IoT botnet attack detection using deep autoencoder and artificial neural networks

Deris Stiawan[1*], Susanto[2], Abdi Bimantara[1], Mohd Yazid Idris[3], and Rahmat Budiarto[4]
[1] Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya
Palembang, 30139 Indonesia
[e-mail: deris@unsri.ac.id; abdibimantara91@gmail.com]
[2] Faculty of Engineering Science, Universitas Bina Insan
Lubuklinggau, 31626 Indonesia
[e-mail: susanto@univbinainsan.ac.id]
[3] School of Computing, Faculty of Engineering, Universiti Teknologi,
Johor Bharu, 81310 Malaysia
[e-mail: yazid@utm.my]
[4] College of Computer Science and IT, Albaha University
Alaqiq, 65779-7738, Saudi Arabia
[e-mail: rahmat@bu.edu.sa]
[*]Corresponding author: Deris Stiawan

Received October 23, 2022; revised March 23, 2023; accepted April 11, 2023; published May 31, 2023

Abstract

As Internet of Things (IoT) applications and devices rapidly grow, cyber-attacks on IoT networks/systems also have an increasing trend, thus increasing the threat to security and privacy. Botnet is one of the threats that dominate the attacks as it can easily compromise devices attached to an IoT networks/systems. The compromised devices will behave like the normal ones, thus it is difficult to recognize them. Several intelligent approaches have been introduced to improve the detection accuracy of this type of cyber-attack, including deep learning and machine learning techniques. Moreover, dimensionality reduction methods are implemented during the preprocessing stage. This research work proposes deep Autoencoder dimensionality reduction method combined with Artificial Neural Network (ANN) classifier as botnet detection system for IoT networks/systems. Experiments were carried out using 3-layer, 4-layer and 5-layer pre-processing data from the MedBIoT dataset. Experimental results show that using a 5-layer Autoencoder has better results, with details of accuracy value of 99.72%, Precision of 99.82%, Sensitivity of 99.82%, Specificity of 99.31%, and F1-score value of 99.82%. On the other hand, the 5-layer Autoencoder model succeeded in reducing the dataset size from 152 MB to 12.6 MB (equivalent to a reduction of 91.2%). Besides that, experiments on the N_BaIoT dataset also have a very high level of accuracy, up to 99.99%.

Keywords: Botnet Detection, Internet of Things, Autoencoder, Artificial Neural Network.

1. Introduction

The Internet of Things (IoT) continues to develop and is always used in everyday life, for example in smart homes, smart city, smart agriculture, industrial and so on [1]. But as IoT applications and devices rapidly grow, cyberattacks are also rising, posing more serious threats to security and privacy [2]. Distributed Denial of Service (DDOS) is one of the serious threats that exist in IoT networks, this is because it can reduce network performance [3]. Furthermore, the attack drains system resources and causes network congestion by creating large amounts of traffic in the system areas even without disrupting critical information or compromising credential files [4]. In its development, attacker can also deploy robot networks (botnets) in launching DDOS attacks [5]. Currently, the types of botnets are very diverse such as Hide and Seek, Muhstik, Linux.Mirai, Hakai, Linux.Hajime, Kenjiro, Torii, Mirai, Okiru, IRCBot, Trojan, and Gagfyt [6].

Several techniques can be used in detecting cyber-attacks including deep learning and machine learning techniques [7]. Saini et al. use a machine learning approach in detecting and classifying DDOS attacks. The proposed approach successfully classifies various types of DDOS attacks [8]. Machine learning consists of variety of methods; one of the methods is ANN [9]. The proposed ANN method shows a better detection accuracy rate with a good value of true positive rate and false positive rate in detecting the attacks [10].

Saied et al. [11] also have shown that detecting DDOS attacks using ANN gives better detection results. Detection performance is further improved by applying the dimensionality reduction method at its pre-processing stage [12]. Autoencoder is one of the most powerful methods for dimensionality reduction and has been intensively applied in data classification [13]. The Autoencoder unlike some other feature reduction techniques, where it provides interlocking range output that makes it an ideal step in data pre-processing [14]. Even though the Autoencoder is effective in detecting the type of attack, it takes time and effort to find the optimal model architecture and hyperparameter settings of the Autoencoder that produce the best detection performance [15].

The proposed work contributes towards the development of a model for botnet attacks detection system on IoT networks. The detection system is proposed by combining the Autoencoder preprocessing method and the ANN classification method.

The paper is compiled with the following composition. Section 2 reviews some of related research works that have been conducted on detection of botnet attacks on IoT networks. Section 3 describes the dataset used for the experiment, the proposed method, and the performance measurement parameters, followed by Section 4 that presents the experimental set up, results and comparison with previous research works, Lastly, Section 5 presents conclusion. At the end, a table of notations and acronyms is provided.

2. Related Work

In detecting IoT botnets process, many studies have been carried out by using various methods. Bahsi et al. [16] performed IoT botnet detection using Decision tree classification method and were compared with k-NN classification. In addition, classification performance is improved by using Fisher's score dimension reduction method. On the other hand, using a balanced dataset requires human resources during the labeling process and running real-time attack simulation is ineffective. Then Alqahtani et al [17] improved classification accuracy of XGBoost method using genetic-based extreme gradient model, on the other hand, in preprocessing data, feature selection was perform using fisher scores. Applying classification

optimization can lead to an increase in computing resources utilization and execution time. Furthermore, Alshamkhany et al. [18] performed computational efficiency process in detecting IoT botnets using PCA as dimensional reduction method and implemented four classifiers, i.e.: Decision Tree, kerned support vector machine (SVM), Naïve Bayes and K-Nearest Neighbors that makes the classification process faster. The data reduction used for each classification method varies, making it difficult to assess the level of accuracy of a more effective approach.

Pokhrel et al. [19] performed feature engineering and oversampling techniques using SMOTE method in balancing datasets from each class which were then combined with machine learning algorithms. Using k-NN classification method and Multi-layer Perception Artificial Neural Network, it is clear that the performance comparison is clearly visible. A higher value of k means more variance and less bias but causes computational overhead and requires more processing time whereas a low value means low variance and high bias. In addition, Nomm and Bahsi [20] performed discriminatory feature selection to reduce features to improve classification performance in one class SVM and isolation forest methods. Nevertheless, the experimental results yet showed low accuracy level. Then Raj et al. [21] performed classification process on IoT botnets detection using one-class support vector machine, elliptic envelope, and local outlier factor methods. Before the classification process is perform the process of introducing the network flow using input packets, protocols, source ports, destination ports, and time. The experiment only uses less than 1300 data records, so it is not able yet to represent large amounts of data traffic in the real world. **Table 1** summarizes research works on the use of pre-processing methods in intrusion detection systems using machine learning approaches for the last five (5) years.

