Native SDN Intrusion Detection using Machine Learning

Mohd Mat Isa, Lotfi Mhamdi
School of Electronic and Electrical Engineering
University of Leeds
West Yorkshire, United Kingdom
elmsbm@leeds.ac.uk, L.Mhamdi@leeds.ac.uk

***Abstract*—Security has been and still is a major challenge for communication networks. Recent advances in networking, notably the emerging of Software Defined Networks (SDN) has brought about major potential in providing highly secure communication networks. SDN decouples the data and control planes, while maintaining a centralised and global view of the whole network. This has resulted in feasible proactive, robust and secure networks. In particular, coupling SDN capabilities with intelligent traffic analysis using Machine Learning and/or Deep Learning has recently attracted major research efforts. However, most efforts have been just a simple mapping of earlier solutions into the SDN environment. This paper addresses the problem of SDN security based on deep learning in a purely native SDN environment, where a Deep Learning intrusion detection module is tailored to the SDN environment with the least overhead. In particular, we propose a hybrid unsupervised machine learning approach based on auto-encoding for intrusion detection in SDNs. The experimental results show that the proposed module can achieve high accuracy with a minimum number of selected flow features. The performance of the controller with the deployed model is tested for throughput and latency. The results shows a minimum overhead on the SDN controller performance, while yielding a very high detection accuracy.**

***Index Terms*—software defined networking, SDN, intrusion detection, deep learning, auto encoder, network security**

I. Introduction

In recent years, SDN has been widely studied and put into practice to assist in network management especially with regards to newly evolved network security challenges. SDN decouples the data and control planes, while maintaining a centralised and global view of the whole network. However, the separation of control and data planes makes SDN vulnerable to security threats. Seven main categories directly related to these risks have been identified which are unauthorized access, data leakage, data modification, compromised application, denial of services (DoS), configuration issue and system-level SDN security. One example of the most concerning issue is related to Distributed denial-of-service (DDoS). DDoS attacks have been a real threat for network, digital and cyber infrastructure. The magnitude of disruptions are massive and can bring down ICT infrastructure services. Various reasons for attack can be synopsized with the main objective of DDos attack as to paralyze network services by bombarding illegitimate traffic to targeted servers, network links or network devices [1]. Another example is that attackers can oversee and falsify network management information, implement saturation attack, become a perpetrator in man-in-the-middle attacks and so on. Therefore, it is important to adopt a defense mechanism for securing SDN architecture by analyzing vulnerability and improve trust management in traffic handling.

One of the proposed solutions for detecting such intrusion is the usage of the intrusion detection system (IDS). There are 2 types of IDS which are signature-based IDS and anomaly detection based IDS. The first type of IDS depends on the pre-identified signature of an intrusion that has been known before whilst the second type of IDS observed the traffic that deviates so much from normal and deemed suspicious or abnormal. Efficient approach in providing a high level of security in areas such as access control, authentication, malicious outbreak detection and others has resulted by adapting machine learning features and capabilities [2]. Network traffic classification either normal or attack are done via the usage of the Naive Bayes algorithm, Support Vector Machine, K-nearest Neighbor, Neural Network, Recurrent Neural Network, Deep Neural Network and others [3] [4]. Reinforcement learning practicality also being embedded in the case of unlabelled data and the needs of the different approaches for intrusion identification and detection. With the abovementioned advancement and additional capabilities provided by SDN architecture, research of incorporating machine learning has been done to improve network security.

Researchers also have embarked autoencoder approach as an alternative in network intrusion detection environments such as the works of [5] and [6]. Autoencoders are unsupervised learning techniques that make use of the neural network for undertaking representation learning. A bottleneck in the network is imposed to force a compressed knowledge from the original input. An ideal autoencoder model can be achieved by balancing the sensitivity of the input for accurate reconstruction build up and take into account overfitting condition which simply memorizing the input data. For most cases, a loss function usually being constructed with the term on encouraging for input sensitivity and secondly by discouraging overfitting or memorization of input data, for example by adding regularizer.

