, a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) is proposed to detect the anomaly traffic traces.

In summary, the major contributions of this chapter are the following:

- We introduce an IDS in the SDN environment using GRU-RNN. To the best of our knowledge, this is the first attempt to use GRU-RNN for intrusion detection in the SDN environment.
- Our GRU-RNN approach yields a detection rate of 89% in the NSL-KDD dataset using a minimum number of features compared to other state-of-the-art approaches. This gives our approach significant potential for real time detection. The GRU-DNN achieves an impressing detection rate of 99% dealing with DDoS attacks in the CICIDS2017 dataset.
- We also evaluate the network performance of our proposed approach in the SDN environment. Then results show that our approach does not significantly degrade the SDN controller's performance.

The remainder of this paper is organized as follows. Section II presents a literature review. In Section III, we give a system description. Section IV presents the intrusion detection performance and network performance analysis. Finally, Section V concludes the paper and presents future work.

2 Literature Review

2.1 What is Software Defined Networking?

The idea of programmable networks has been proposed as a way to facilitate the evolution of the conventional network. The idea of programmable networks and decoupled control logic has been around for several years. In general, SDN is built under four principles:

- *Seperation of Control and Data Plane*: these planes must be logically separated and connected via an interface. The control aspect is removed from forwarding devices and delegated to an external entity.
- *Network Programmabitity*: The provision of open API is a core aspect of the SDN architecture. Software and scripts should be able to access, configure, and modify network elements with ease.

- *Network Abstraction*: the view of the network is virtualized for any elements of a higher hierarchy. Services and applications are aware of the state of the whole network, but physical attributes and resources are irrelevant for configurations and computations.
- *Logically Centralized Control*: all forwarding devices of a domain are linked to a controlling entity and are subjected to its enacted policies.

SDN is defined by the Open Networking Foundation (ONF) which was founded in 2011 by Microsoft, Google, Facebook, Yahoo, Verizon and Deutsche Telekom. As of 2015 the organization has more than 150 industry members and receives endorsement by several network equipment vendors such as Cisco, Dell, Brocade and HP. An SDN architecture decouples the network control and forwarding functions enabling the network control to become directly programmable. The separation of the control plane from the data plane lays the ground for the SDN architecture. Network switches become simple forwarding devices and the control logic is implemented in a physical or logical centralized controller. The logical centralized controller dictates the network behaviour and offers several benefits. Firstly, it is simpler and less error-prone to modify network policies through software from a single place without reconfiguring individual devices. Secondly, a control program can automatically react to dynamic changes of the network and thus maintain the high level policies in place. Thirdly, the centralised control logic has global knowledge of the whole network, including the network topology and the state of the network resources, thus giving the flexibility and simplifying the development of more sophisticated network functions. For example, the controller can dynamically adjust flow tables to avoid congestion or apply different routing algorithms to different types of traffic.

The SDN architecture is divided into three layers: infrastructure layer, control layer and application layer.

- Infrastructure layer (Tier-1): This layer consists of the forwarding hardware such as switches/routers.
- Control layer (Tier-2): Network intelligence is installed in software-based logically centralized SDN controllers. The control layer regulates and manages forwarding hardware. The controller is the core of SDN networks. It lies between network devices at the one end and the applications at the other end. The SDN controller takes the responsibility of establishing every flow in the network by installing flow entries on switch devices.
- Application layer (Tier-3): Application and services take advantage of the control and infrastructure layer. Conceptually the application layer is above the control layer and enables easy development of network applications. These applications perform all network management tasks . Some examples of SDN application are load balancers, network monitors, and intrusion detection systems (IDS).

2.2 Security in SDN

The SDN concept was originally designed with significant advantages over the traditional networking. Even though SDN brings significant advantages to network security, it also introduces new targets for potential attacks. The main causes of concern actually lie in SDN's main benefits: network programmability and control logic centralization. These capabilities actually introduce new fault and attack planes, which open the doors for new threats that did not exist before or were harder to exploit. According to Kreutz et al. [14], there are seven main potential threat vectors identified in SDN and summarized in Table 1.

Table 1 SDN Threat Vectors

No.	Threat Vector
1	Forged or faked traffic flow
2	Attacks on vulnerabilities in switches
3	Attacks on control plane communications
4	Attacks on and vulnerabilities in controllers
5	Lack of mechanisms to ensure trust between the controller and management application
6	Attacks on and vulnerabilities in administrative stations
7	Lack of trusted resources for forensics and remediation

Among these seven threat vectors, number 3,4 and 5 are not present in traditional networking. They are specific to SDNs as they arise from the separation of the control and data plane and the logical centralization of the controllers. Other vectors were already presented in traditional networking. Threat vector number 5 would have the most severe impact on SDN architecture because it could affect the entire network. The controller emerges as a potential single point of attack. Attackers can attack vulnerabilities of controllers and run several dangerous scripts. Therefore, if there are no security settings to protect the controller, it would be under the control of attackers. The controller provides an interface for SDN applications to manage the network. However, this also give chances for malicious SDN applications to take over the network because of the lack of trust between the controller and SDN applications. Malicious hosts also can cause severe damage to the SDN architecture. Malicious hosts can perform Denial of Service (DoS) attacks to controllers and other hosts or network topology poisoning. DoS attack is one of the most dangerous attacks in SDN. It can be done by flooding the network with a large number of forged packets. These packets would trigger the switches to send a large number of requests to the controller for new flow rules. Therefore the control channel bandwidth and the controller CPU resources will be heavily consumed. As a result, the controller would respond slowly to legitimate requests. At the same time, the switches would also suffer from traffic congestion because the packets could quickly exhaust the memory of the flow table storage in the switches. Compromised switches not only have the same capabilities as the malicious hosts, but they

are also capable of performing more dynamic and severe attacks. Firstly, they can be used for traffic eavesdropping. Both data and control flows passing through the compromised switches can be replicated and sent to the attacker for further processing. Furthermore, the attacker can interfere with the control traffic passing through the compromised switches to perform man-in-the-middle attacks. By doing so, the attacker can act as the controller to some target switches.