Table 1. Comparison of the use of pre-processing methods in intrusion detection systems using machine learning

Ref. # and Year	Pre-processing Method	Machine Learning Method	Accuracy
[16], 2018	Fisher's score	Decision tree	98.43%
		k-NN	98.05%
[17], 2020	Fisher's score	XGBoost	90.6%
[18], 2020	PCA	Decision tree	93.7%
		SVM	75.99%
		k-NN	94.65%
[19], 2021	SMOTE	k-NN	99.2%
		Multi-layer Perception Artificial Neural Network	87.4%
[20], 2019	Entropy	SVM	93.15%
		Isolation forest	83.85%
	Variance	SVM	64.03%
		Isolation forest	41.91%
	Hopkins	SVM	79.01%
		Isolation forest	57.01%
[21], 2021	Network flow-based	One-class support vector machine	99%

3. Proposed Methodology

3.1 System Workflow

The proposed system combines the ANN classification method and the Autoencoder as data dimension reduction method. The experiment follows the workflow presented in **Fig. 1**. The experiment process begins with the testing on data reduction using 3-layer, 4-layer, and 5-layer. Having done an evaluation, the selection of the best results is carried out. Then the selected model is validated on the training data validation and the testing data. In addition, validation using Balance Accuracy (BACC) and Matthew's Correlation Coefficient (MCC) are also performed.

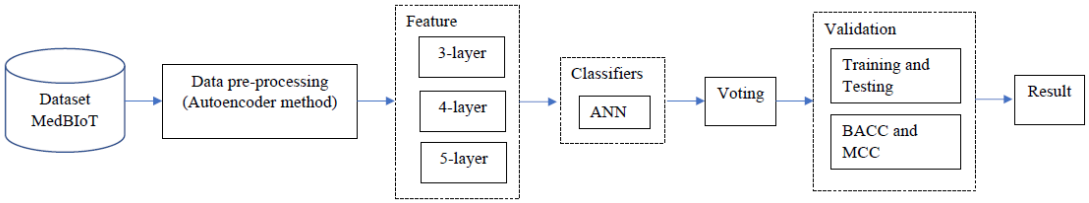

Fig. 1. Proposed workflow of the experiment

3.2 Dataset

For the experiments purpose, the Medium Botnet IoT (MedBIoT) dataset in the format of pcap file [22] is used. This dataset is generated from collection of Physical and Virtual networks that consists of 80 devices. The MedBIoT dataset contains four types of traffic data, i.e.: Normal, Bashlite, Mirai and Torii traffic data. Nevertheless, experiments in this research work only consider, Normal, Mirai and Torii botnet traffic data. Then Mirai and Torii traffic data are combined into one attack traffic. The list of data files used for the experiments is presented in **Table 1**.

Table 1. Distribution of data used

Data label	Traffic Data	Device	Number of files
Attack	Mirai	Fan	3
		Light	3
		Switch	3
	Torii	Fan	2
		Lock	3
		Rasberry	2
Normal	Normal	Rasberry	2
		Light	2
		Lock	2
		Switch	2
Total			24

The N-BaIoT dataset [23], which has data traffic types of Benign, Mirai, and Bahslite is used for validating the selected model. This dataset has a total of 7062606 records, however, only 20% of the records are considered in the experiments. The distribution of the N-BaIoT dataset for the experiments is presented in **Table 2**.

Table 2. Distribution of 20% data N-BaIoT

New Label	Label File	Number of data
Benign	Benign	111179
Bashlite	Combo	103030
	Junk	52158
	Scan	51022
	TCP	171969
	UDP	192873
Mirai	Ack	128764
	Scan	107596
	Syn	146660
	UDP	246001
	UDPplain	104660
Total		1415912

3.3 Feature Extraction

In the feature extraction stage, this research work uses CICFlowMeter tool [24]–[26]. This tool is developed using Java programming language. This feature extraction produces 83 features in .csv file format. The result of the feature extraction process is presented in **Table 3**.

Table 3. Extraction feature result

Extracted Attributes	
Src IP	Idle Time (Min, Mean, Max, Std)
Dst IP	Active Time (Min, Mean, Max, Std)
Src Port	Flow Packet Length (Min,Mean,Max,Std)
Dst Port	Flow Forward Bytes
Flow Duration	Flow Backward Bytes
Total Forward Bytes	Flow (Fin,Syn,Rst, Psh, Ack, Urg, Cwr,Ece)
Total Backward Bytes	Initial Window Forward
Forward Header Length	Initial Window Backward
Backward Header Length	Segment Size Forward (Max, Min)
Total Forward Packets	Segment Size Backward (Max, Min)
Total Backward Packets	Forward Arrival Time (Min,Mean,Max,Std)
Forward Packet Length (Min,Mean,Max,Std)	Backward Arrival Time (Min,Mean,Max,Std)
Backward Packet Length (Min, Mean, Max, Std)	

3.4 Autoencoder

After the dataset successfully passed through the feature extraction process, the data is fed to the data dimensionality reduction process. The data dimensionality reduction process is useful to decrease the size of dataset. This research work uses deep Autoencoder algorithm as dimensionality reduction method. The formulae of the Autoencoder are expressed in (1) and (2) [27].

$$Y = f(X) = s(WX + bx) \quad (1)$$

$$X' = g(Y) = s(W'Y + by) \quad (2)$$

Where,

Y = $f(x)$ = Encoded Function
X' = $g(y)$ = Encoded Function
S = Activation function
by = Bias decoded
W = Weighted encoded
bx = Bias encoded
W' = Weighted decoded

The Autoencoder architecture and parameters are presented in **Table 4**.

Table 4. Parameters of the Autoencoder

Parameter	Value
Node Input layer	74
Node Ouput layer	74
Node on Hidden layer 1	50
Node on Hidden layer 2	30
Node on Hidden layer 3	20
Node on Hidden layer 4	10
Node on Hidden layer 5	5
Activation function of hidden layer	Relu
Activation function of Ouput layer	Sigmoid and Softmax
Learning rate	0.001
Loss Function	Binary Crossentropy
Optimization Function	Adam

The data dimensional reduction process using the Autoencoder is shown in **Fig. 2**.

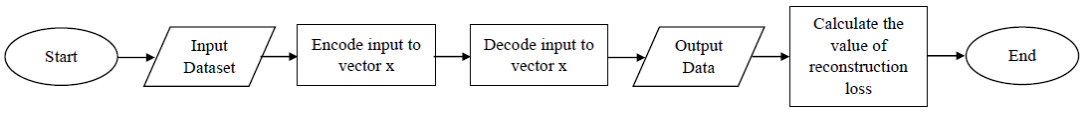

Fig. 2. Flowchart for the Autoencoder dimensional reduction

3.5 Classification

The ANN is machine learning algorithm that can be used for prediction and classification purposes [28]. The ANN is designed to simulate the working of human brain in processing of incoming information. In pattern recognition process, the ANN gathers their knowledge from the experiments on determined data training [29]. In this research work, the ANN architecture consists of 2 Hidden layers, where the numbers of neurons in each layer are 50 neurons, while Input layer has 5 neurons. The ANN uses Sigmoid activation function and Softmax as the loss function. **Fig. 3**. illustrates the architecture of the proposed ANN.