Embracement of machine learning and autoencoder such as

978-1-7281-5320-9/20/$31.00 ©2020 IEEE

the works above has outset promising solution in intrusion detection. Progression in the accuracy level, reducing training and execution time needed can still be improved especially to be embedded in a fast speed of SDN environment. This further improvement motivates in outlining this research. In summary, the contributions of this paper are the following:

- We introduce a hybrid combination of an autoencoder (AE) and random forest (RF) algorithm in the SDN environment.
- Our AERF approach yields a detection rate of 98.4% using a minimum number of features.
- We also evaluate the network performance of the proposed approach in the SDN. The test results show that our approach is a significant potential for real-time detection with minimum impact on the controller and noteworthy processing time.

The rest of this paper is organized as follows. Section II briefly discusses the related work. In Section III, we present the methodology used. The network performance analysis is described in Section IV. Finally, Section V concludes the paper and presents future work.

II. Related Work

Author of [4] proposed a deep learning approach for flow-based anomaly detection in the SDN simulation environment. A Deep Neural Network was introduced with 3 hidden layers to perform binary classification with six basic feature selection from the NSL-KDD dataset. The accuracy reported was 75.75%. Research in [5] proposed a non-symmetric deep auto-encoder (NDAE) model for intrusion detection and achieving a 5-class accuracy of 85.42% with NSL-KDD dataset while [7] adopted big data visualization and statistical analysis method with the combination of autoencoder for potential threat detection. The accuracy reported from the approach achieved up to 87% accuracy.

Increased accuracy of detection in a similar approach of deep learning has shown a good achievement and yet there is still potential for improvement. High consuming training times, multi-layered complex processing, inconsistent performance are some examples of deficiencies that can be further improved. Experimenting with model optimization which related to the process of training, calculating loss and controlling activation of functions are still needed and can be improved especially in intrusion detection mechanisms. Therefore, the work presented in this paper is focusing on further analyzing the optimization process and we feel that it can contribute to the current cumulative knowledge of intrusion detection in the SDN environment.

III. Methodology

A. SDN-based Intrusion Mitigation Architecture

The proposed architecture is as shown in Fig. 1. Two main functionalities are Assembly Module for statistics collection and Adaptive ML Module which will be adopting deep learning for analyzing traffic received for intrusion detection and enforcing policies enforcement with capabilities of machine learning in the SDN environment. The proposed solution will

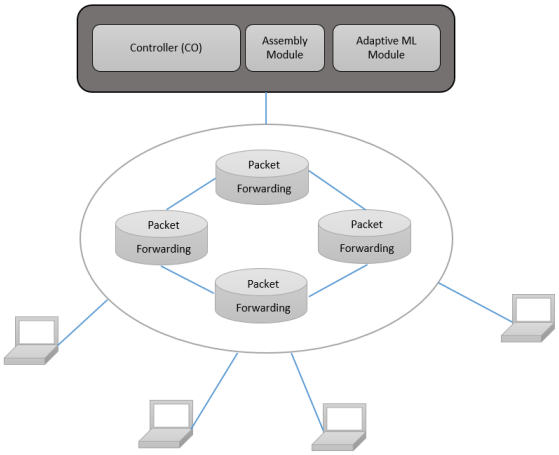

Fig. 1. SDN-based Intrusion Mitigation Architecture

focus on mitigating the attack with enrichment of machine learning benefits. The functionality of the main components is presented as follows:

a) Assembly Module: Collecting traffic information of the SDN network. Collected data will be assembled as needed input for the controller to take proper actions. Utilized standard Openflow protocol which periodically sends an Openflow flow stats request message to all Openflow connected switches. Managed switches will respond back to the message that contains traffic flow statistics.

b) Adaptive ML Module: Proposed intrusion detection using AERF will be deployed and attack traffic will be penalized when detected by the controller using machine learning functionality in the adaptive ML module.