2.3 Related Work

Researchers have employed classical machine learning approaches such as SVM, K-Nearest Neighbour (KNN), ANN, Random Forest, etc., for intrusion detection for several years. These proposed methods have achieved various degrees of success while also exhibiting some inherent limitations. Parwez et al. [22] employ K-means and hierarchical clustering to detect anomalies in call detail records (CDRs) of mobile wireless networks data. In [19], Bayesian networks are employed for anomaly and intrusion detection such as DDoS attacks in cloud computing networks. Bang et al. [2] use Hidden semi-Markov models to detect LTE signalling attacks. Principal Component Analysis (PCA) and SVM are also combined in [10] to detect intrusions. PCA is used for dimensional reduction on network data and SVM is used to detect intrusion on those data. These work only focus on traditional network with a large set of features that cannot be applied to SDNs. These classical mechanisms are still employed for intrusion detection in the context of SDN.

Braga et al. [3] present a lightweight approach using a Self Organizing Map (SOM) to detect Distributed Denial of Service (DDoS) attacks in the SDN. This approach based on six traffic flow features gives quite high detection accuracy. In [17], the authors use four traffic anomaly detection algorithms (threshold random walk with credit based rate limiting, rate limiting, maximum entropy and NETAD) in the SDN environments. The experiments indicate that these algorithms perform better in the SOHO (Small Office/Home Office) network than in the ISP (Internet Service Provider) and they can work at line rates without introducing any new performance overhead for the home network.

In the literature, the DDoS attacks are some of the most focused attacks for the SDN context because of their severe effects. The controller is a single point of failure in the SDN architecture. Therefore, if intruders trigger the DDoS attacks on the controller and take control of it, they can also control the whole network. Support Vector Machine (SVM) is also a quite popular algorithm for its high detection accuracy. In [12] and [24], SVM is used to detect DDoS attacks quite efficiently. Winter et al. [31] train a one-class SVM with a malicious dataset in order to reduce the false alarm rate. K-Nearest Neighbour and graph theory are combined to classify DDoS attacks from benign flows in SDNs by AlEroud et al. in [1]. Mousavi et al. [18] propose an early DDoS attack detection method against the SDN controller based on the variation of the entropy of the flow's destination IP addresses. They assume that the destination IP addresses are evenly distributed for the benign flows, while the

malicious flows are destined for a small amount of hosts. Thus, the entropy drops dramatically when any attack happens. In [20], the authors propose a deep learning based approach using a stacked autoencoder (SAE) for detecting DDoS attacks in the SDN. They achieve a quite high accuracy rate and low false alarm rate on their own dataset.

In 2016, we applied a Deep Neural Network (DNN) under the context of SDNs to train and test the NSL-KDD dataset [29]. We obtained a potential accuracy of 75.75% with just six basic features. In this paper, we continue this trend by using GRU-RNN to improve the detection accuracy and reduce the false alarm rate of the system.

3 Methodology/System Description

In this section, the Recurrent Neural Network (RNN) and Gated Recurrent Units (GRUs) are briefly reviewed. Then the architecture of the SDN-based IDS is described in detail.

3.1 Recurrent Neural Networks

A RNN is an extension of a conventional feed forward neural network. In general, a neural network architecture is as shown in Fig. 1.

The RNN is called "recurrent" because it performs the same task for every element of a sequence, with the output being dependent on the previous computations. Mathematically, the hidden states of the RNN are computed as:

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h), t = 1, 2, \ldots, T, \tag{1}$$

where $\sigma(\cdot)$ is a non-linearity function, $\mathbf{x}_t$ is an input row vector at time t, $\mathbf{h}_t$ is a hidden state row vector at time t, $\mathbf{W}$ is an input to hidden weight matrix, $\mathbf{U}$ is a hidden to hidden weight matrix, and $\mathbf{b}_h$ is a row vector bias term.

The Backpropagation Through Time (BPTT) algorithm is used for training the RNN. However, BPTT for the RNN is usually difficult due to a problem known as vanishing/exploding gradient [9]. Long Short Term Memory (LSTM) [8] networks and GRUs [5] were proposed to solve this problem and are among the most widely used models in DL.

GRUs are selected in our research because of their simplicity and faster training phase compared to LSTMs [7].