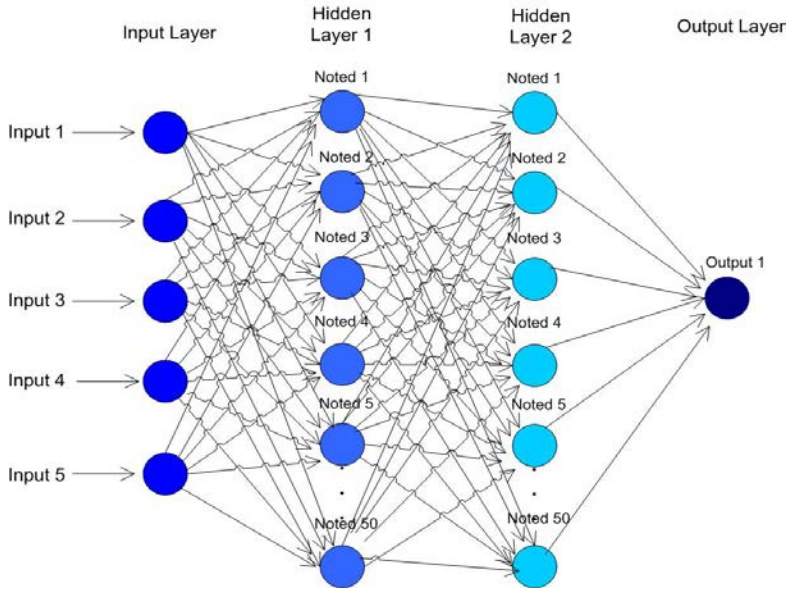

Fig. 3. ANN architecture

3.6 Validation

Validation stage is needed to justify whether the proposed classification engine is as planned. The validation stage in this study consists of 4 stages, as follows.

3.6.1 Confusion Matrix

The first stage of the validation is performed using the values obtained from the confusion matrix [30]. In binary classification system, labels are divided into two classes, so that the confusion matrix formed follows the number of classes. The confusion binary matrix is displayed in **Fig. 4**.

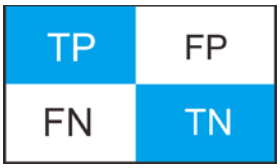

Fig. 4. Binary confusion matrix

The performance metrics of accuracy, precision, sensitivity, specificity, and F1-score are considered, as represented by (3) – (7), where, TP= True Positive, TN= True Negative, FP= False Positive, and FN= False Negative.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} x\ 100\% \quad (3)$$

$$Precision = \frac{TP}{TP+FP} x\ 100\% \quad (4)$$

$$Sensitivity = \frac{TP}{TP+FP} x100\% \quad (5)$$

$$Specificity = \frac{TP}{TP+FN} x100\% \quad (6)$$

$$F1\ Score\ = \frac{2x\ Precission\ x\ Sensitivity}{Precission+Sensitivity} x100\% \quad (7)$$

3.6.2 Autoencoder Validation

Validation of the use of the number of autoencoder layers is presented in **Table 5**.

Table 5. Validation of total of layer number

Number of layers	Learning rate
3 hidden layer	0.00001
4 hidden layer	0.0001
5 hidden layer	0.001

3.6.3 Data Testing and Data Training Separation

Separation of training and testing data is performed for five scenarios as presented in **Table 6**.

Table 6. Validation of total data separation

Validation	Data
First Validation	Training Data 50% and Testing Data 50 %
Second Validation	Training Data 60% and Testing Data 40 %
Third Validation	Training Data 70% and Testing Data 30 %
Fourth Validation	Training Data 80% and Testing Data 20 %
Fifth Validation	Training Data 90% and Testing Data 10 %

3.6.4 BACC Validation

Performance level measurement of classification process is carried out using Balance Accuracy (BACC) and Matthew's Correlation Coefficient (MCC) parameters [30], [31]. The BACC parameter is used to measure the imbalanced data accuracy, while the MCC parameter is useful for measuring the sensitivity level of the imbalanced data. The description of binary BACC parameters are represented by (8), and binary MCC parameters are represented by (9).

$$BACC\ = \frac{\frac{TP}{P}+\frac{TN}{N}}{2} \quad (8)$$

$$MCC\ = \frac{(TP\ x\ TN)-(FP\ x\ FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FN)(TN+FP)}} \quad (9)$$

4. Experimental Setup, Result, and Discussion

4.1 Experimental Setup

Hardware and software configuration for the experiments are provided in **Table 7** and **Table 8**.

Table 7. Hardware configuration

No	Device	Specification
1	Processor	Intel Core i5 4th Gen.
2	RAM	8GB DDR3L
3	Storage	256GB SSD
4	VGA	Intel HD 4000
5	Operating system	Linux

Table 8. Software Configuration

No	Method	Application
1	Dimensionality reduction Autoencoder	Autoencoder function in Python
2	ANN	Neural network Sklearn Library in Python

4.2 Experimental Setup

4.2.1 The Autoencoder Dimensionality Reduction

The experiments are carried out for the epoch value of 100, the batch size of 64, the learning rate value of 0.001 and the loss function was binary_crossentropy. For optimization purpose the Adam optimizer is implemented. **Fig. 5 – Fig. 7** show the loss of the models from the scenarios mentioned in **Table 5** and **Table 6**.

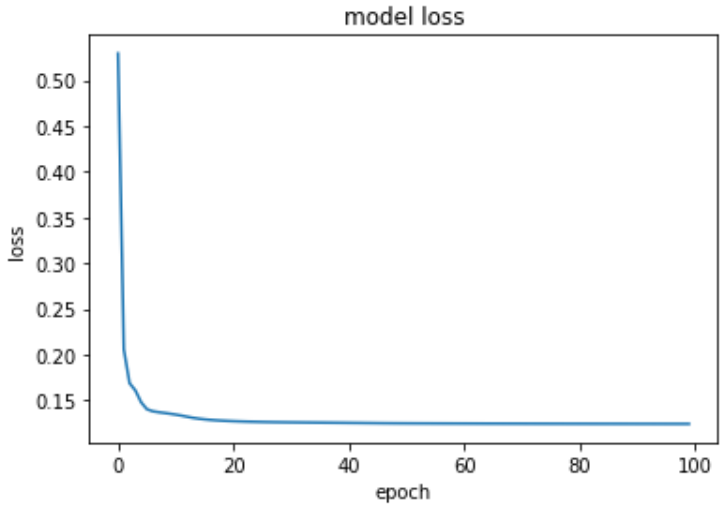

Fig. 5. Loss of the 3-layer *Autoencoder* model

Fig. 5 shows the loss of data dimensionality reduction process using 3-layer Autoencoder model. The training process was executed for ± 953 seconds duration and obtaining loss value of 12.46 after 100 epochs. The dimensionality reduction process decreases the datasets dimension from the originally 74 columns to 5 columns as displayed in **Table 9**. Feature_0 has a result of 0, because it corresponds to the Autoencoder rule which forces the Hidden layer to activate only a few hidden units per data sample. With activation, if the value of the j-th hidden unit is close to 1 it will be activated, otherwise it will be deactivated. The output from disabled nodes to the next layer is zero. The initial dataset size was 152 MB, and then after gone through dimensionality reduction process reduces to 10.5 MB.

Table 9. Dimensionality reduction result by the 3-layer Autoencoder

Layer	Feature_0	Feature_1	Feature_2	Feature_3	Feature_4
0	0.0	31.701628	29.087816	0.000000	4.778793
1	0.0	32.227139	28.878605	0.011039	5.134368
2	0.0	32.886936	28.604403	0.222657	5.585252
3	0.0	31.395220	29.214219	0.000000	4.580541
4	0.0	31.232479	29.298254	0.000000	4.507546

Next experimental result generates a plot of loss for the *Autoencoder* model with 4-layer as displayed in **Fig. 6**.