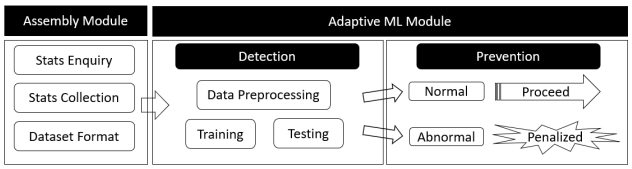

Fig. 2. Intrusion Mitigation Module

The functionality of the proposed module are as shown in Fig. 2. Assembly Module will handle the statistics inquiry and collection from all OpenFlow enable switches. The gathered data will then be processed with the required format for the controller to proceed with detection and prevention in the adaptive ML module. The training and testing process of the detection process needs to be fast and yield high accuracy as it will deal with the high speed data flow. Upon the classifying process, normal traffic will proceed as usual whilst abnormal traffic will be penalized.

2

B. Dataset

Most of the approaches in machine learning required a training dataset to train the algorithm before being tested. Usage of the dataset in classifying attack has been done in [5] and a total of 63% of the reported research used the KDD-99 dataset for training and testing. In this work, a newer version of KDD-99 which is an NSL-KDD dataset will be used. The dataset has been corrected from enormous duplication of data and defects that cause bias in evaluation. NSL-KDD dataset has 41 features and three of these features are symbolic while others are numeric. Our model is trained by the KDDTrain+ dataset and tested by the KDDTest+ dataset. In addition, the KDDTest+ dataset contains 17 different types of attacks compared to 22 attack types of the KDDTrain+ dataset. Thus, the KDDTest+ dataset is a reliable indicator of the performance of the model on zero-day attacks as well. Distribution of both dataset according to network traffic classes are as shown in TABLE I below.

KDDTrain+ dataset contains 125,973 records and KDDTest+ dataset contains 22,544 records. The data processing task will be deployed to prepare the dataset for training and testing purposes. Simplified autoencoder with the fine-tune of overfitting factors is applied to the NSL-KDD training and testing dataset and only selected 6 features from Section III. The selected features are service, flag, src_bytes, dst_bytes, logged_in and serror_rate.

TABLE I
DISTRIBUTION OF KDDTRAIN+ AND KDDTEST+ DATASET ACCORDING TO TRAFFIC CLASSES

Dataset	Normal	Dos	Probe	R2L	U2R	SUM
Train+	67,343	45,927	11,656	995	52	125,973
Test+	9,711	7,458	2,754	2,421	200	22,544

Many of the features of the NSL-KDD dataset have very large ranges between the maximum and minimum values, such as the difference between the maximum and minimum values in "duration [0, 58329]." In this example, the maximum value is 58,329 and the minimum is 0. A large difference also exists in other feature values, such as "src-bytes" and "dst-bytes" thereby making the feature values incomparable and unsuitable for processing. Hence, these continuous features are normalized by using max-min normalization for mapping all feature values to the range [-1, 1] according to,

$$x_i = \frac{x_i - Min}{Max - Min} \tag{1}$$

Features that are discrete such as service and flag are being processed using a one-hot encoding. As the speed of processing is a concern, using a one-hot implementation typically allows a state machine to run at a faster clock rate than any other encoding of that state machine. Both of these values will then be used as input to the model for training and testing.

C. AERF Model

Deep learning approaches have good potential to achieve effective data representation for building improved solution. For the approach to be adopted in the real environment, it has to be composed with a fast simplified approach for performance. Therefore, in this proposal, an autoencoder with fine-tuning the factors that effect overfitting is being put forth. A hybrid combination using the random forest algorithm for further classification from the preliminary autoencoder calculation process is proposed. A preliminary test with the usage of the NSL-KDD dataset will be used in order to test the approach by targeting high accuracy detection but with the main objectives of simplifying the detection within the minimum time processing needed.

