Performance Comparison of DoS Detection on SDN Network Using Various Machine-Learning Techniques

***Abstract*—Software-Defined Network (SDN) is a technology that has advantages in networking, such as automation, flexibility, and resource utilization. One of the SDN implementations can use P4 Language-based, where the SDN user can make specific data plane they need with programmable advantages of P4, the Software-Defined Network itself is famous for its centralized architecture by separating Control Plane and Data Plane. But there are critical aspects of the SDN architecture, one of which is that it is vulnerable to DoS attacks that can cause the network to lose the availability principle of the CIA Triangle. A Denial of Service (DoS) attack is a criminal activity carried out to exploit the computer network, this activity has the goal of making the network inaccessible this can happen by using the flooding packet method on the computer network. Therefore we can prevent this attack by detecting the attack using an early detection system to solve this issue proposed to build an Intrusion Detection System (IDS). So in this paper we proposed anomaly-detection system based on a machine learning algorithm. The proposed system will use the deep learning method by using Long Short-Term Memory (LSTM) algorithm combined with the Naive Bayes algorithm (NB) to become LSTM-NB. The proposed system achieved an accuracy 88% on SDN-DL Dataset, 98% on NSL-KDD, and 96% on CICIDS2017 with FNR score between 1-2%, beside that we compared our proposed system with another machine-learning and deep-learning methods, beside comparing with dataset we evaluated proposed system with simulation dataset we generated with P4 Based SDN network. Through extensive experimental evaluation, we conclude that our proposed approach exhibits a strong potential for DoS detection in the SDN environments.**

***Index Terms*—Computer Network Security, Intrusion Detection System (IDS), Machine Learning, Deep Learning, Denial of Service (DoS).**

I. Introduction

Computer Networking is a complex matter and difficult to manage. This is due to the large number of equipment used on the [1] network. Besides that, the devices on traditional networks have designs, software, and hardware that is related to one vendor. Each other vendor has different designs and devices [2], then there is a technology that changes in the context of network design and management, namely Software Defined Network or well known as SDN [1]. SDN has different characteristics when compared to traditional networks, the difference is the separation of the control plane, and data plane of a network device [1], [2] SDN applies the concept of centralization to its network architecture like a traditional network. SDN architecture network is very vulnerable to cyber-attacks [3]. There are three types of attacks targeting SDN networks. These attacks are fraud attacks, intrusion attacks, and malicious tampering attacks [4]. One of the attacks that occurred on the SDN network was Denial of Services, although, with the centralization applied to SDN, this vulnerability itself was caused by the architecture of SDN [5].

Denial of Services, commonly known as DoS, is a cyber-crime with the method of sending packets excessively and aiming to exploit the resources of the network [6]. The DoS attack itself is a threat to network security. DoS in SDN is caused by the separation of the control plane and data plane, causing a vulnerability in the SDN architecture [5]. Attackers can exploit both the control plane and data plane [7], so it will disrupt the flow rule decision and can also result in the occurrence of a bottleneck on the network. It can be harmful if there is a failure on the network component [8], there are two types of DoS attacks, namely volumetric attacks such as ICMP-Flood, UDP-Flood, and TCP-SYN Flood, and application-layer attacks [8]–[10], to prevent DoS attacks on the network. Early detection measures can carry out using an Intrusion Detection System, so this paper will explore the performance comparison between machine-learning techniques for detecting DoS attacks. We proposed LSTM-NB methods to solve this problem and comparing with other machine-learning techniques.

A. *Related Work*

An intrusion Detection System (IDS) can use to prevent DoS attacks. IDS has a task to inspect every activity that occurs on the network [6], [11], [12]. Basically for detection using IDS, two types of approaches used, namely **signature-based** and **anomaly-based**. Both approaches have drawbacks, such as low intelligence and weak adaptability if applied traditionally. So it is ineffective when implemented in many scenarios [13].

From that, we need a dynamic approach that can solve this. During the last decade, there have been many surveys and reviews of the technology used in IDS, one of which is technology by applying the machine learning method. This method can be applied to Intrusion Detection System [3], [13]. Machine-learning methods commonly used are SVM, Random Forest, KNN, and technologies such as Artificial Neural Network [13].