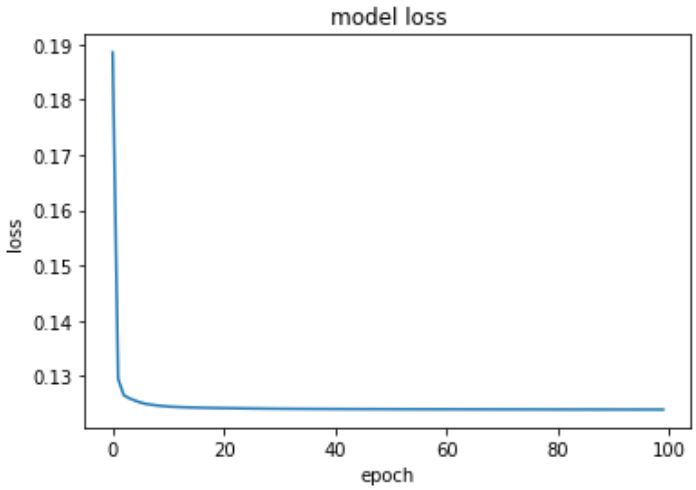

Fig. 6. Loss of the 4-layer *Autoencoder*

Fig. 6 shows the loss of data dimensionality reduction process using 4-layer Autoencoder model. The training process was executed for ± 1005 seconds duration and obtaining loss value of 12.39 after 100 epochs. The dimensionality reduction process decreases the datasets dimension from the originally 74 columns to 5 columns as displayed in **Table 10**. The initial dataset size was 152 MB, and then after gone through dimensionality reduction process reduces to 12.7 MB.

Table 10. Dimensionality reduction result by the 4-layer Autoencoder

Layer	Feature_0	Feature_1	Feature_2	Feature_3	Feature_4
0	27.785494	28.817701	37.764332	6.023538	24.255562
1	25.205114	27.489313	33.871586	5.866804	22.742176
2	21.386585	18.537936	20.131233	11.004463	23.40733
3	11.885992	6.257492	18.669674	0.000000	15.868389
4	16.455261	14.538950	14.790109	8.503696	22.956312

Next experimental result generates a plot of the loss for the *Autoencoder* model with 5-layer as displayed in **Fig. 7**. The figure shows the loss of data dimensionality reduction process using 5-layer Autoencoder model. The training process was executed for ± 1125 seconds duration and obtaining loss value of 12.38 after 100 epochs. The dimensionality reduction process decreases the datasets dimension from the originally 74 columns to 5 columns as displayed in **Table 11**. Feature_1 has a result of 0 this is because it corresponds to the Autoencoder rule, which forces the Hidden layer to activate only a few hidden units per data sample. With activation, if the value of the j-th hidden unit is close to 1 it will be activated, otherwise it will be deactivated. The output from disabled nodes to the next layer is zero. The initial dataset size was 152 MB, and then after gone through dimensionality reduction process reduces to 12.6 MB.

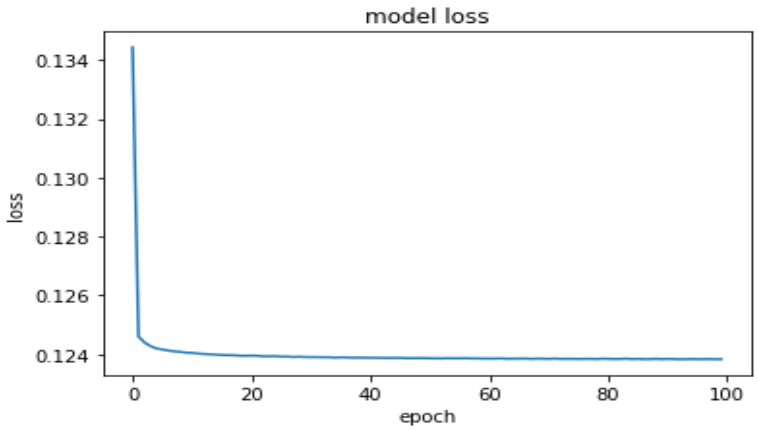

Fig. 7. Loss of the 5-layer *Autoencoder*

Table 11. Dimensionality reduction result by the 5-layer Autoencoder

Layer	Feature_0	Feature_1	Feature_2	Feature_3	Feature_4
0	3.296371	0.0	1.468847	1.399031	0.520432
1	3.330577	0.0	1.434509	1.572026	0.519765
2	3.412631	0.0	1.372878	1.945709	0.517105
3	3.282127	0.0	1.485179	1.313242	0.521467
4	3.273893	0.0	1.493417	1.265805	0.522149

The next experiment was to carry out the classification process using the data separation of 70% training data and 30% testing data. **Table 12** presents the overall experimental results on the confusion matrix using the Autoencoder. It is observed that the 5-layer Autoencoder model provided the lowest False Positive while the 4-layer Autoencoder model provided the lowest False Negative values.

Table 12. Confusion matrix of classification result comparison

Autoencoder model	TP	FP	FN	TN
3-layer	58644	206	194	15092
4-layer	58656	194	87	15199
5-layer	58671	59	194	15152

Based on the obtained confusion matrix, the level of accuracy for 3-layer, 4-layer and 5-layer Autoencoder models is computed and presented in **Table 13**. The experimental results show that the use of 5 layers has a higher level of accuracy.

Table 13. Comparison of validation results of autoencoder

Autoencoder	Accuracy of Training	Testing				
		Accuracy	Specificity	Sensitivity	F1-Score	Precision
3-layer	99.49	99.46	98.73	99.64	99.66	99.67
4-layer	99.63	99.62	99.43	99.67	99.76	99.85
5-layer	99.64	99.65	98.73	99.89	99.67	99.78

In addition, we analyze the computational overhead and complexity at each layer as shown in **Fig. 8**. The figure shows that the use of 3-layer Autoencoder model takes a little time, while the use of 4-layer Autoencoder model has a higher overhead.

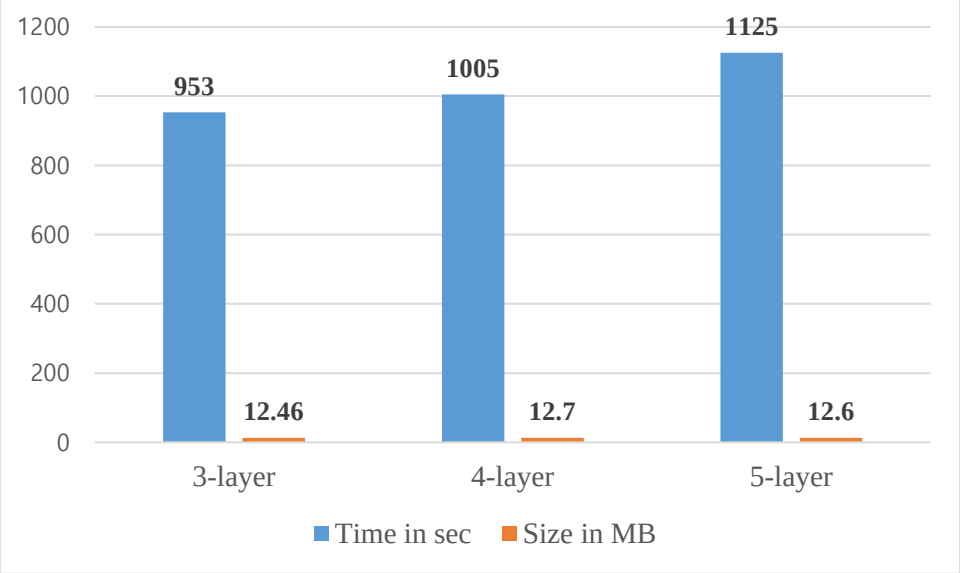

Fig. 8. Computational overhead and complexity

4.2.2 Validation Results of Data Testing and Data Training on 5-layer

We performed binary classification, to determine whether the traffic data in the dataset is attack or normal traffic. We start with the experiment to determine the Confusion matrix. For this purpose, we recorded the binary classification results, then presented them in the form of Confusion matrix. **Table 14** presents the overall experimental results on Confusion matrix using different scenario of data separations. It is observed that the data separation scenario of 90:10 provided the lowest False Positive and False Negative values.