The AE has two phases which are encoding and decoding. For the encoding process, input data x is compressed into a low-dimensional representation h and then the decoder reconstructs the input based on the low-dimensional representation.

$$h = f(Wx + b) \tag{2}$$

$$y = f(W'h + b') \tag{3}$$

where f(·) is a non-linearity activation function, W and W' are hidden weight matrices, b and b' are biases and y is output vector. Minimizing the differences between the input x and the output y is the main goal during the training process. The average squared difference between the estimated values and the actual values is used as follows:

$$L(x, y) = ||x - y||_2^2 \tag{4}$$

During the training process, only normal traffic is used from the training dataset. Attack datasets not used because of unbalanced and certain attack types are not well represented. In this approach, autoencoder with dropout on the inputs is proposed which consists of an input layer of 85 neurons due to the number of features for each sample is 85. The input layer of 85 neurons after the selection of 6 features is used for the training and testing process. 85 neurons are consists of service (70 input), flag (11 input), src_bytes (1 input), dst_bytes (1 input), logged_in (1 input) and serror_rate (1 input).

A total of 125,973 with 85 input and 22,544 with 85 input for dataset training and testing respectively. After a dropout layer, a hidden layer of 8 neuron units is proposed with the hidden representation of the autoencoder has a compression ratio of 85/8 forcing it to learn interesting patterns and relations between the features. An output layer of 85 units with the activation of both the hidden layer and the output layer using the most popular rectified linear unit (ReLu) activation function as shown in Fig. 3 below. It gives an output x if x is positive and 0 otherwise.

$$A(x) = max(0, x) \tag{5}$$

During the reconstruction of the inputs, the autoencoder was trained using only the samples labeled "Normal" in the

3

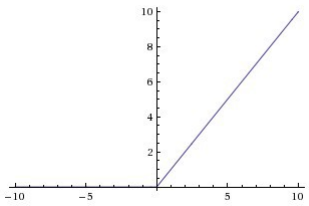

Fig. 3. ReLu activation function

training dataset. This is for the purpose of allowing it to capture the nature of normal traffic and to minimize the mean squared error between the input and output. Preventing the autoencoder from simply copying the input to the output and overfitting the data, regularization constraints are enforced. Model architecture details are as shown in Fig. 4 and the parameter used for the model depicted in TABLE II.

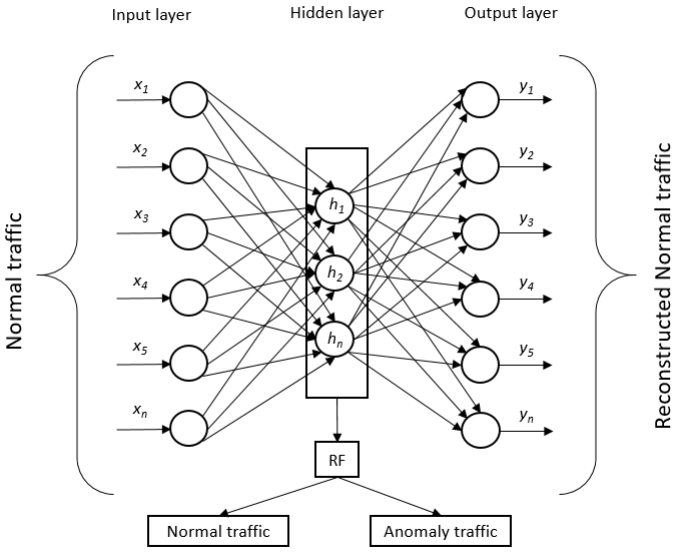

Fig. 4. AERF Model

TABLE II
AERF MODEL ARCHITECTURE

Input layer	Dropout	Hidden layer	Act regularizer	Output layer	Activation Func
85	0.5	8	10e-5	85	ReLu

The hidden layer of the model is being used as the input for the second layer classification using random forest algorithm. Random forest applies the principle of ensemble learning method on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. Examples of research works that employ RF for intrusion detection is such as [8] and [9]. Comparative studies with regards RF as one of the best algorithms selected for a similar area being concluded by the works of [10] and [11]. In our proposed model, the encoded representations learned by the autoencoder is being used as the input for the RF classifier to classify either normal or anomalous of the traffic.

IV. PERFORMANCE ANALYSIS

A. Evaluation Metrics

For evaluation purpose, Precision (P), Recall (R), F-measure (F) and Accuracy (ACC) metrics are used.