For a GRU the activation $\mathbf{h}_t$ is computed differently from (1) as follows:

$$\mathbf{h}_t = (1 - \mathbf{z}_t)\mathbf{h}_{t-1} + \mathbf{z}_t\tilde{\mathbf{h}}_t, t = 1, 2, \ldots, T, \tag{2}$$

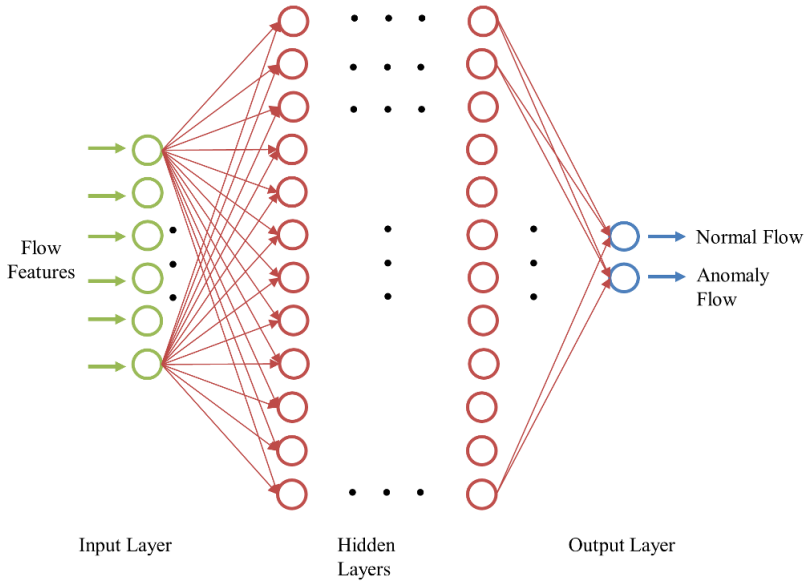

Fig. 1 The Deep Neural Network Structure

where an update gate $\mathbf{z}_t$ defines how much of the previous memory to keep and $\tilde{\mathbf{h}}_t$ is the candidate activation. The update gate is computed by

$$\mathbf{z}_t = \sigma(\mathbf{x}_t \mathbf{W}_z + \mathbf{h}_{t-1} \mathbf{U}_z). \tag{3}$$

The candidate activation $\tilde{\mathbf{h}}_t$ is computed by

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{x}_t \mathbf{W}_h + (\mathbf{h}_{t-1} \odot \mathbf{r}_t) \mathbf{U}_h), \tag{4}$$

where a reset gate $\mathbf{r}_t$ determines how to combine the new input with the previous activation vector and $\odot$ is an element-wise vector multiplication. The reset gate is computed by

$$\mathbf{r}_t = \sigma(\mathbf{x}_t \mathbf{W}_r + \mathbf{h}_{t-1} \mathbf{U}_r), \tag{5}$$

where $\sigma(\cdot)$ is again a non-linearity function in (3) and (5). Finally, all the $\mathbf{W}$ and $\mathbf{U}$ terms are learned weight matrices.

3.2 System Architecture

As mentioned earlier, the SDN decouples the control plane and data plane from network devices. The data plane, termed as switches, is just simple packet forwarding

elements. The control plane can be either a single computer or a group of logically centralized distributed computers. The two entities communicate in order to exchange network information and manage the whole network. In principle, the SDN network works in the following manner: packets from the internet traverse an SDN switch under the control of the SDN controller. Switches have flow tables for flow rules that contain match-fields, counters and actions that control traffic flows in the network. If the arriving flow does not match any rule in the SDN switch, the switch will send a *packet-in* message with the arriving flow header's information to the controller. The controller sends back a *packet-out* or *flow-mod* message to the switches to instruct them how to process the corresponding flow. Applications that run inside a controller can program the data plane for different purposes such as firewall, load balancer, IDS, etc. This paper focuses on the use of the SDN paradigm as network infrastructure for intrusion detection.

The IDS is implemented as an application on the SDN controller. The SDN-based IDS architecture is described in Fig. 2. It has three main components: Flow Collector, Anomaly Detector and Anomaly Mitigator.

- **Flow Collector**: This module is triggered when a *packet-in* message arrives. It will extract all the flow statistics such as protocol, source and destination IP and source and destination port. This module is also triggered by a timer function to send a *ofp_flow_stats_request* message to switches to request all the flow statistics information. Once the request is received, a *ofp_flow_stats_reply* message, which contains all the aggregated statistics of all flow entries, is sent back to the controller. All the features needed for anomaly detection will be created from these statistics and sent to the Anomaly Detector module.
- **Anomaly Detector**: We have chosen the GRU-DNN algorithm for the core of the Anomaly Detector module in this paper. The anomaly detector module loads a trained model, receives the network statistics and decides if a flow is an anomaly or not.
- **Anomaly Mitigator**: Through the Anomaly Detector's results, the Anomaly Mitigator module can make decisions on what to do with the flow. For example, it is possible to immediately stop the flow in order to prevent possible further attacks or to mirror the traffic to a deep packet inspector to further analyze the flow.