Table 14. Confusion matrix of classification result comparison

Data Ratio	TP	FP	FN	TN
50:50	97701	382	163	25313
60:40	78307	160	172	20209
70:30	58671	179	134	15152
80:20	39166	68	108	10082
90:10	19585	**32**	**35**	5060

Then for each classification result on each data separation scenario, we evaluated further using Accuracy, Loss, and Precision-Recall metrics. First, we consider the experiment with 50:50 data separation. **Fig. 9** and **Fig. 10** show the Accuracy and Loss comparison during the training and testing phases, respectively.

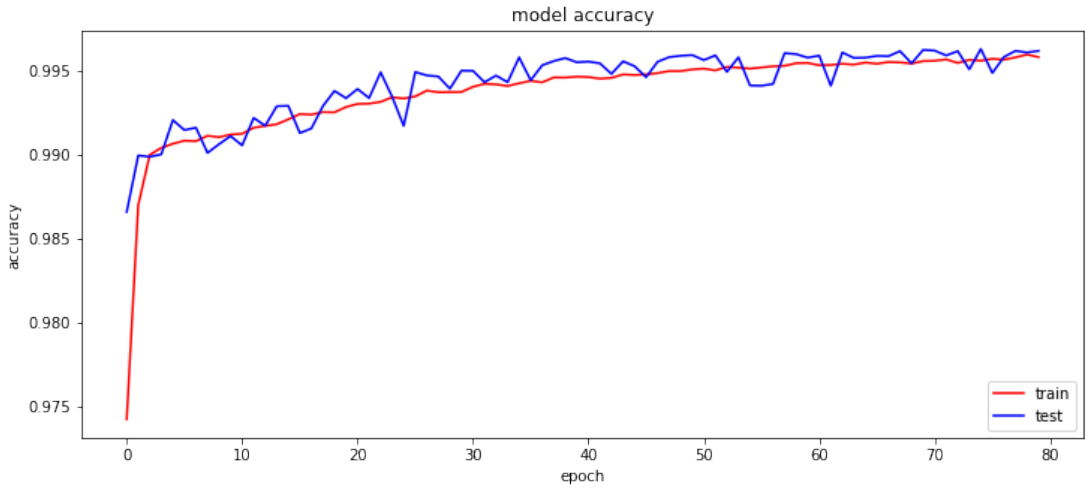

Fig. 9. Accuracy of classification for 50% training data and 50% testing data separation

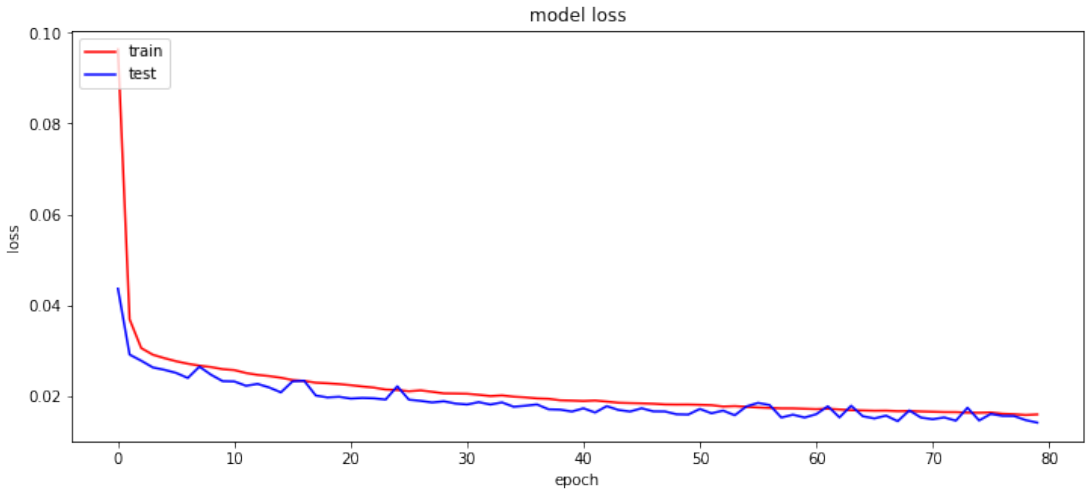

Fig. 10. Loss of classification for 50% training data and 50% testing data separation

It can be seen that the accuracy value and the loss value are fluctuate; indicating a non-convergent graph, thus, the performance of classification model is still not good enough. Furthermore, to describe more detail on the classification performance, the ROC curve and the Precision-Recall curve are calculated and presented in **Fig. 11** and **Fig. 12**, respectively.