Machine learning can be improved; machine learning generally has two variations. On IDS implementation, most of them are currently still using variations of Shallow Learning; shallow learning needs continuous learning on the model to update its capabilities. Besides, it needs more in-depth analysis to select the features used [5]. The next drawback is that the system can only detect some DoS attacks [3], [13]. Deep Learning methods can be used to solve this problem [3], [5]; Deep Learning was chosen to solve the problem because of its learning ability, and generalization of the existing attributes [5].

B. Paper contribution and organization

This paper proposed to use the Deep-Learning, namely Recurrent Neural Network (RNN) using the Long Short-Term Memory (LSTM) algorithm and then combined with the Naive Bayes algorithm (NB) to build a machine learning-based system.

This system was inspired by research conducted by Ahuja et al. at their research; combined two different deep learning algorithms (CNN-LSTM). Using two different deep learning algorithms means we need more computation resources. Besides that, if using deep learning, we can skip the feature selection stage, but in some cases, we need a flexible and configurable system, so we need to know some features knowledge. Because of that, we propose to use LSTM-NB. We implemented deep learning and shallow learning; we can improve shallow learning performance by using deep learning but using fewer computation resources.

The RNN-based algorithm was chosen because of its advantages in handling time-series data; besides that, Musumeci et al., in their research, advised implementing the RNN algorithm for DoS Detection Recurrent Neural Network (RNN) has shown great success in language modelling, text generating. Speech recognition based on Tang et al. research RNN is believed to be a powerful technique to represent the relationship between current and past events and enhance anomaly detection system [2]. However, RNN has some disadvantages. One of them is the Long Dependency Problem; in theory, RNN can solve that problem, but in practice, RNN cannot solve it. This problem is called the Vanishing gradient problem.

LSTM was chosen because of its advantages, and it can optimize the long dependency problem in the Recurrent Neural Network. Besides that, LSTM also can keep records of information on packets that have passed the system built [14] so that with this method, the packet analysis is expected to be more accurate.

Another algorithm used is Naive Bayes; it was chosen because it is pretty simple to implement and has high accuracy [14], so the proposed system will achieve better performance, so we can conclude our main contributions to the paper are as follows:

- We compare different ML algorithms to detect DoS attacks such as Naive Bayes, LSTM, ANN, and LSTM-NB. In terms of accuracy, recall, precision, and false-negative rates.
- We compare the performance of the machine-learning model on NSL-KDD, SDN-DL, and simulation-generated datasets, approach 88% accuracy and present a False Negative Rate to show model performance.
- We provide P4-based data plane code for simulation, implementing deep learning intrusion detection system, and simulation features extraction code.

The present paper is organized as follows. In Sec. II, we provide background on Software Defined Network, Denial of Services attacks and P4 language. In Sec. III, we overview the proposed system schema. In Sec. IV, we describe the experimental framework, such as evaluation methods, dataset, and data preprocessing. In Sec V, we provide the result of the experiment conducted. In Sec VI, we provide the conclusion and remarks.

II. Background

A. DoS Attack

Denial of Service well known as DoS are one of cyber-security attacks aim computer network and making computer network inaccessible [15], this type of attack aiming communication nodes such as network infrastructure or components, the methods use by flooding network with packet and make network low on resource (overwhelm), and make other user can't accessed the network for several time, currently there are several types of prevalent DoS attacks, namely UDP Flood, ICMP Flood, TCP Flood, HTTP Flood, HTTPS Flood [16] or we can grouping them based on their protocol like UDP, TCP, HTTP, ICMP attack, the attack can easily classified based on the methods use, DoS attacks have many varieties and methods use to make network done, but have some similarity usually they overwhelm network using packet [6], and from Kaspersky Lab data since the beginning of COVID-19 pandemic DoS attack rate increase up to 20%, since online activities increased.