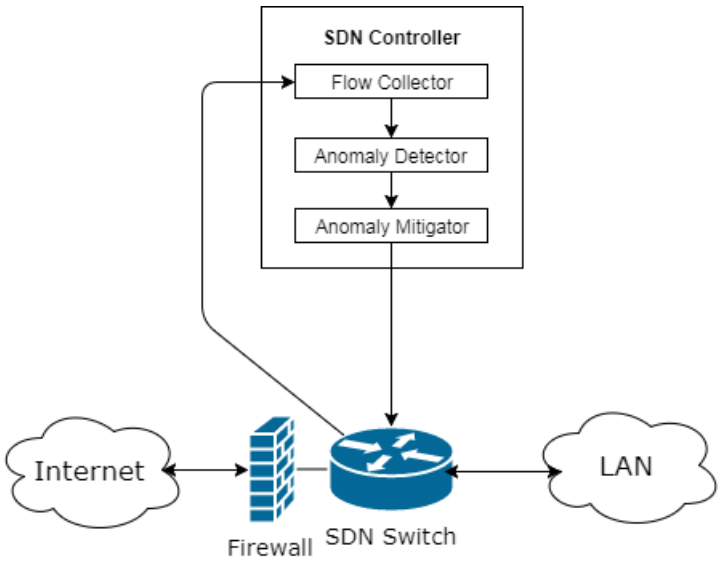

Fig. 2 SDN-based IDS Architecture

4 Performance Analysis

In this section, we firstly describe all datasets used in our experiment. Secondly, we explain all metrics used to evaluate our model performance. Thirdly, we describe all experimental setups. Detection results are given and compared with other works for a better overview in next section. Finally, the network performance of our GRU-RNN is evaluated and analyzed.

4.1 Datasets

Currently, there are only a few public datasets available for IDS evaluation (i.e., KDD Cup 99, NSL-KDD, DAPRA, and ISCX 2012) and none of them are specified for the SDN architecture. The KDD Cup 99 [11] and NSL-KDD datasets are some of the most popular datasets used in the literature to assess the NIDS performance. As mentioned in [15], the KDD Cup 99 has several inherent problems that makes the classifier fail to deliver a better accuracy. The NSL-KDD dataset [30] is introduced by Tavallaee et al. to solve these problems. However, the above datasets are out of date, lack traffic diversity and feature sets. Recently, the CICIDS2017 dataset [27] was published to deal with these issues. The SDN architecture is still under development, and so the NIDS dataset for the SDN is still quite rare and unpublished. Several researchers still use the conventional dataset to evaluate their approaches. In this paper, the NSL-KDD and CICIDS2017 datasets are chosen to evaluate our particular method.

4.1.1 NSL-KDD Dataset

The NSL-KDD contains 125,973 and 22,544 records in the training set and testing set respectively. Each traffic trace in this dataset has 41 features that are categorized into three types of features: basic, content-based and traffic-based features. Attacks in this dataset are categorized into four categories according to their characteristics. Our IDS is trained by the *KDDTrain+* dataset and tested by the *KDDTest+* dataset. In addition, the *KDDTest+* dataset contains 18 different types of attacks in addition to 22 attack types out of the *KDDTrain+* dataset. Details of each attack category are described in Table 2. Thus, the *KDDTest+* dataset is a reliable indicator to the performance of the model on zero-day attacks as well.

Table 2 Attacks in The NSL-KDD Dataset

Category	Training Set	Testing Set
DoS	back, land, neptune, pod, smurf, teardrop	back, land, neptune, pod, smurf, teardrop, **mailbomb, processtable, udpstorm, apache2, worm**
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, **xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named**
U2R	buffer-overflow, loadmodule, perl, rootkit	buffer-overflow, loadmodule, perl, rootkit, **sqlattack, xterm, ps**
Probe	ipsweep, nmap, portsweep, satan	ipsweep, nmap, portsweep, satan, **mscan, saint**

Within the context of SDN, the packet content is not directly accessible in the current OpenFlow protocol. The OpenFlow protocol does not allow us to get all the 41 features in the NSL-KDD dataset. This leads to a reduction of the features for the anomaly detection. So we just focus on the basic features and traffic-based features of the NSL-KDD dataset. Some of the features in these two categories can be easily retrieved from the SDN switches. In our research, a mixed feature set that contains six features from both the basic feature and traffic-based feature set are selected out of 41 features of the NSL-KDD dataset. The selected feature are *(duration, protocol_type, src_bytes, dst_bytes, srv_count, dst_host_same_src_port_rate).*These are SDN related features. Table 3 shows details of these features. These features are selected based on their SDN related nature without any feature selection or optimization algorithms.

The NSL-KDD dataset contains both the numerical and symbolic features, and so we will transform the symbolic features into numerical values. After converting, the dataset is normalized into the range of [0-1] by Min-Max scaling as follows:

$$x^{'} = \frac{x - \min(x)}{\max(x) - \min(x)}, x > 0, \tag{6}$$

where $x^{'}$ is the normalized value and x is the original value.

Table 3 The NSL-KDD's Feature Description

Feature Name	Description
duration	Length (number of seconds) of the connection
protocol_type	Type of the protocol (e.g. tcp, udp, etc.)
src_bytes	Number of data bytes from source to destination
dst_bytes	Number of data bytes from destination to source
srv_count	Number of connections to the same service as the current connection in the past two seconds
dst_host_same_src_port_rate	Number of connections that were to the same source port

4.1.2 CICIDS2017 Dataset

The CICIDS2017 dataset covers seven types of common attack families (i.e., Brute Force Attack, Heatbleed Attack, Botnet, DoS Attack, DDoS Attack, Web Attack, and Infiltration Attack). In this chapter, we choose a Wednesday dataset focusing on DoS, Heartblead, Slowloris, Slowhttptest, Hulk and GoldenEye Attacks. These types of attacks are on the rise and are a major threat to the SDN architecture.

In this dataset, we use 622,265 and 69,141 records for training and testing sets respectively. We extract a subset of nine features out of 80 features of this dataset for our research. These features have an SDN-related nature and can be extracted easily by SDN controllers. Details of these features can be seen in Table 4. This dataset is also normalized into the range of [0-1] by the Min-Max scaling as with the NSL-KDD dataset.