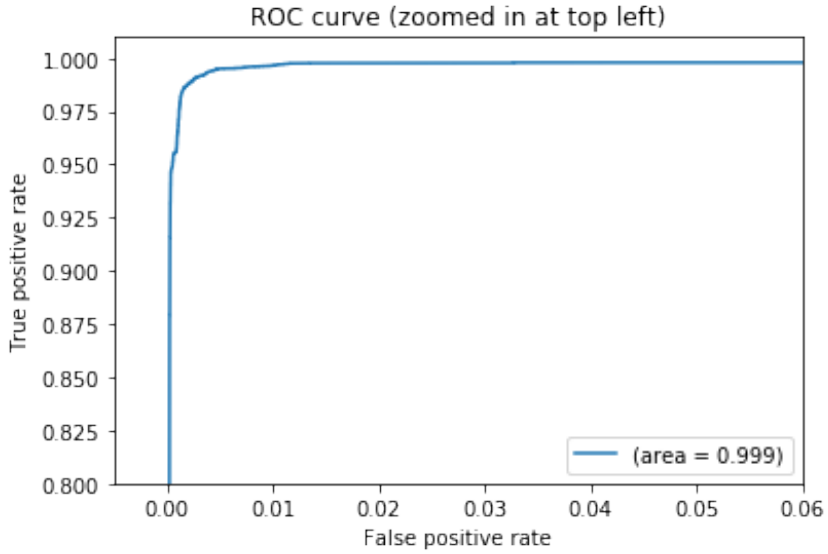

Fig. 11. ROC for 50% training data and 50% testing data separation

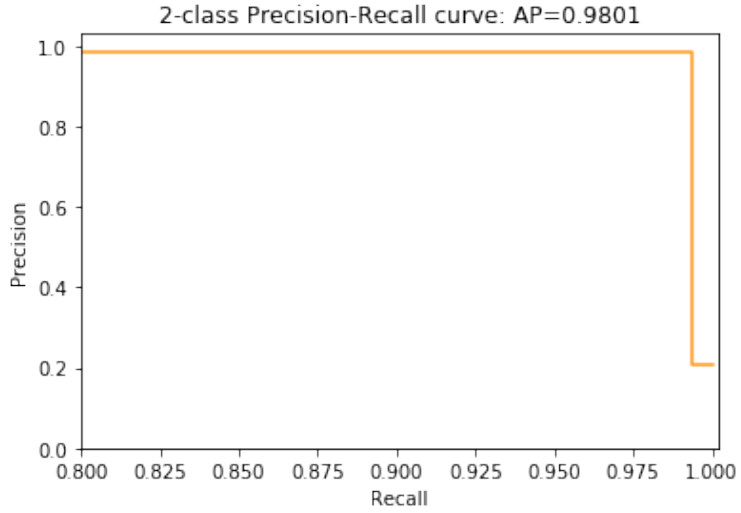

Fig. 12. Precision-Recall for 50% training data and 50% testing data separation

From the ROC curve in **Fig. 11** it can be seen that the true positive rate is 0.9983, while the false positive rate is 0.0149 and from the Precision-Recall curve in **Fig. 12**, it can be seen that the Precision value is 0.9961, while the Recall value is 0.9983, this value is obtained from the values parameter contained in confusion matrix, i.e.: TP value= 97701, FP value= 382, FN value =163 and TN value= 25313. The FN value and FP value are still high, so the model is still not good for the classifying process.

Next, we consider the experiment with 60:40 data separation. **Fig. 13** and **Fig. 14** show the Accuracy and Loss comparison during the training and testing phase.

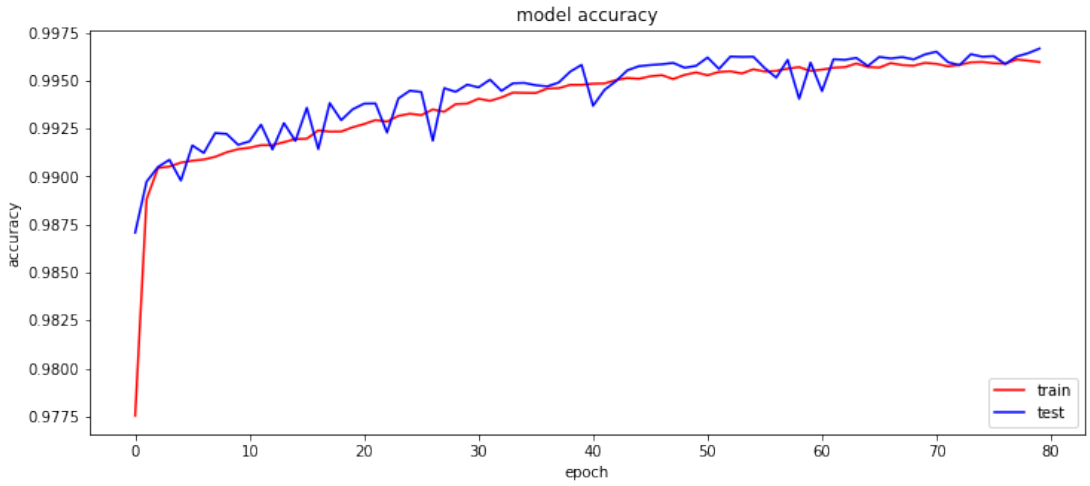

Fig. 13. Accuracy of classification with 60% training data and 40% testing data

In **Fig. 13**, it is shown that the resulting accuracy yet experiences drastic fluctuated in the 28th and the 40th epochs, and then begins to stabilize on the 60th epoch. The figure shows a graph that does not converge, so the performance of the resulting classification model is still not good. Similarly in **Fig. 14**, it can be seen that the loss value is fluctuated as well. The two figures still show non-convergent graph, thus the performance of the resulting classification model is still not good. Furthermore, to describe more detail on the classification performance, the ROC curve and the Precision-Recall curve are calculated and presented in **Fig. 15** and **Fig. 16**, respectively.