We know the attack mainly consists of three types of attack SYN, UDP, and TCP attack, where SYN attacks are 78.20%, followed by UDP 15.17%, and followed by TCP attacks as much as 5.67%. From research conducted by Sangodoyin et al., the effects on an SDN network will affect the network performance. Such as throughput parameters and further jitter if the attack becomes intense. In this case, using the exhaustive method can cause a decrease in the capabilities of the SDN network. It occurs both in the control plane and in the data plane. One of the solutions to minimalize DoS attack impact is making an early detection system and mitigation. In this paper proposed solution is to build a machine-learning-based intrusion detection system. Therefore, the LSTM-NB method is proposed LSTM-NB is an application of an intelligent system including Machine-Learning and Deep Learning Based to perform anomaly detection, then significantly increase performance on Accuracy, Precision, and False Negative Rate on detection.

B. Software Defined Network

Software-Defined Network (SDN) is an innovation in networking that changes how we design and manage the network itself, using SDN management, control, and creating innovation that is easier and possible to implement [1]. In addition, traditional networks are closed, proprietary to the control part and have a different configuration from other vendor products, making it for network administrators hard to manage and configure [1], [2]. SDN changes all that because SDN provides a new paradigm option that implements a centralized system; it separates two parts of the network, namely the control plane that had tasked with making decisions to control the network and the data plane that is in charge of doing forwarding packet according to what the control plane command [1], [17], [18].

SDN builds on the Application, Control, and Infrastructure layers. The main difference between a traditional network and SDN is the separation of the control plane and data plane; this architecture allows SDN to implement the centralized architecture and then have the advantage of modifying the network from one region. Furthermore, Infrastructure and all these layers are connected using southbound and northbound API to communicate with each other; currently, the well-known SDN protocol is Openflow, which provides a simple and robust SDN system, but OpenFlow has disadvantages they lack programmability. For example, we cannot add a new header to our system; to solve this problem, we can use another solution using P4-language Dataplane.

C. P4 Language

P4 language is a high-level programming language for routers and switches, designed to allow programming on data plane components such as hardware or software switches, network interface cards, routers, etc [3], P4 is open-source language, not like OpenFlow P4 use Top-Down design, on Figure 1 described differences between OpenFlow and P4 design.

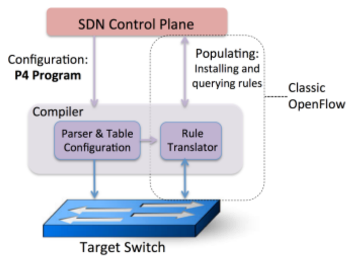

Fig. 1. P4 Top Down Design [19]

From Figure 1 main difference between P4 and OpenFlow is programmability. When developing a P4, users will make a P4-based program to satisfy user requirements. Then users need to compile the program before its usage either on the behavioural model or the switch. In this paper, Simulation will use the behavioural model with the Mininet simulator to simulate a DoS attack, based on Musumeci et al. a P4 program composed of the following component:

- *Parsers* had a function to identify the allowed protocols and fields in the program. Typically, they contain the names of the used headers and their size in bits.
- *Control Plane (Ingress/Egress)* had a function to describe the order of processing rules that will be applied to the packet.
- *Table* had a bunch of processing rules that form "match-action". When the P4 program processes packets, the ingress pipeline is executed to look for the matching rule(s) which fit the incoming packet.

In addition, P4 defines programmable packet metadata, associating extra information to the packet and stateful objects that may use to implement Finite State Machine and perform context-based processing [3].

III. Proposed Detection System Schema

A. System Overview

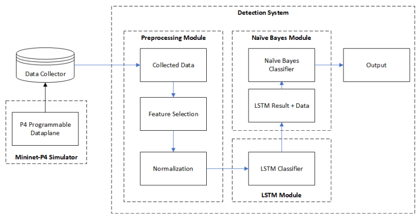

Fig. 2. Proposed System Architecture

The system will be built with two main components, as depicted in Figure 2. The Detection System will have three main modules *Preprocessing* this module will prepare data from the data collector matching with classification. Module input type, this module will contain feature selection module and normalization module on this paper, we using Min-Max Scaler module to Normalize data after that data will be passed to LSTM Module.