Table 4 The CICIDS2017's Feature Description

Feature Name	Description
Source Port	Source port of the flow
Destination Port	Destination port of the flow
Protocol	Protocol type of the flow
Flow Duration	Duration of the flow in microsecond
Fwd Packet Length Mean	Mean size of packets in forward direction
Flow Bytes/s	Number of flow bytes per second
Flow Packet/s	Number of flow packets per second
Flow IAT Mean	Mean inter-arrival time of packets
Fwd Packet/s	Number of forwarded packets per second

4.2 Evaluation Metrics

In order to evaluate our proposed approach, Precision (P), Recall (R), F1-measure (F1) and Accuracy (ACC) metrics are used. These metrics are calculated by using four different measures - True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN), defined as follows:

- TP: the number of anomaly records correctly classified.
- TN: the number of normal records correctly classified.
- FP: the number of normal records incorrectly classified.
- FN: the number of anomaly record incorrectly classified.

So the four metrics are:
Accuracy (AC): the percentage of true detection over total traffic records,

$$AC = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%. \tag{7}$$

Precision (P): the percentage of predicted anomalous instances predicted are actual anomalous instances,

$$P = \frac{TP}{TP+FP} \times 100\%. \tag{8}$$

Recall (R): the percentage of predicted anomalous instances versus all the anomalous instances presented,

$$R = \frac{TP}{TP+FN} \times 100\%. \tag{9}$$

F1-measure (F1): the harmonic of the precision and recall metrics to express the performance of the model,

$$F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} \times 100\%. \tag{10}$$

4.3 Experimental Setup

According to our experiments, A DNN with three hidden layers gives best results in all experiment cases. Therefore, we propose a GRU-RNN with three hidden layers. For the training phase, the batch size and epoch number are 100 and 1000 respectively. We use a Nadam optimizer [28] and Mean Squared Error (MSE) function for the model. In addition, we added L1-regularization to our model to prevent over fitting during the training phase. A DNN is also implemented with the same structure as the proposed GRU-RNN. We used Keras [6] to implement our GRU-RNN, DNN, and VanilaRNN models. Scikit-learn library [23] is used to implement the SVM algorithm and measure all the evaluation metrics.

4.4 Experiment Results

To start with, we present the anomaly detection performance of our proposed model in terms of Precision, Recall, F1-measure and accuracy on the NSL-KDD dataset. Details of the results given in Table 5 show that our GRU-RNN performs well for all the evaluation metrics. Both the legitimate and anomaly traffic traces are detected really well by the GRU-RNN. The detection rates of the legitimate and anomaly traffic traces is 89% and 90% respectively. The anomaly detection accuracy of 90% shows that the GRU-RNN is good at detecting zero-day attacks.

Table 5 Performance Metric Evaluation for the NSL-KDD Dataset

Class Name	Precision (%)	Recall (%)	F1-measure (%)
Legitimate	87	89	88
Anomaly	91	90	90

We also compare the performance of our proposed model with other popular algorithms like VanilaRNN, Support Vector Machine (SVM) and DNN using the same subset of six features. As we can see in Fig. 3, the GRU-RNN outperforms other algorithms in all the evaluation metrics. The GRU-RNN yields good results for both legitimate and anomaly traffic traces, while other algorithms just work well in only one class.

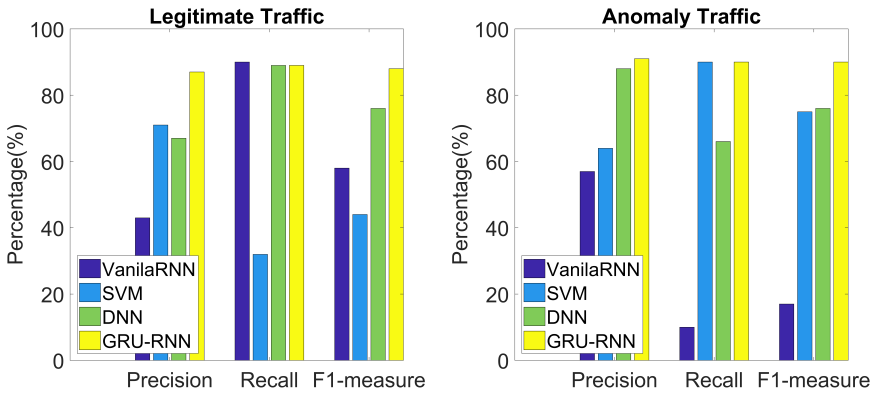

Fig. 3 Performance Metric Comparison

The results in Table 6 show that our approach outperforms other methods in terms of accuracy. The DNN, coming in second place, shows the potential of the DL approach in anomaly detection. The VanilaRNN gives the worst result compared with its counterpart GRU-RNN.

Table 6 Accuracy Comparison with Other Algorithms

Algorithm	**Accuracy**
VanilaRNN	44.39%
SVM	65.67%
DNN	75.9%
GRU-RNN (Proposed Model)	**89%**

The Receiver Operating Characteristic (ROC) curve is also presented to evaluate our proposed approach. The ROC curve is created by plotting the false positive rate versus the true positive rate. The area under the ROC curve (AUC) is used to determine which classifier predicts the classes best. The higher the AUC, then the better is the classifier. Fig. 4 shows that the proposed GRU-RNN achieves the highest AUC amongst all the tested algorithms with a True Positive Rate of 90% and a False Alarm Rate of 10%. The VanilaRNN gives the worst result as expected. As we can see, the GRU-RNN has a lowest False Positive Rate which is an important factor of the IDS.