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

These metrics are calculated by using four different measures, true positive (TP), true negative (TN), false positive (FP) and false negative (FN):

- 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

B. Experimental Results

The model is trained for 10 epochs using an Adam optimizer with a batch size of 100 as shown in Fig. 5. A total of 60,608 samples are used tor training and 10% of the total normal traffic which are 6735 samples as validation. Tensorflow backend is used to run and simulate the proposed model.

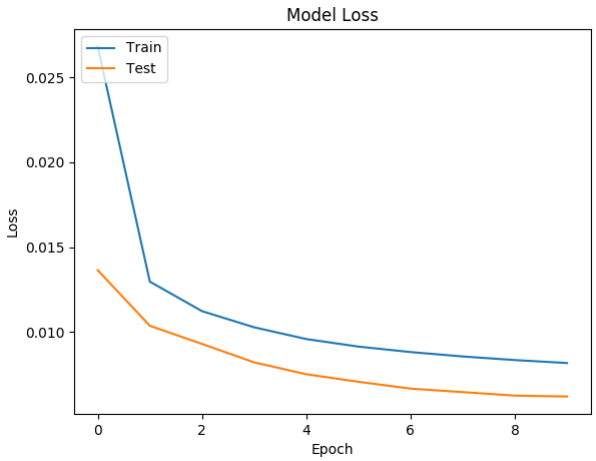

Fig. 5. Model training process

Repetition of training and test has been performed with stabilized outcome. Results that have been collected from the test are promising with the mean accuracy of 90.19% achieved as shown in TABLE III from the autoencoder process alone.

4

TABLE III
AUTOENCODER PERFORMANCE OVER TEST DATASET

Method	Accuracy	Precision	Recall	F1
Autoencoder	90.19	88.78	94.70	91.64

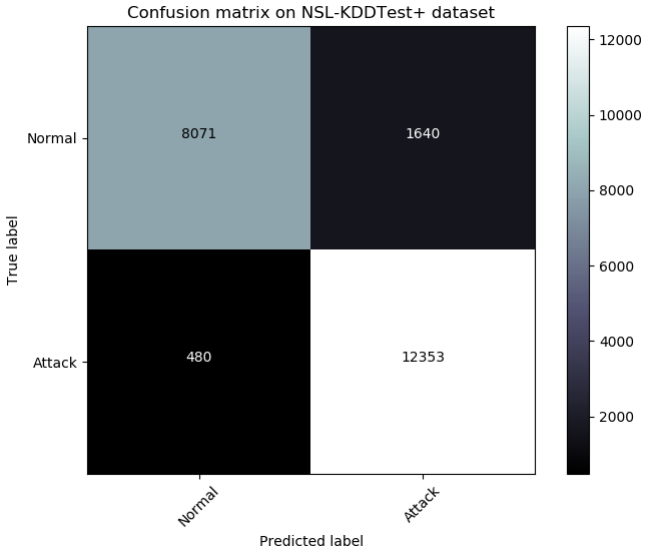

Fig. 6. Autoencoder confusion matrix

Example confusion matrix from the autoencoder process for the test dataset is as shown in Fig. 6.

The threshold for autoencoder is based on the calculated loss from the training process for the separation of normal and anomaly traffic. Distribution of detection from the autoencoder is as shown in Fig. 7 below. It is noticed that the error rate for normal and abnormal traffic is highly distinctive from the calculated threshold value. An average amount of 10% of the calculated error is in the wrongly determine area. As for the next process, the encoded representation of the autoencoder is inputted into RF for a further breakdown of the category of the traffic.