1) LSTM Module

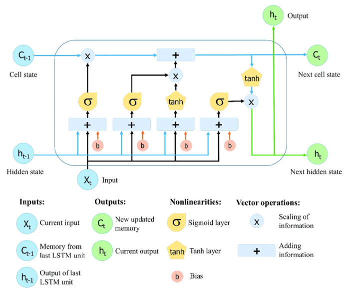

Fig. 3. LSTM Architecture [20]

LSTM Module will contain LSTM-classifier, LSTM is an application of the Recurrent Neural Network which has the ability to learn about long-standing dependencies [5], where the structure of the LSTM cell itself is depicted as shown in Figure 3. In Figure 3, every *t* time of this LSTM cell will be controlled by various logic gates, which aim to maintain or reset the values in the cell, in Figure 3 itself there are three types of gates, the gates are located sequentially from the left side, the gate consists of *Forget gate* (f_t), *Input gate* (i_t) , and *Output gate* (o_t) all of which have sigmoid activation functions, then there is one gate that uses the tanh function called *candidate value gate*.

LSTM Classifier will build using Tensorflow with Python programming language. This module will have sequential architecture and contains LSTM Layer, Dropout Layer, and Fully Connected Layer using RELU and Softmax activation function, this module will give LSTM prediction value output, and this value will pass to the Naive Bayes module.

2) Naive Bayes Module

Naive Module will contain Naive Bayes Classifier; Naive Bayes is a method for classifying based on Naive Bayes theory [21], this method is a classification using a simple probabilistic approach. We need to calculate a set of probabilities by adding up the frequency and combination of values from the data. This classification model method will consider attributes it is not interdependent.

$$P(H|E) = \frac{P(H|E)P(H)}{P(E))} \quad (1)$$

$$P(H|E) = P(E1|H) \times P(E2|H) \times \cdots \times P(En|H) \times P(H) \quad (2)$$

So that values can be assigned to these attributes and the patterns resulting from these calculations will be used for classification, the Naive Bayes system itself is classified as *supervised learning* in the application of *machine-learning* [21], it defined like equation 1 and equation 2, in Naive Bayes, the calculation of the probability of an event H which is a condition in data E is carried out by first calculating the probability from data E with condition H **(P(H|E))**, after that, the result **(P(H|E))** multiplied by the probability condition H **(P(H))**, then divided by the probability data E (P(E)). So that the detection of DoS attacks can be carried out, using a calculation of the probability of the attack occurring on the data then divided by the probability of data E [21].

IV. Experimental Framework

This section describes the datasets and evaluation metrics used in the experimental framework. We also describe the data preprocessing procedure

A. Dataset

The experimental evaluation framework will be uses a widely used datasets such as NSL-KDD and CICIDS 2017 [2], NSL-KDD datasets are one of the most popular datasets used on NIDS Performance, this datasets introduced by Tavallace et al but this dataset are out of date and lack of traffic diversity and feature sets, to solve that problem CICIDS2017 will be used because that datasets relative new, but this two datasets are not specified for SDN architecture, this can be happen because of the lack of public datasets about DoS attack on SDN architecture, beside that SDN architecture is still under development [2], SDN dataset generated manually by several researcher but it closed and quite rare to find it, so many of researcher still use conventional dataset to evaluate their model [2], but in this paper used another dataset related to SDN architecture, the dataset provided by Ahuja et al research to develop deep learning anomaly detection system on SDN based network [22] the usage this dataset is to proving our proposed system will be worked on SDN architecture and achieve significantly improvement on performance, and lastly using simulation dataset created using SDN simulation using Iperf3 and Hping3 to generate network traffic and then captured by network sniffing application.

B. Data Preprocessing

On NSL-KDD and CICIDS 2017, we performed one-hot encoding, scaling, and label transformation for features needed. We did not do a feature selection process for this dataset when input to the LSTM module, but we selected some features, mainly LSTM prediction result and protocol type feature. For the SDN-DL dataset, we performed feature selection using Heatmap, Chi-Square, Tree Classifier, and Data Slice. Besides that, we performed missing value and duplicate value handling. Lastly, we performed normalization using min-max methods. We did label encoding for some features like protocol type, and the last dataset we used is the simulation dataset we generated before we performed data normalization and feature selection because this data was used to train our model to simulation data. .

