Intrusion Detection using Deep Neural Network Algorithm on the Internet of Things

Syariful Ikhwan
[a]*Doctoral Program of Information System*
School of Postgraduate Studies
Diponegoro University
Semarang, Indonesia

[b]*D3 Telecommunication Engineering*
Institut Teknologi Telkom Purwokerto
Banyumas, Indonesia
syariful@ittelkom-pwt.ac.id

Adi Wibowo
Computer Science Informatics
Department of Science and Mathematics
Faculty
Diponegoro University
Semarang, Indonesia
bowo.adi@live.undip.ac.id

Budi Warsito
Statistics Department of Science and
Mathematics Faculty
Diponegoro University
Semarang, Indonesia
budiwrst2@gmail.com

2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT) | 978-1-6654-6030-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/COMNETSAT56033.2022.9994499

***Abstract*— The increasing use of IoT devices on future networks is very helpful for humans in their lives. However, the increase in devices connected to IoT networks also increases the potential for attacks against those networks. Vulnerabilities in Internet of Things (IoT) networks can be exposed at any time. Artificial intelligence can be used to protect the IoT network by being able to detect attacks on the network so that they can be prevented. In this study, network detection was carried out using the Deep Neural Network (DNN) algorithm. The test was carried out using the UNSW Bot-IoT dataset with a comparison of training data of 75% of the overall data. The results obtained show the ability of the algorithm to detect attacks on average with 99.999% accuracy. The validation loss and training loss look very small. In this study, there is a validation loss that still occurs in overfitting, but the difference is very small.**

Keywords*— DNN, IoT, Intrusion Detection, Network, Bot-IoT Dataset

I. INTRODUCTION

An intrusion Detection System (IDS) is a system that is used to detect an attack on the network by detecting all packets going to the network and then selecting the packet whether it is an attacker group or not [1]. IDS was first introduced in 1980. Since then, it has developed using various methods to detect attacks. There are several things that later became important issues in the development of IDS. The issue is the ability of IDS to separate between attacks that are real attacks and attack that are not attacks[2]. Sometimes the IDS incorrectly identifies the real attack and considers the actual access to be an attack. The ability of IDS to detect a large number of passing packets at one time is also an issue that is quite important to be resolved. Another issue that the researchers focus on is the changing attack patterns that occur over time. Attacks that have not been previously known, termed "unknown attacks", are then difficult to detect by IDS.

The application of intrusion detection on Internet of Things (IoT) networks is currently an interesting research topic. This is based on predictions that, in the future, the use of IoT technology will continue to grow rapidly. IoT devices send various data packets to the internet network in continuous and massive amounts. Therefore, it is not possible to detect intrusion using traditional methods. Detection of attacks on IoT can be grouped in two ways, namely by using statistics and using a machine learning approach. The development of machine capabilities in the field of artificial intelligence is then proposed to be a better way to solve problem of detecting attacks [3]. Various machine learning methods are then used to improve the IDS's ability to detect known and unknown attacks [4].

In [5] explained that Denial of Service, Data Type Probing, Malicious Control, Malicious Operation, Scan, Spying, and Wrong are attacks and anomalies that can cause failures in IoT systems. To counter the attacks, tests were carried out using the Logistic Regression (LR) algorithm, Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN) with the Distributed Smart Space Orchestration System (DS2OS) dataset. The results obtained show that Random Forest has a better performance than other algorithms. However, in this study, it can be seen that the more data is tested, the ANN's performance is getting closer to RF performance.

Detection of traffic anomalies using the proposed Channel Boosted and Residual learning based deep Convolutional Neural Network (CBR-CNN) is better than existing machine learning techniques and gives promising results on the validation set and shows a significant performance improvement on datasets that have new attacks [6]. Another method based on a hybrid neural network is also proposed to detect anomalies by analyzing certain features [7]. A one-dimensional convolution network is implemented to analyze sequence features in a hybrid neural network, while a deep neural network is used to study the characteristics of high-dimensional feature vectors including general statistical features and environmental features. It is concluded that the proposed method can be applied to anomaly detection applications with reasonable performance.