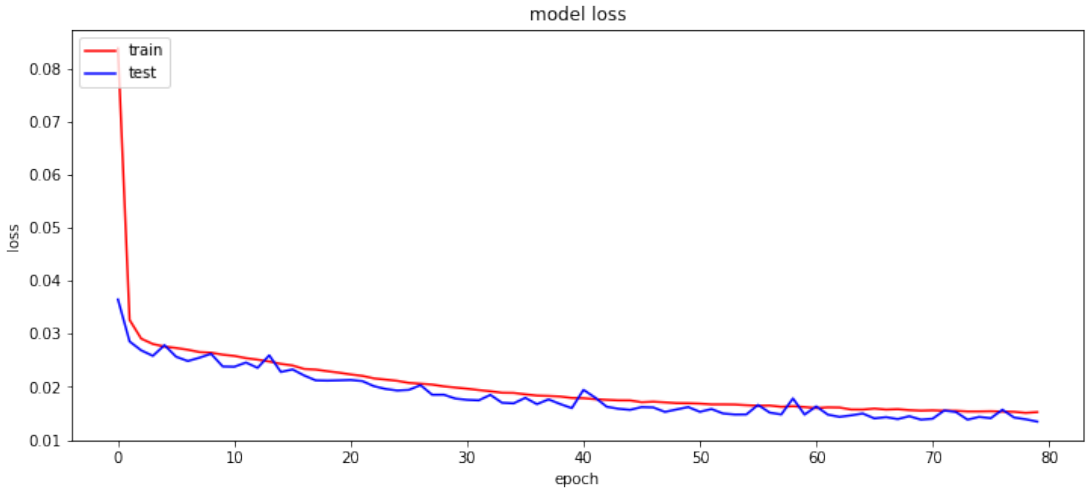

Fig. 14. Loss of classification with 60% training data and 40% testing data

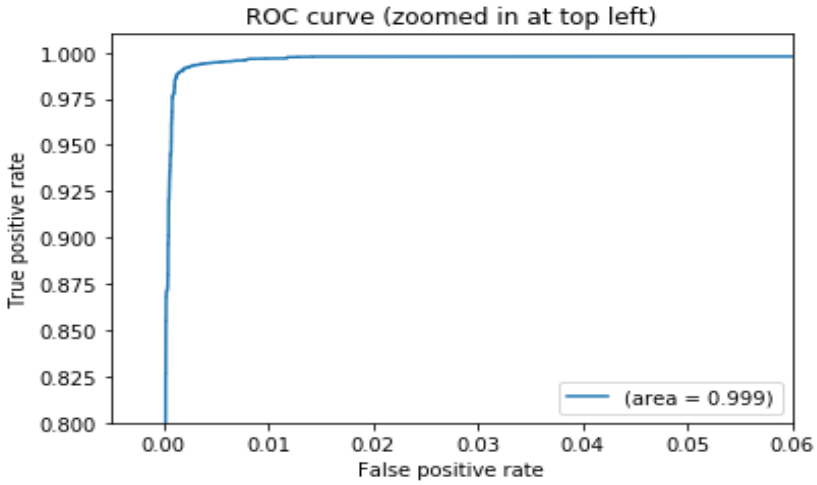

Fig. 15. ROC of classification with 60% training data and 40% testing data

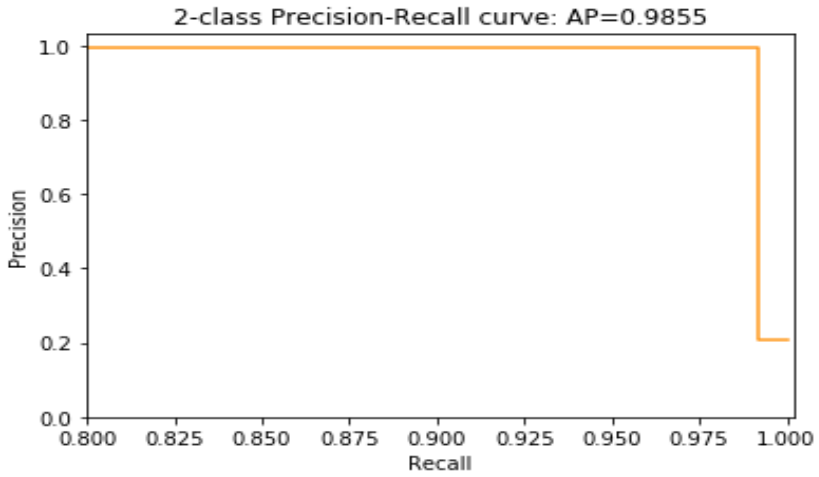

Fig. 16. Precision-Recall of classification with 60% training data and 40% testing data

ROC curve in **Fig. 15** and the Precision-Recall curve in **Fig. 16** describe the parameter values contained in matrix confusion, where TP value is 78307, FP value is 160, FN value is 172, TN value is 20209. The FN value has increased and the resulting FP value has actually decreased from the previous experiment results.

Next, the validation results for experiment with 70% training data and 30% testing data separation are presented. **Fig. 17** and **Fig. 18** show the Accuracy and Loss comparison during the training and testing phases. It can be seen from **Fig. 17** that the resulting loss is still fluctuated, where the significant increment in Loss occurs at the 20th epoch. While in **Fig. 18**, it is shown that the Accuracy is also still fluctuated, where significant decline and upswing occurred in the 22nd and 32nd epochs. From the two graphs, it is shown that the graphs are non-convergent, so the performance of the resulting classification model is still not good.

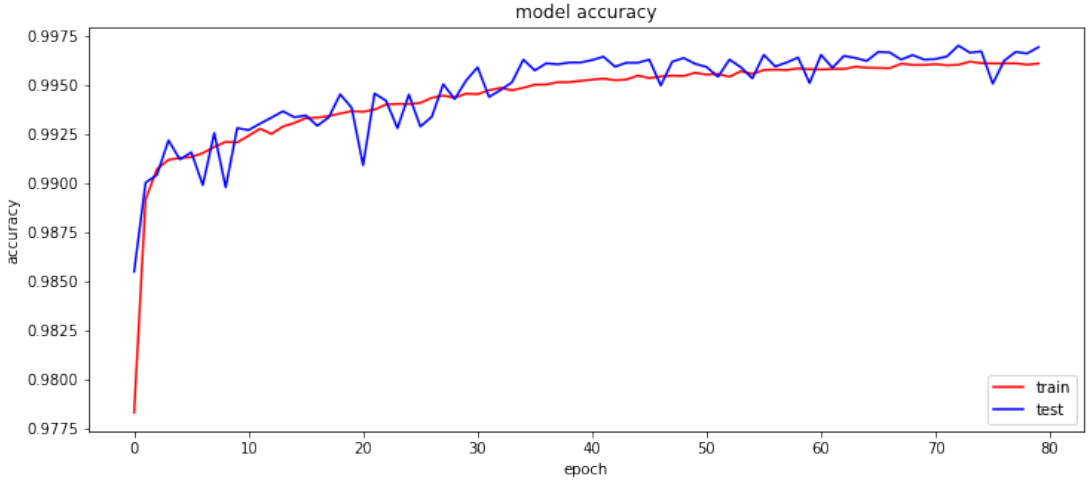

Fig. 17. Accuracy of classification with 70% training data and 30% testing data

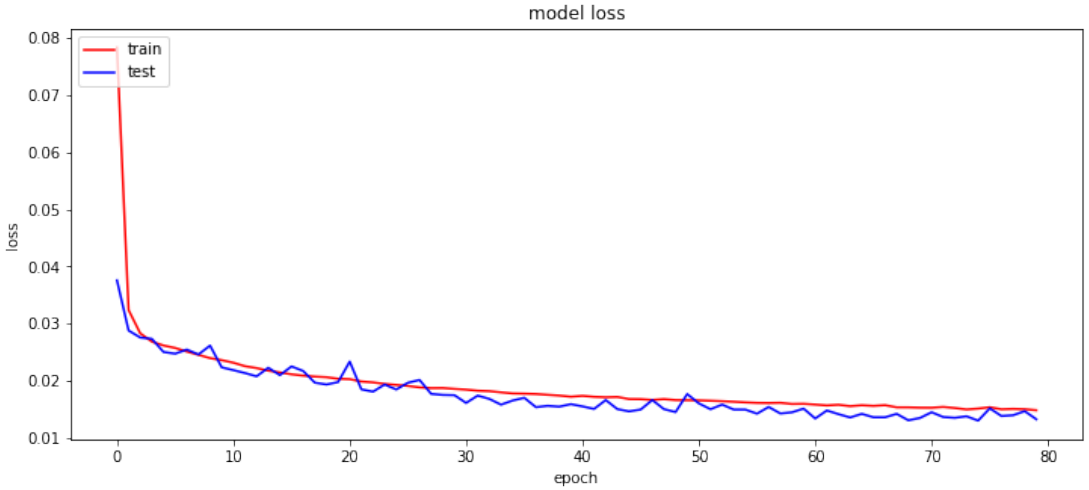

Fig. 18. Loss of classification with 60% training data and 40% testing data

Furthermore, to describe more detail on the classification performance, the ROC curve and the Precision-Recall curve are calculated and shown in **Fig. 19** and **Fig. 20**, respectively. ROC curve in **Fig. 19** and the Precision-Recall curve in the **Fig. 20** describe the parameter values contained in the matrix confusion, where TP value is 58791, the FP value is 59, the FN value is 194, and the TN value is 15092. Based on the confusion matrix, the value of FP has decreased from the previous experiment results. However, the resulted FN and FP values are considered still understated.