C. Evaluation Methods

This paper uses parameters such as Accuracy, Precision and False Negative Rate to calculate detection system performance. To produce that, we need to make a confusion matrix. Based on Qin et al. confusion matrix will be like Table I

TABLE I
CONFUSION MATRIX

Real Condition	**Detection Result**	
	Intrusion	Normal
Intrusion	True Positive	False Negative
Normal	False Positive	True Negative

- **True Positive (TP)** is parameter of DoS Packet classified as DoS Condition.
- **True Negative (TN)** is parameter of Normal Packet classified as Normal Condition.
- **False Positive (FP)** is parameter of Normal Packet classified as DoS Condition.
- **False Negative (FN)** is parameter of DoS Packet classified as Normal Condition..

Based on the parameter we got from Table I, we can calculate performance parameters such as Accuracy, Precision, and False Negative Rate. We can use an equation based on Li et al. and Aljarwaneh et al. research to calculate the performance parameter.

- ***Accuracy*** compares the correct classification with the total number in the dataset.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

- ***Precision*** is part of the data that is classified as positive and has a true positive value.

$$Precision = \frac{TP}{TP + FP} \quad (4)$$

- ***False Negative Rate*** is a comparison that shows the number of incorrect packets that are classified as true.

$$FNR = \frac{FN}{TP + FP} \quad (5)$$

From Equation 3 - 5 will be seen some parameter, this parameter consist of four values: TN (True Negative), TP (True Positive), FN (False Negative), and FP (False Negative).

V. Experiment and Result

In this section, we describe three evaluations and discuss the results. Evaluation 1 will use used SDN-DL dataset, and then we will compare proposed methods using some Vanilla algorithms on Sklearn and Tensorflow library, such as ANN, Naive Bayes, and LSTM. After that, all methods will be evaluated using evaluation parameters. Evaluation 2 will use NSL-KDD and CICIDS datasets and compare using the performance of Tang et al. research. In Evaluation 3, we evaluate the performance of the P4-based SDN network using the dataset we generated before and present the result.

A. Evaluation 1 - SDN-DL Dataset Evaluation

The goal of evaluation 1 is to carry out comparative analysis of proposed system with research before using same dataset, this evaluation will focused to find feature have optimal performance which have highest Accuracy and lowest False Negative Rate, on this evaluation we perform data exploration, cleansing, transform, and normalize data using Min-Max Scaling, after that we split data to three parts: training, validation, and testing, which contains 70% data for training from training split we split again 70% data for training and 30% data for validation, and lastly 30% for testing, first process conducted is data exploration on SDN-DL Dataset didn't contains any missing value and duplicated value, then the data majority on numerical format beside on some attribute like protocol still on string format so attribute transformation needed, on this research we using Label Encoding to transform object type to numerical type, after that on this data we found outlier in some attribute and we choose to delete the outlier on pktrate and pktperflow attribute, but on duration, dst, and pktcount. we decided to keep the outlier because of the reason behind that. For example, duration its have many variations in session time. Besides, the DoS attack can be exploited during abnormal times, so we decided to keep the outlier.

Next step we conducted feature selection because on research conducted before using this dataset, researcher using

TABLE II
Feature Selection Result

Algorithm	Accuracy (%)		Precision (%)		FNR (%)	
	Train	Test	Train	Test	Train	Test
ANN (Full Feature)	**92.96**	**92.85**	**93**	**92**	**14.5**	**14.6**
ANN (Chi Square)	69.2	69.29	78	78	43.9	43.9
ANN (Extra Tress)	93.81	93.87	94	94	15.6	15.2
ANN (Heatmap Slice)	84.99	85.10	84	84	10.9	10.8
LSTM (Full Feature)	**94.24**	**94.09**	**94**	**94**	**14.2**	**14.2**
LSTM (Chi Square)	80.78	80.39	81	81	16.6	16.6
LSTM (Extra Tress)	90.79	90.78	90	90	13.4	13.4
LSTM (Heatmap Slice)	88.57	87.86	88	87	12.6	12.5
NB (Full Feature)	60.79	61.08	30	31	-	-
NB (Chi Square)	60.79	61.08	30	31	-	-
NB (Extra Tress)	**61.16**	**60.51**	**31**	**30**	-	-
NB (Heatmap Slice)	61.01	60.78	31	30	-	-