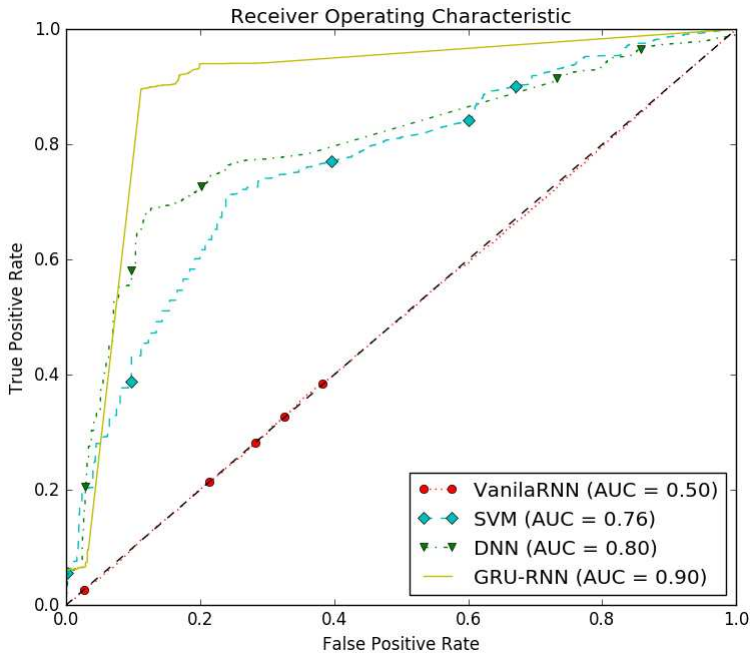

Fig. 4 ROC Curve Comparison for Different Algorithms

The Precision vs Recall curve shows the trade off between Precision and Recall for different thresholds. A high area under the curve represents both high recall and

high precision. An ideal system with a high area under the curve will return many results, with all results labelled correctly. As seen in Fig. 5, the GRU-RNN gives us the best results amongst all the algorithms. As the Recall threshold increases, the Precision decreases significantly for all the algorithms, except for GRU-RNN where increases Precision increases to 89%.

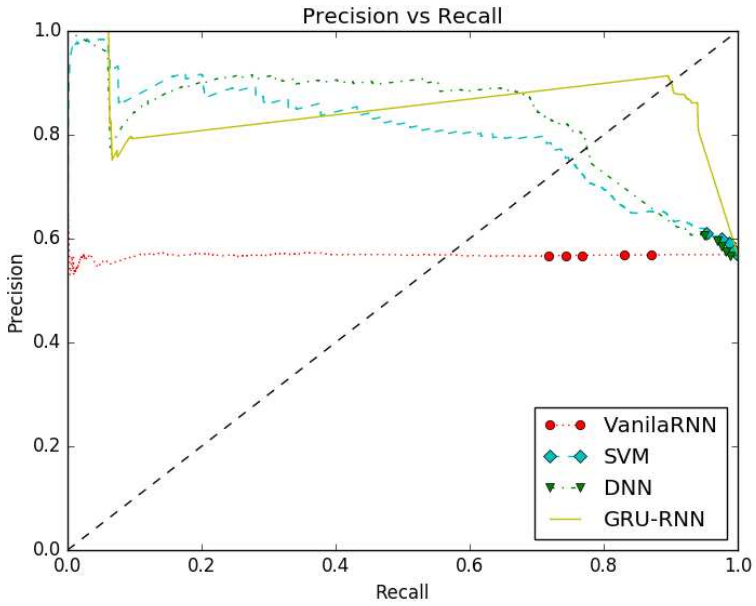

Fig. 5 Precision vs Recall Curves

Furthermore, we also compared the performance of our proposed model with others in the literature. Our GRU-RNN is compared with other state-of-the-art algorithms like SVM, DNN and NB Tree algorithms. The NB Tree gave the best result in [30]. The results in Table 7 show that our proposed model outperforms all the previous methods. Our GRU-RNN performs better than the SVM and NB Tree algorithms that use the whole set of 41 features for training and testing. The GRU-RNN result also indicates a significant improvement in accuracy compared to the basic DNN in our previous work.

Table 7 Accuracy Comparison with Previous Studies

Method	**Accuracy**
SVM [30]	69.52%
DNN [29]	75.75 %
NB Tree [30]	82.02%
GRU-RNN (Proposed Model)	**89%**

For further investigation, we evaluate the GRU-DNN performance as regards detecting DDoS attacks in the CICIDS2017 dataset. We compare the proposed GRU-DNN with DNN and ID3 algorithms. Results of ID3 algorithm are the best achieved from the CICIDS2017 dataset in [27]. Table 8 gives details of our evaluation. As can be seen, the proposed GRU-DNN has better results in all the evaluation metrics compared with the best result from [27]. The DNN yields slightly lower results than that of the ID3. The proposed GRU-DNN can work well with diverse and complex traffic traces and detect almost all types of DDoS attacks in the CICIDS2017 dataset.