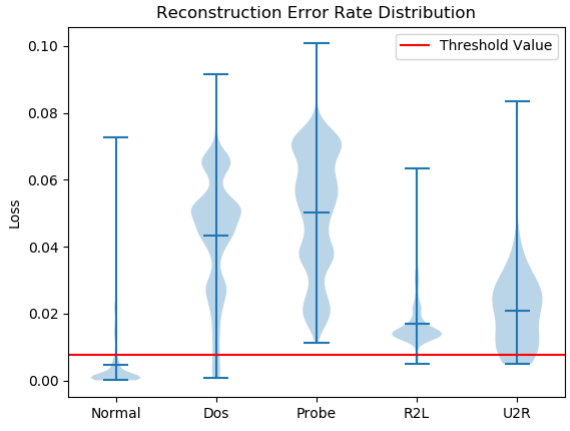

Fig. 7. Distribution of detection over test dataset

RF classification that has been adopted has yielded an accuracy of 98.4% for the input from the autoencoder. Comparison from previous related works that have also using the same NSL-KDD dataset and intrusion detection are tabulated in TABLE IV. An increase in accuracy is achieved with the proposed model and enables further improvement to be carried out.

TABLE IV
PERFORMANCE COMPARISONS IN BINARY CLASSIFICATION USING KDDTRAIN+ AND KDDTEST+ DATASET

Method	Accuracy Rate
Naive Bayesian [12]	75.56%
MLP [12]	77.41%
Random Forest [12]	80.67%
J48 [12]	81.07%
RNN [13]	83.28%
Fuzzy approach [14]	84.12%
AE + Gaussian NB [15]	84.34%
STL-IDS [12]	84.96%
S-NDAE [5]	85.42%
SAE+SMR [6]	88.39%
GRU-LSTM [4]	89.00%
AERF	**98.40%**

V. SDN CONTROLLER PERFORMANCE

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

A. *Experimental Setup*

The AERF is implemented as an application written in Python language in a Ryu controller. Cbench is a standard tool used for evaluating the SDN controller performance which supports two running modes, throughput and latency. The throughput mode computes the maximum number of packets handled by the controller and latency mode computes the time needed to process a single flow by the controller. We run our experiments on a virtual machine having an Intel(R) Xeon(R) E3-1226 3.3GHz with 3 cores available and 8GB of RAM. The operating system is Ubuntu 18.04.2 LTS 64bit. The controller performance is tested with a different number of virtual OpenFlow switches emulated by Cbench. The performance of the stand-alone Ryu controller is considered as a baseline for our evaluation.

B. *Analysis of Results*

The average response rate of the controller for both the baseline and AERF evaluation is as depicts in Fig. 8. The throughput achieved a slight decrease when the number of switches increases. Although the throughput decreased, comparison with the baseline stand-alone Ryu controller is not so much different and still acceptable.

An interesting observation is the Ryu controller which has a negligible impact on its latency performance as shown in Fig. 9. When we increase the number of connected switches, the latency achieved are between 3 to 4 ms and not so much

5

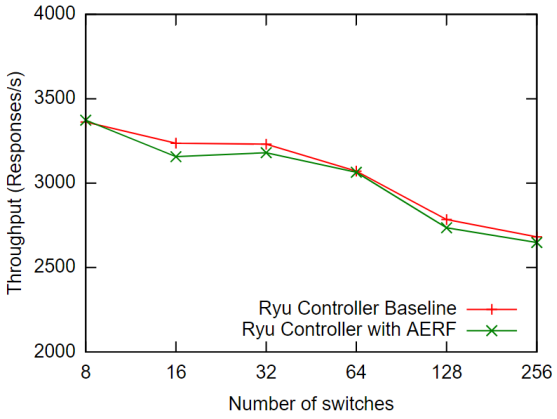

Fig. 8. SDN environment throughput evaluation

overhead introduced. This is a similar commentary that has been done by [16] regarding the latency of the Ryu controller performance achieved. A comparison of time taken for training and testing NSL-KDD dataset are as shown in TABLE V below.