all Deep-Learning methods so their didnt explore any feature possiblity for Deep-Shallow combination methods, we select feature using four ways, the first way we use all feature, second we using chi square calculation to get feature importance score, we got *'dst','src','Protocol','pktcount','pktcount'* feature selected, third we using extratreesclassifier to calculate methods importance, then from this methods we got *'pktrate','pktperflow','Protocol', 'src','dst'*, and the last methods used is check the correlation heatmap and combined with another selected feature we decided to choose *'dst','src','Protocol','bytecount'*, after the feature selected we perform normalization process using two methods: Normalization, and Standarization methods, but after performing evaluation some algorithm we used cant handle standarization value so we choose Normalization for feature selection evaluation, we evaluate feature selection using LSTM, VanillaNB, and ANN methods and we got result described on Table II.

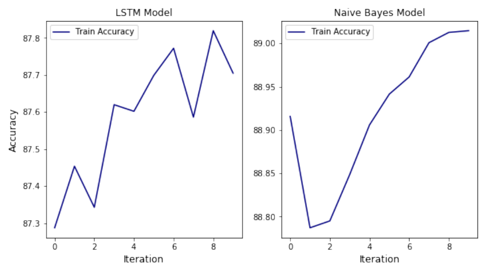

Fig. 4. LSTM and Naive Bayes Model Accuracy on Training Session

From Table II we know Deep-Learning Based methods such as LSTM and ANN preferable to use all feature, but for shallow learning algorithm on this research Naive Bayes needed feature selection process, beside that we know Naive Bayes Methods have lousy performance because only can detect non-anomalies value, from this result we know heatmap method have lowest performance value, so we decided to improve the performance using proposed methods, when developing proposed methods we use 10 epoch to train LSTM model, and using Cross-Validation to find optimum var_smoothing value, we use 10 fold and 1000 tries on Naive Bayes model, we get the optimum value is 0.732596542821523, and the result we can make model with accuracy up to 89%, the development accuracy can be seen on Figure 4, we improving Naive Bayes Slice Methods performance from 61% to 89%, after created our proposed schema we conduct test and comparing with research before and the result is on Table III - V.

TABLE III
ACCURACY PERFORMANCE COMPARISON

Algoritma	**Accuracy (%)**		
	Training	**Validation**	**Testing**
LSTM + NB (Combined)	88.77	88.18	88.02
LSTM	88.59	87.89	88.57
Naïve Bayes	62.01	61.46	61.85
ANN	87.698	87.67	87.69
CNN-LSTM [22]	**99.48**		

TABLE IV
PRECISION PERFORMANCE COMPARISON

Algoritma	**Precision (%)**		
	Training	**Validation**	**Testing**
LSTM + NB (Combined)	85.21	82.41	84.16
LSTM	86.02	85.03	83.14
Naïve Bayes	51.35	51.20	51.30
ANN	83.29	83.51	83.36
CNN-LSTM [22]	**99.55**		

TABLE V
FNR PERFORMANCE COMPARISON

Algoritma	**FNR (%)**		
	Training	**Validation**	**Testing**
LSTM + NB (Combined)	13.60	10.78	14.53
LSTM	15.72	16.67	10.58
Naïve Bayes	64.16	62.26	64.47
ANN	14.09	14.09	14.09
CNN-LSTM [22]	**3**		