Network infrastructure is more vulnerable to cyber-attacks because it is connected to the internet. The most widely used attacks are distributed denial of service (DDoS) attacks that disrupt services. The most important factor in combating DDoS attacks is early detection and segregation of network traffic. Research [8] proposes using a deep neural network as a deep learning model that detects DDoS attacks on packet samples captured from network traffic. The results of the experiment conducted on the CICDDoS2019 dataset containing the types of DDoS attacks created in 2019 were observed. It was observed that attacks on network traffic were detected with 99.99% success, and the attack types were classified with an accuracy rate of 94.57%. The high accuracy values obtained indicate that deep learning models can be used effectively in combating DDoS attacks.

2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)

The research in this paper examines the ability of Deep Neural Network (DNN) to recognize attacks contained in a number of datasets. The tested dataset consists of a set of separated data. Then testing is also carried out on the data that has been collected in large quantities. The contribution of this paper is first to determine the ability of DNN to recognize attacks or not on small datasets. The implementation is done by analyzing the dataset, which is divided into several files. Second, knowing the capabilities of DNN when the data provided is in large quantities. The dataset used is a collection of the previous dataset. Third, compare DNN capabilities when the data is small and when the data is large.

II. METHODOLOGY

The purpose of this study is to apply a deep neural network algorithm to IoT network traffic. The dataset used is the Bot-IoT dataset created by University of New South Wales (UNSW) Center for Cyber Security (ACCS) Canberra, Australia (in the Cyber Range Labs)[9] [10]. This dataset was created using smart home devices. The smart home devices include weather monitoring systems, smart cooling devices, smart lights, smart door opening and closing systems, and others. The traffic on the network was a mix of regular and botnet traffic. The source files for the dataset are offered in a variety of forms, such as the original pcap files, the produced argus files, and csv files. To help with labeling, the files were divided based on attack category and subcategory. More than 72.000.000 records can be found in the 69.3 GB-sized collected pcap files. The extracted flow traffic is 16.7 GB in size and is in csv format. The DDoS and DoS attacks are further categorized according to the protocol employed in the dataset, which also includes OS and Service Scan, Keylogging, and Data Exfiltration assaults.

In this study, the dataset used is a dataset that has been in the form of a CSV file extracted from the raw data and then shared publicly. The file used is only a small part of the entire CSV file. Files are then divided into two groups namely small and large. The dataset used for the small category has a total frame of 1,000,000 lines, consisting of attack and normal frame. There are 10 parts of the dataset, each of which has a different number of attacks. The comparison between training data and validation data is 75% and 25%. Data comparison split is done automatically using the train_test_split library in Python.

On a large number of datasets, the test is performed by combining the datasets 1 to 5 and the combined datasets 6 to 10. So, we get two files, each containing 5 million lines of frame. All data were analyzed using Python on the infrastructure provided by Google, namely Google Colabs. Frame attacks on small and large groups consisting of DoS, Theft, and Reconnaissance attacks.

The flow of testing the dataset in this study was carried out in several phases. First, the dataset was inputted and read in the form of a csv file extension. The imported dataset does not yet have a header; therefore, the data is then given a header according to the column description used. The headers represent the features that will be used in data analysis. Each column is a feature that is different from the other columns. The next step is to delete the columns that have NaN values (no value) and columns that have values other than numeric. Values other than numeric are removed because the machine can only calculate numeric values. This is done so that the trend value of the analyzed data can be calculated. Even if all non-numeric values are removed, one feature that is used as a marker feature is excluded from deletion. The selection of the features used will greatly affect the results obtained later [11]. In this study, the feature column used as a marker is the category column. A summary of all the features used in this study can be seen in Table 1.

DNN is a subtype of MLP (Multilayer Perceptron), a sort of Feed Forward Neural Network (FFN) with more than two layers, which has one input layer, one output layer, and more than one hidden layer. Each layer contains a number of neurons, all of which are fully linked to one another in the forward direction. Deep Neural Network uses a feature vector as its input. This vector size always has a fixed length. Resizing the feature vector means recreating the entire neural network. Although feature vectors are called "vectors", this is not always the case [12]. In this study, 1-dimensional vector input was used.

The IoT is subject to various types of attacks due to vulnerabilities present in devices. Due to the many features of IoT network traffic, machine learning models take time to detect attacks [13]. Feature selection or reduction is an important process for an intrusion detection system (IDS) in finding optimal features. Irrelevant features present in the data set increase the load on computing resources and affect system performance [14]. Table 1 displays the features and feature names. All features found in the dataset are set as headers. Then after all the headers are given, the values other than numeric are omitted so that it can be seen in the second row in Table 1, there are 21 features used in the analysis process. An explanation of the names of the features shown in Table 1, can be seen in the study [15].