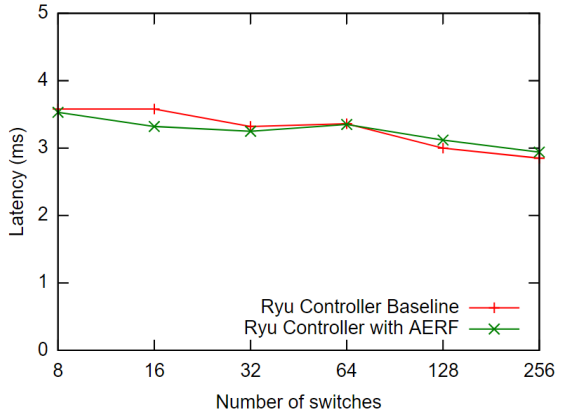

Fig. 9. SDN environment latency evaluation

TABLE V
PERFORMANCE COMPARISONS IN BINARY CLASSIFICATION USING KDDTRAIN+ AND KDDTEST+ DATASET

Method	Training Time	Testing Time
STL-IDS [12]	673.031	4.468
S-NDAE [5]	644.84	-
AERF	**18.32**	**2.63**

Simplicity is the focus of this approach. The simplified single hidden layer of minimum neurons has made the training and testing process very fast which are suitable for the SDN environment which demands high processing capability and real-time implementation. Adaptive calculation of threshold within the model has bypassed the needs of human intervention to manually set it. The value of the threshold could also be manually adjusted in order to suit the needs between sensitivity and specificity accordingly. The accuracy achieved with the very minimum processing time is a good sign of future applicability in real SDN environment deployment.

VI. CONCLUSION AND FUTURE WORK

In this paper, we present a hybrid combination of an autoencoder and random forest algorithm for intrusion detection in a native SDN environment. We show that the proposed AERF surpasses other previous works with an accuracy of 98.4% in addition to a decreased amount of time needed for training and execution. From the results accomplished, it proves that the model is optimistically efficient for real-time intrusion detection. In addition, the throughput and latency experimental showed that our proposed approach does not significantly affect the SDN controller performance. Therefore, it is practical for implementation and will be adapted in the SDN-based intrusion mitigation architecture for further prevention process.

REFERENCES

[1] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, 2017.

[2] S. Das, Y. Liu, W. Zhang, and M. Chandramohan, "Semantics-based online malware detection: Towards efficient real-time protection against malware," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 2, pp. 289–302, 2016.

[3] L. Barki, A. Shidling, N. Meti, D. G. Narayan, and M. M. Mulla, "Detection of distributed denial of service attacks in software defined networks," *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016*, pp. 2576–2581, 2016.

[4] 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," *2018 4th IEEE Conference on Network Softwarization and Workshops, NetSoft 2018*, no. NetSoft, pp. 462–469, 2018.

[5] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[6] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, "A deep learning approach for network intrusion detection system," *EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, 2015.

[7] C. Ieracitano, A. Adeel, M. Gogate, K. Dashtipour, F. C. Morabito, H. Larijani, A. Raza, and A. Hussain, "Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10989 LNAI, pp. 759–769, 2018.

[8] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," *Proceedings - 2017 IEEE International Conference on Computational Science and Engineering and IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, CSE and EUC 2017*, vol. 1, pp. 635–638, 2017.

[9] Y. Y. Aung and M. M. Min, "An analysis of random forest algorithm based network intrusion detection system," *Proceedings - 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2017*, pp. 127–132, 2017.

[10] M. Anbar, R. Abdullah, I. H. Hasbullah, Y. W. Chong, and O. E. Elejla, "Comparative performance analysis of classification algorithms for intrusion detection system," *2016 14th Annual Conference on Privacy, Security and Trust, PST 2016*, pp. 282–288, 2016.

[11] S. Choudhury and A. Bhowal, "Comparative analysis of machine learning algorithms along with classifiers for network intrusion detection," *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, ICSTM 2015 - Proceedings*, no. May, pp. 89–95, 2015.

[12] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

6

[13] C. Yin, Y. Zhu, J. Fei, and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.
[14] R. A. R. Ashfaq, X. Z. Wang, J. Z. Huang, H. Abbas, and Y. L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2016.04.019
[15] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 3854–3861, 2017.
[16] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN Controllers: Benchmarking & Performance Evaluation," pp. 1–14, 2019. [Online]. Available: http://arxiv.org/abs/1902.04491

7