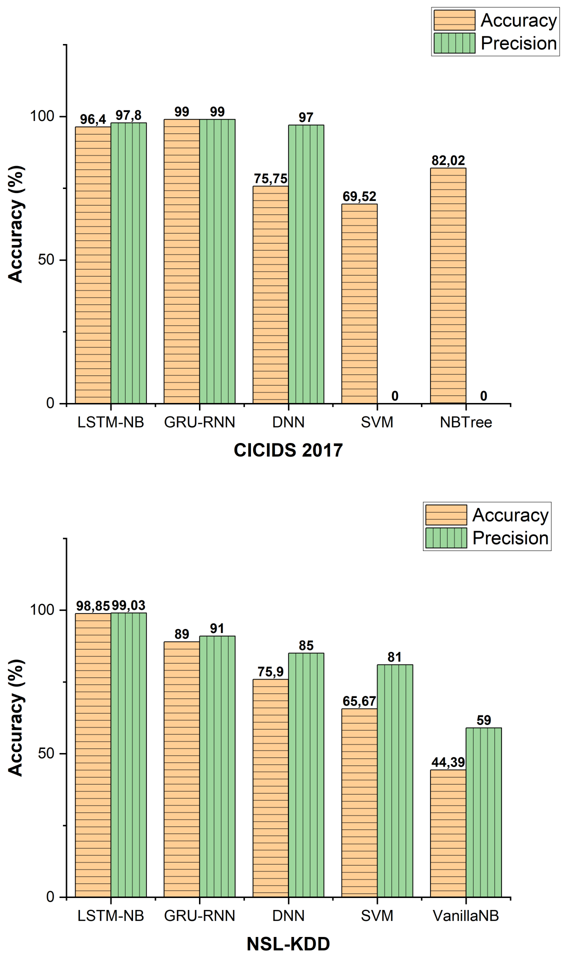

Fig. 5. CICIDS 2017 and NSL-KDD Performance Comparison

From Table III - V our proposed system still has the lowest performance compared to research conducted by Ahuja et al., but it is expected because of research before their use all Deep Learning methods to solve this problem, but like we said before use Deep Learning meaning need more resource to run and did not have feature knowledge to configure the network. However, if we compared it with another vanilla machine-learning and deep learning algorithm, our proposed system has the highest accuracy, precision, and lowest FNR rate.

B. Evaluation 2- NSL-KDD and CICIDS Dataset Evaluation

The goal of evaluation 2 is to carry out a comparative analysis with research conducted by Tang et al. the main reason this evaluation was conducted was to evaluate the proposed system with a public network dataset. Besides that, Tang et al. proposed using RNN-based methods to compare the LSTM-based model with the RNN-based model for this evaluation. Like evaluation one, we will split data into three parts: Training, Validation, and Testing, which contains 70%:30% data partition on Training and Testing. After that, we conduct data preprocessing, and then we use all attribute to train our LSTM-NB model; the performance parameter we used to make the comparison is accuracy and precision. FNR was not used because research before did not describe that value, and from this evaluation, we get the result described in Figure 5.

From Figure 5 our proposed system can achieve an accuracy of 99.85% on the NSL-KDD dataset and 96.4% on CICIDS2017, and that means our proposed system works significantly better than DNN, SVM, VanillaNB, and Tang et al. proposed system GRU-RNN on NSL-KDD dataset. Besides that, on CICIDS 2017, our proposed system achieved slightly lower than the GRU-RNN dataset, meaning the proposed system worked significantly accurate and had a good performance. To achieve this value, we conduct data transformation using one-hot encoding and data preparation; after that, we split data 70%:30%. We developed with the same methods as evaluation one, but the difference is that we did not choose any feature for the LSTM module, but we selected 'lstm_result_1', 'lstm_result_2', 'lstm_result_3', and protocol name on Naive Bayes module. We developed the Naive Bayes module using Grid Cross-Validation methods to get the optimum var_smoothing value, and the value used is 0.732596542821523. Besides that, on this evaluation, we got an FNR rate range between 1-2% and this value was lower than previously proposed methods CNN-LSTM.

C. Evaluation 3- Simulation Dataset Evaluation

The goal of evaluation 3 is to know feature can be used on P4-based SDN Network, because P4 have some difference feature can be extracted with OpenFlow SDN, we extracted 'src'. 'dst'. 'lenght', and 'protocol'. That parameter we use to evaluate our model performance with P4-Mininet Simulation, this simulation generate using Hping3 and Iperf, then we conduct 30 minutes network simulation, after that we save data based on session we conduct, first session we conduct normal traffic with Iperf3 and variate windows and packet lenght between 1000-3000 Bytes, and other session we conduct attack session we conduct UDP, SYN, TCP Flood, after that we preprocessed data, and use proposed system, and performance we got is accuracy 100% with FNR 0%, this can be happen because the variation of DoS attack is obsolute. However, this proved system can classification DoS attack.

VI. Conclusion Future Works

We proposed P4 Based SDN DoS Attack Detection in this system using LSTM-NB. The system we proposed consists of three modules with two main modules, the first module is a data collector and normalization module, second is the LSTM module, the module that makes the predicted value named lstm_result. The last module will use this value to predict with high accuracy and low FNR, and the last module is the Naive Bayes Module used to compute the final classification of anomaly.