TABLE I. BOT-IOT DATASET FEATURES

Features	Features Name
All Features	pkSeqID, Stime, flgs, proto, saddr, sport, daddr, dport, pkts, bytes, state, ltime, seq, dur, mean, stddev, smac, dmac, sum, min, max, soui, doui, sco, dco, spkts, dpkts, sbytes, dbytes, rate, srate, drate, attack, category, subcategory
Features used	pkSeqID, stime, pkts, bytes, ltime, seq, dur, mean, stddev, sum, min, max, spkts, dpkts, sbytes, dbytes, rate, srate, drate, attack, category

The next phase is the preprocessing phase. At this phase, we encode the feature vector into two functions. The first is to encode it into the z-score function, and the second is to create a dummy variable from the category column. After the preprocessing phase is complete, the next step is to train the neural network to classify data in the category column. In this test, two hidden layers were carried out. The results obtained are then used to display the values of training loss, validation loss, and the accuracy value of the model.

Accuracy refers to the amount of data is predicted to be correct for the entire test dataset. If the accuracy value increases, then the registered model becomes right. Accuracy is calculated as

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

where, true positive (TP) is when the model classifies the attack as an attack. True negative (TN) is when the model

classifies normal traffic as normal. False positive (FP) is when the model classifies normal traffic as an attack while False negative (FN) is when the model classifies an attack as normal traffic.

The loss function is the best parameter, and using it is crucial for getting better results. The difference between the goal and predicted values can be calculated using the loss function. It attempts to learn a function's approximate identity by minimizing reconstruction error during the learning phase. The loss function aids in determining the degree to which the predicted value deviates from the desired value. To determine the loss function and categorize the assault, the target and features were fed into the model.

III. Results and Discussion

The tests we have carried out present results that describe the ability of the DNN algorithm to study the data provided and then validate the model that has been obtained from the training. The dataset used in the training is a public dataset that is generally available. Thus it can be seen how the DNN algorithm can recognize intrusions. The tests carried out got the results as shown in Table 2. Each dataset tested got the best value in a fairly good range. The accuracy obtained is in the range of 99.99888% to 100%. These values are obtained after 5 epochs of repetition. In the tests made, limited to 5 epochs, it is hoped that later the system will provide the same treatment for each dataset.

TABLE II. Test Results

Test	Total Frame (rows)	Normal	Attack	Category	Accuracy (%)
Dataset 1	1,000,000	1993	998,007	DoS	100
Dataset 2	1,000,000	4941	169,840	DoS	99.9988
			823,632	Reconnaissance	
			1,587	Theft	
Dataset 3	1,000,000	107	999,893	DoS	100
Dataset 4	1,000,000	98	999,902	DoS	99.9996
Dataset 5	1,000,000	76	999,924	DoS	99.9996
Dataset 6	1,000,000	33	999,967	DoS	100
Dataset 7	1,000,000	41	999,959	DoS	100
Dataset 8	1,000,000	27	999,973	DoS	100
Dataset 9	1,000,000	38	999,962	DoS	100
Dataset 10	1,000,000	30	999,970	DoS	100

Figure 1 shows a graph of the training loss and accuracy loss that occurred. In this graph, it is known that the trend of loss of accuracy and loss of training is good. Loss training can follow loss validation in 8 datasets, while in datasets 4 and 5 (Fig.1(d) and Fig.1(e)), loss validation and loss training do not find any similarities. The loss of training that occurs in the test, as shown in Fig. 1, begins to decrease significantly during the first epoch. Then, the training loss value tends to remain the same until the end of the training.