Table 8 Performance Metric Evaluation for the CICIDS2017 dataset

Method	**Precision**	**Recall**	**F1-measure**
DNN	97%	97%	97%
ID3 [27]	98%	98%	98%
GRU-RNN (Proposed Model)	**99%**	**99%**	**99%**

From the above results, the GRU-DNN shows its strong potential in dealing with low-dimensional and raw traffic. Therefore, it is a potential solution for intrusion detection in the SDN paradigm.

4.5 Network Performance Evaluation

In this section, we evaluate the effect of our proposed GRU-RNN on the performance of the POX controller in the SDN environment.

4.5.1 Experiment Setup

The GRU-RNN is implemented as an application written in Python language in a POX [25] controller. Cbench [4] is a standard tool used for evaluating the SDN controller performance. Cbench runs in two modes: *throughput and latency modes*.

- *The Throughput Mode*: a stream of packet-in messages is sent to the controller for a specified period of time to compute the maximum number of packets handled by the controller.
- *The Latency Mode*: a packet-in message is sent to the controller and then waits for the reply to compute the time needed to process a single flow by the controller.

We ran our experiments on a virtual machine having an Intel Core i5-4460 3.2GHz with 3 cores available and 8GB of RAM. The operating system is Ubuntu 14.04 LTS-64bit. The controller performance is tested with a different number of

virtual OpenFlow switches emulated by Cbench. The performance of the POX controller running stand-alone is considered as a baseline for our evaluation. We also compare the proposed GRU-RNN algorithm with the DNN algorithm in our previous work [29].

4.5.2 Experiment Results

Throughput evaluation indicates the performance of the controller under heavy traffic conditions. Fig. 6 depicts the average response rate of the controller under three testing scenarios. As we can see, both the DNN and GRU-RNN cause overhead on the controller. The DNN algorithm is simpler than the GRU-RNN, and so it gives a slightly better network performance than that of the GRU-RNN. However, the GRU-RNN outperforms the DNN in terms of the detection accuracy. The affect of the GRU-RNN on the controller performance is predictable. The network throughput decreases slightly when the network size increases from 32 switches to 64 switches. The network performance degrades by about 3.5% when the network size is under 32 switches. When we increase the size to over 64 switches, the throughput drops by about 4%. The overhead on the controller of the GRU-RNN module is unavoidable. The GRU-RNN module has to send several *ofp_flow_stats_request* messages and process *ofp_flow_stats_reply* messages while processing packet-in messages. However, the throughput degradation is quite low and can be improved in the future.

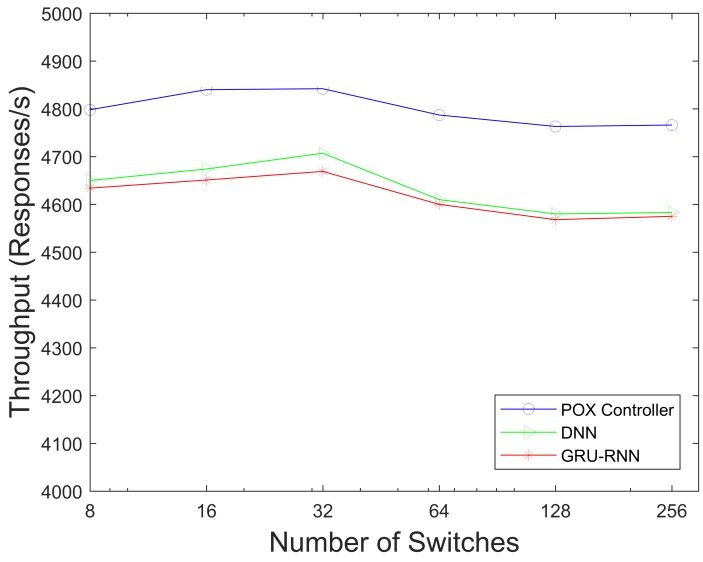

Fig. 6 Throughput Evaluation

Latency evaluation indicates the length of time that the controller takes to process one single packet. As we can see in Fig. 7, the network latency increases along with increasing the network size. When we increase the network size, the load on the controller is increased as well and this causes the overhead. The GRU-RNN still has the highest overhead amongst all. It takes time for the GRU-RNN to process the *ofp_flow_stats_reply* messages, *packet-in* messages and detect anomaly flows, so the overhead is unavoidable. The overall degradation is about 7% in all cases. This overhead is not significant and can be improved in the future.

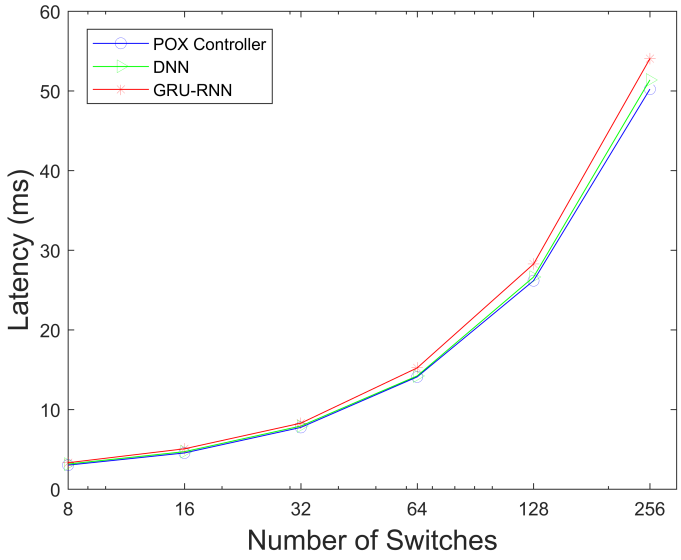

Fig. 7 Latency Evaluation

All in all, the overhead caused by the GRU-RNN on the SDN controller is quite low, and so our proposed approach has significant potential for real-time intrusion detection in the SDN paradigm. So there is a trade off between performance and security. However, the network performance still can be improved in the future.