The proposed schema can be used; the evaluation results prove it. On SDN-DL evaluation, we got available SDN general features that can be used. The evaluation conducted proposed schema got 88% accuracy, 85% precision, and 10% FNR; this performance result is better if compared with other algorithms but lesser if compared with CNN-LSTM. from evaluation with NSL-KDD and CICIDS, our proposed schema achieved $\geq 96\%$ accuracy. The last data get from simulation using P4-Mininet achieved 100% accuracy. It proves that LSTM-NB (Proposed Methods) can be implemented with good performance. For future work, the architecture used can be remodelled with a better tuning approach or another combination method and make a model that can detect more than the DoS attacks type currently known by the system.

References

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[2] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Intrusion detection in sdn-based networks: Deep recurrent neural network approach," in *Deep Learning Applications for Cyber Security*. Springer, 2019, pp. 175–195.

[3] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted ddos attack detection with p4 language," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[4] Y. Qin, J. Wei, and W. Yang, "Deep learning based anomaly detection scheme in software-defined networking," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–4.

[5] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, "Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment," *IEEE Access*, vol. 8, pp. 83 765–83 781, 2020.

[6] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 128, pp. 33–55, 2019.

[7] A. Sangodoyin, T. Sigwele, P. Pillai, Y. F. Hu, I. Awan, and J. Disso, "Dos attack impact assessment on software defined networks," in *International Conference on Wireless and Satellite Systems*. Springer, 2017, pp. 11–22.

[8] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2020, pp. 391–396.

[9] S. R. Talpur and T. Kechadi, "A survey on ddos attacks: Router-based threats and defense mechanism in real-world data centers," in *2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 978–984.

[10] O. Yevsieieva and S. M. Helalat, "Analysis of the impact of the slow http dos and ddos attacks on the cloud environment," in *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*. IEEE, 2017, pp. 519–523.

[11] S. Anwar, J. Mohamad Zain, M. F. Zolkipli, Z. Inayat, S. Khan, B. Anthony, and V. Chang, "From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions," *Algorithms*, vol. 10, no. 2, p. 39, 2017.

[12] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[13] M. Zhang, J. Guo, B. Xu, and J. Gong, "Detecting network intrusion using probabilistic neural network," in *2015 11th International Conference on Natural Computation (ICNC)*. IEEE, 2015, pp. 1151–1158.

[14] Y. Li and Y. Lu, "Lstm-ba: Ddos detection approach combining lstm and bayes," in *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*. IEEE, 2019, pp. 180–185.

[15] V. Zlomislić, K. Fertalj, and V. Sruk, "Denial of service attacks, defences and research challenges," *Cluster Computing*, vol. 20, no. 1, pp. 661–671, 2017.

[16] M. Aamir and M. Arif, "Study and performance evaluation on recent ddos trends of attack & defense," *International Journal of Information Technology and Computer Science*, vol. 5, no. 8, pp. 54–65, 2013.

[17] H. Polat, O. Polat, and A. Cetin, "Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, vol. 12, no. 3, p. 1035, 2020.

[18] S. Azodolmolky, *Software defined networking with OpenFlow*. Packt Publishing, 2013, vol. 153.

[19] P4.org, "P4 Language Evolution," https://opennetworking.org/news-and-events/blog/p4-language-evolution/, 2015, [Daring; diakses 23-Oktober-2021].

[20] X.-H. Le, H. V. Ho, G. Lee, and S. Jung, "Application of long short-term memory (lstm) neural network for flood forecasting," *Water*, vol. 11, no. 7, p. 1387, 2019.

[21] A. Mehmood, M. Mukherjee, S. H. Ahmed, H. Song, and K. M. Malik, "Nbc-maids: Naïve bayesian classification technique in multi-agent system-enriched ids for securing iot against ddos attacks," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 5156–5170, 2018.

[22] N. Ahuja, G. Singal, and D. Mukhopadhyay, "Dlsdn: Deep learning for ddos attack detection in software defined networking," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2021, pp. 683–688.