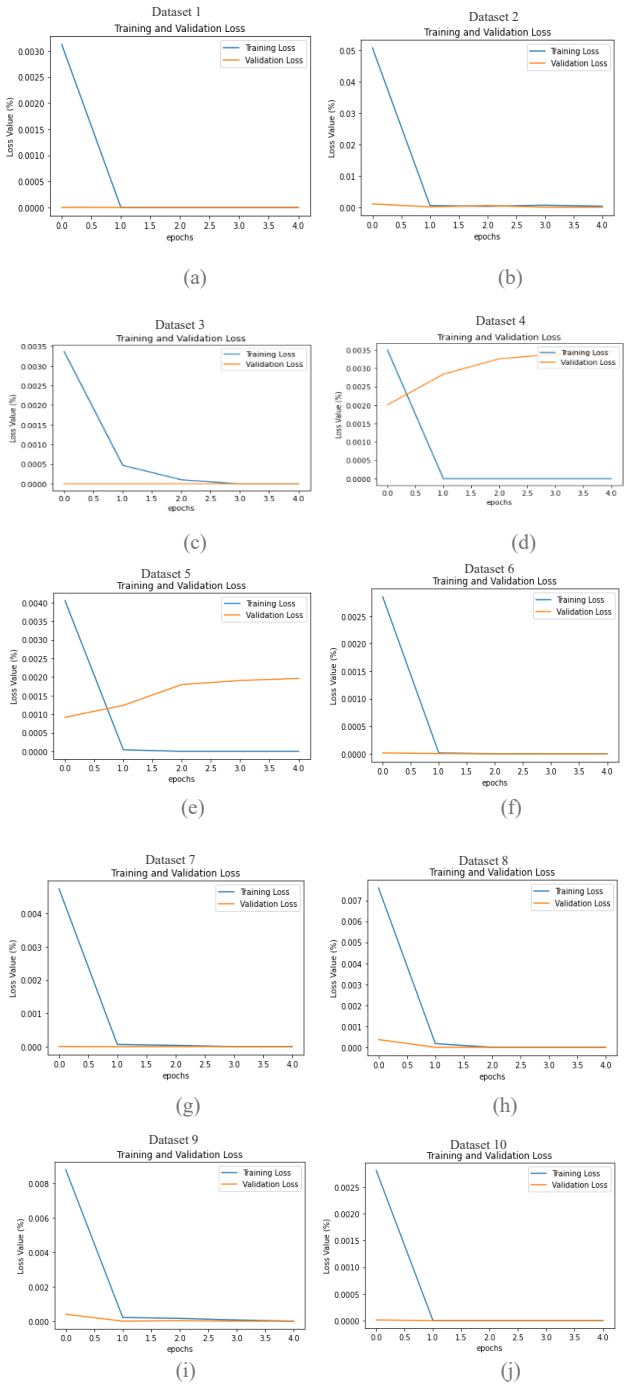


Fig. 1. Graph of the results of testing the DNN algorithm on datasets.

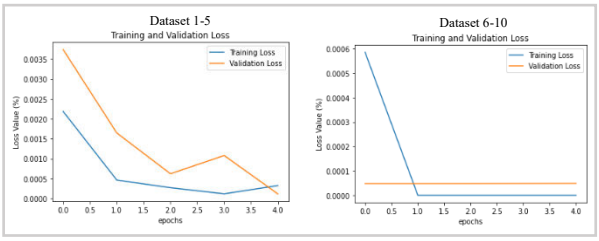
TABLE III. Test Results Larger Group

Test	Total Frame (rows)	Normal	Attack	Category	Accuracy (%)
Dataset 1-5	5,000,000	7215	3169559	DoS	99.9997
			1821639	Reconnaissance	
			1587	Theft	
Dataset 6-10	5,000,000	169	4999831	DoS	99.9999

The test results in the second scenario, namely in the larger data group, can be seen in Table 3 that the accuracy value is not much different when carried out in the first scenario. The accuracy value is 99.999%. The value obtained is almost the

same as the test in the first scenario. As explained earlier that the first test uses 1 million rows of data while the second scenario uses 5 million rows of data. When compared, we get a graph that is almost similar between the first and second scenarios. This means that the model has been as expected. From Table 2 can also be seen that the addition of the number of rows used in the model does not affect the results obtained.

Figure 2 shows the graph obtained in the large group test. Figure 2(a) shows that there is overfitting but not too big. It should be because datasets 4 and 5 resulting bad results such as shown in Fig. 1. It can be seen that the trend of validation loss and train loss has followed each other. This means that the model made works well and meets expectations.



(a) (b)

Fig. 2. Graph of the results of testing the DNN algorithm on a larger amount of data.

As shown in Table 2 can be said that DNN is able used and developed to detect intrusions on IoT networks. Further development will also be usefully made in prevention efforts against such intrusions. That can be addressed by adding a firewall and the like on the networks. If the network is entered by a suspicious intrusion as recognized by the intrusion detector, the firewall acts to block the incoming intrusion.