5 Conclusion

This paper has made an attempt to briefly introduce the SDN architecture and its security issues. DL has great potential to be the key technology for intrusion detection in SDN. Despite the recent wave of success of DL in computer vision and language processing areas, there is a scarcity of DL applications in solving SDN security issues. In this chapter, we present an Anomaly-based IDS in the SDN en-

vironments using the GRU-RNN algorithm. We show that our proposed approach outperforms other state-of-the-art algorithms with an accuracy of 89% and 99% for the NSL-KDD and CICIDS2017 datasets. Our scheme uses a minimum number of features compared to other state-of-the-art approaches so computational costs can be reduced significantly. In addition, the network performance evaluation showed that our proposed approach does not significantly affect the controller performance. This makes our model a strong candidate for real-time detection. In the future, we will optimize our model and use other features to increase the accuracy and reduce the overhead on the controller. We will also try to extend our research to unsupervised intrusion detection approaches.

References

[1] AlEroud A, Alsmadi I (2017) Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach. Journal of Network and Computer Applications 80:152–164

[2] Bang Jh, Cho YJ, Kang K (2017) Anomaly detection of network-initiated lte signaling traffic in wireless sensor and actuator networks based on a hidden semi-markov model. Computers & Security 65:108–120

[3] Braga R, Mota E, Passito A (2010) Lightweight ddos flooding attack detection using nox/openflow. In: Local Computer Networks (LCN), 2010 IEEE 35th Conference on, IEEE, pp 408–415

[4] Cbench (n.d) https://github.com/mininet/oflops/tree/master/cbench. Accessed 04 Jul 2018

[5] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078

[6] Chollet F (2015) Keras. https://github.com/fchollet/keras. Accessed 04 Jul 2018

[7] Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555

[8] Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

[9] Hochreiter S, Bengio Y, Frasconi P, Schmidhuber J, et al (2001) Gradient flow in recurrent nets: the difficulty of learning long-term dependencies

[10] Ikram ST, Cherukuri AK (2016) Improving accuracy of intrusion detection model using pca and optimized svm. Journal of computing and information technology 24(2):133–148

[11] KDDCup99 (1999) http://kdd.ics.uci.edu/databases/kddcup99/. Accessed 04 Jul 2018

[12] Kokila R, Selvi ST, Govindarajan K (2014) Ddos detection and analysis in sdn-based environment using support vector machine classifier. In: Advanced

Computing (ICoAC), 2014 Sixth International Conference on, IEEE, pp 205–210

[13] Kreutz D, Ramos F, Verissimo P (2013) Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking, ACM, pp 55–60

[14] Kreutz D, Ramos FM, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S (2015) Software-defined networking: A comprehensive survey. Proceedings of the IEEE 103(1):14–76

[15] McHugh J (2000) Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. ACM Trans Inf Syst Secur 3(4):262–294, DOI 10.1145/382912.382923, URL http://doi.acm.org/10.1145/382912.382923

[16] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J (2008) Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38(2):69–74

[17] Mehdi SA, Khalid J, Khayam SA (2011) Revisiting traffic anomaly detection using software defined networking. In: International Workshop on Recent Advances in Intrusion Detection, Springer, pp 161–180

[18] Mousavi SM, St-Hilaire M (2015) Early detection of ddos attacks against sdn controllers. In: Computing, Networking and Communications (ICNC), 2015 International Conference on, IEEE, pp 77–81

[19] Nie L, Jiang D, Lv Z (2017) Modeling network traffic for traffic matrix estimation and anomaly detection based on bayesian network in cloud computing networks. Annals of Telecommunications 72(5-6):297–305

[20] Niyaz Q, Sun W, Javaid AY (2016) A deep learning based ddos detection system in software-defined networking (sdn). arXiv preprint arXiv:161107400

[21] ONF (n.d) Software-defined networking (sdn) definition. https://www.opennetworking.org/sdn-definition/. Accessed 12 Feb 2018

[22] Parwez MS, Rawat D, Garuba M (2017) Big data analytics for user activity analysis and user anomaly detection in mobile wireless network. IEEE Transactions on Industrial Informatics

[23] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830

[24] Phan TV, Van Toan T, Van Tuyen D, Huong TT, Thanh NH (2016) Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks. In: Communications and Electronics (ICCE), 2016 IEEE Sixth International Conference on, IEEE, pp 13–18

[25] POX (2009) https://github.com/noxrepo/pox. Accessed 04 Jul 2018

[26] Scott-Hayward S, Natarajan S, Sezer S (2016) A survey of security in software defined networks. IEEE Communications Surveys & Tutorials 18(1):623–654

[27] Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceed-

ings of fourth international conference on information systems security and privacy, ICISSP

[28] Sommer R, Paxson V (2010) Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE symposium on security and privacy, IEEE, pp 305–316

[29] Tang T, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M (2016.) Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (WINCOM'16), Fez, Morocco

[30] Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) 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

[31] Winter P, Hermann E, Zeilinger M (2011) Inductive intrusion detection in flow-based network data using one-class support vector machines. In: New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on, IEEE, pp 1–5