Our work is similar to that carried out in [16], but in the study they are carried out unsupervised feature learning using the nonsymmetric deep autoencoder (NDAE) method on the NSL-KDD dataset. The classifier used is stacked NDAEs. While in our research, we use supervised learning features and the UNSW Bot-IoT dataset. Research [17] uses a vector convolutional deep learning (VCDL) approach to analyze anomalies in IoT traffic using all Bot-IoT dataset traffic records. The results obtained show an accuracy of 99.74%. The results obtained are better than other comparison methods.

IV. Conclusion

Intrusion detection in this study was implement using the DNN algorithm on the UNSW BoT-IoT dataset. The results obtained after the testing was very good. Thus, it can be said that DNN can be applied to distinguish between attacks and non-attacks on IoT networks. In the tests carried out, the feature elimination process at the preprocessing stage will determine the results obtained. Elimination of features in this study is still manually. In the future, it is hoped that automatic feature selection will be carried out by the system. Features other than numeric can also actually be considered to be included in the calculation. This is done by converting it to numeric using the data encoding process. Further research in the application of this DNN, how to make the attack dataset and normal balanced. Subsequent research must consider the balance of the data so that the model obtained becomes more tested. Various dataset balancing methods can be used at the preprocessing phase.

References

[1] Z. Zhang, Q. Liu, S. Qiu, S. Zhou, and C. Zhang, "Unknown Attack Detection Based on Zero-Shot Learning," *IEEE Access*, vol. 8, pp. 193981–193991, 2020, doi: 10.1109/ACCESS.2020.3033494.

[2] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion Detection Systems : A Cross-Domain Overview," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3639–3681, 2019.

[3] A. S. Alzahrani, R. A. Shah, Y. Qian, and M. Ali, "A novel method for feature learning and network intrusion classification," *Alexandria Engineering Journal*, vol. 59, no. 3, pp. 1159–1169, 2020, doi: 10.1016/j.aej.2020.01.021.

[4] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences (Switzerland)*, vol. 9, no. 20, 2019, doi: 10.3390/app9204396.

[5] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. M. A. Hashem, "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches," *Internet of Things (Netherlands)*, vol. 7, p. 100059, 2019, doi: 10.1016/j.iot.2019.100059.

[6] N. Chouhan, A. Khan, and H. ur R. Khan, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," *Applied Soft Computing Journal*, vol. 83, p. 105612, 2019, doi: 10.1016/j.asoc.2019.105612.

[7] C. Ma, X. Du, and L. Cao, "Analysis of multi-Types of flow features based on hybrid neural network for improving network anomaly detection," *IEEE Access*, vol. 7, pp. 148363–148380, 2019, doi: 10.1109/ACCESS.2019.2946708.

[8] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed forward based deep neural network model," *Expert Syst Appl*, vol. 169, no. December 2020, p. 114520, 2021, doi: 10.1016/j.eswa.2020.114520.

[9] UNSW, "Bot-IoT Dataset," 2018. https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE?path=%2FCSV%2FEntire Dataset (accessed Dec. 22, 2021).

[10] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041.

[11] C. Kalimuthan and J. Arokia Renjit, "Review on intrusion detection using feature selection with machine learning techniques," *Mater Today Proc*, vol. 33, pp. 3794–3802, 2020, doi: 10.1016/j.matpr.2020.06.218.

[12] J. Heaton, *Applications of Deep Neural Networks*, 1st ed. Heaton Research, Inc., 2020.

[13] P. Nimbalkar and D. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," *ICT Express*, vol. 7, no. 2, pp. 177–181, 2021, doi: 10.1016/j.icte.2021.04.012.

[14] D. Kshirsagar and S. Kumar, "An efficient feature reduction method for the detection of DoS attack," *ICT Express*, vol. 7, no. 3, pp. 371–375, 2021, doi: 10.1016/j.icte.2020.12.006.

[15] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019, doi: 10.1016/j.future.2019.05.041.

[16] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A Deep Learning Approach to Network Intrusion Detection," *IEEE Trans Emerg Top Comput Intell*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

[17] B. A. Bhuvaneswari and S. S., "Anomaly detection framework for Internet of things traffic using vector convolutional deep learning approach in fog environment," *Future Generation Computer Systems*, vol. 113, pp. 255–265, 2020, doi: 10.1016/j.future.2020.07.020.