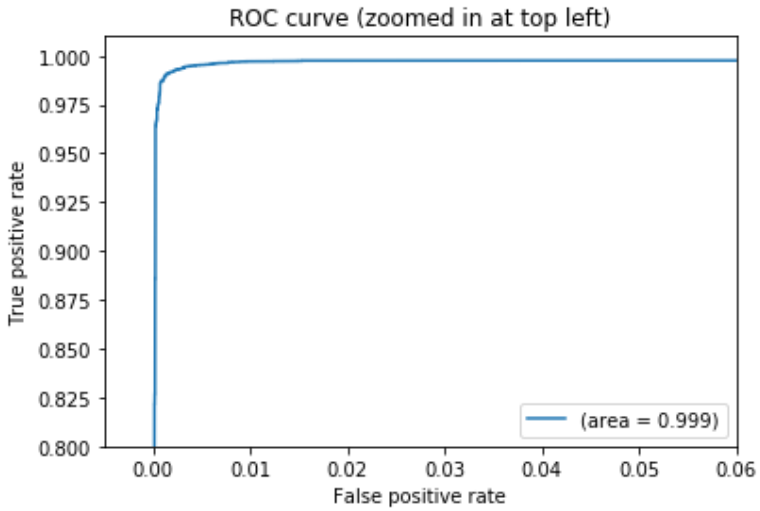

Fig. 19. ROC of classification with 70% training data and 30% testing data

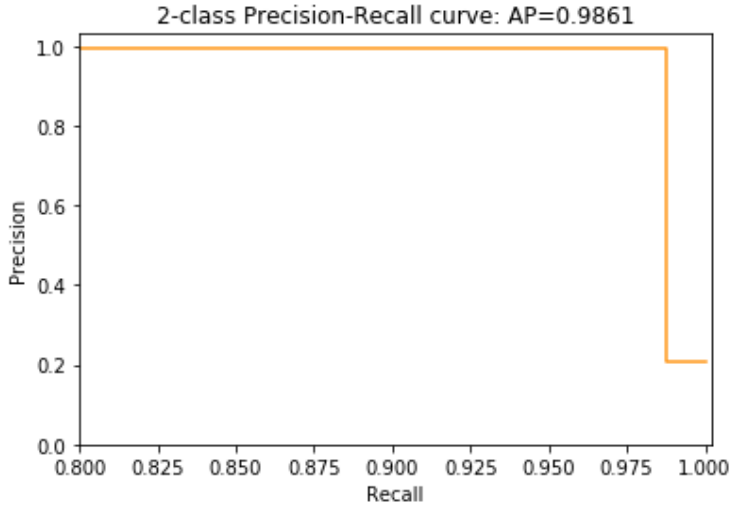

Fig. 20. Precision-Recall of classification with 70% training data and 30% testing data

Next is the validation result for experiment with 80% training data and 20% testing data. **Fig. 21** and **Fig. 22** show the Accuracy and Loss comparison during the training and testing phases. It can be seen in **Fig. 21**, the resulted loss is fluctuated, where the significant increment in the Loss occurred at the 59th epoch. **Fig. 22** shows that the resulted Accuracy still shows fluctuated result, where the significant decline and increment occurred in the 49th and 48th epochs. The two figures exhibit a non-convergent graph, so the performance of the resulted classification model is still not good. Furthermore, to describe more detail on the classification performance, the ROC curve and the Precision-Recall curve are calculated and shown in **Fig. 23** and **Fig. 24**, respectively.

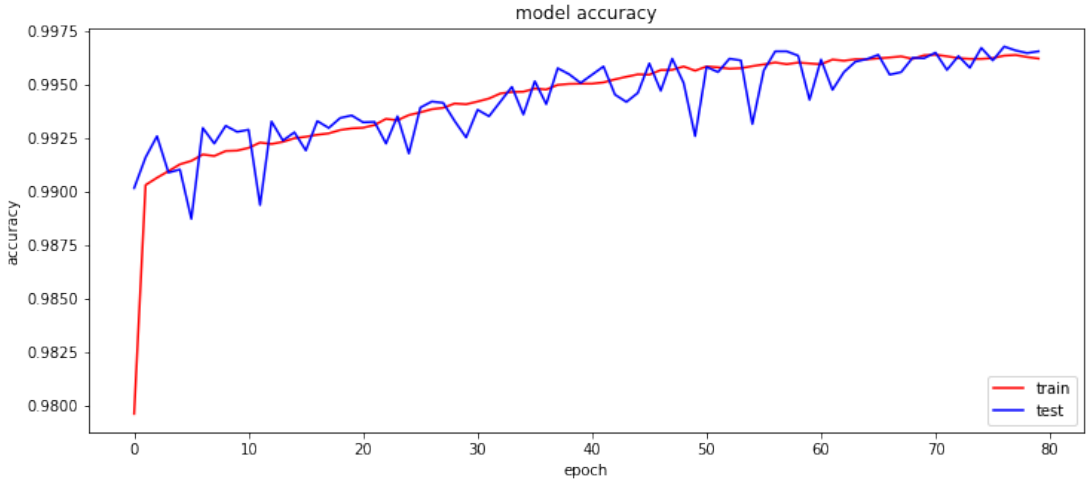

Fig. 21. Accuracy of classification with 80% training data and 20% testing data

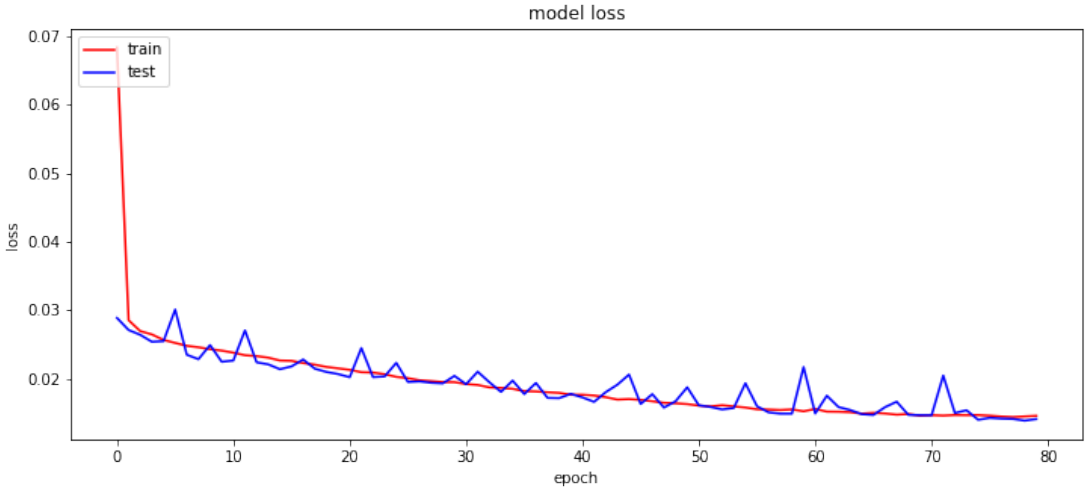

Fig. 22. Loss of classification with 80% training data and 20% testing data

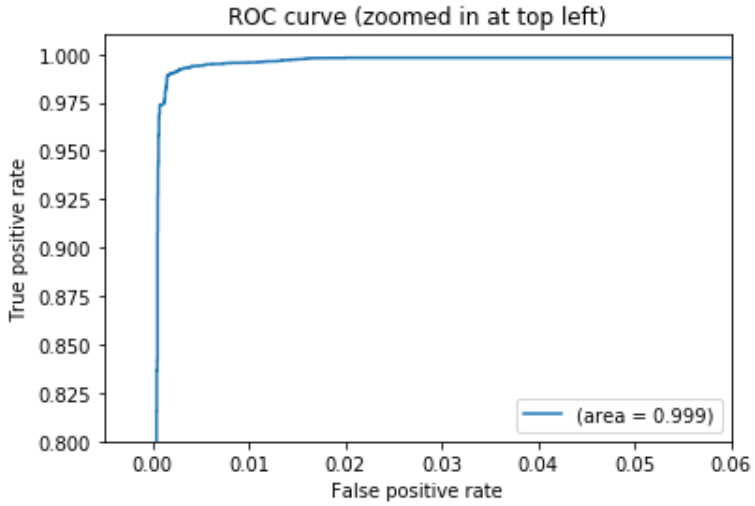

Fig. 23. ROC of classification with 80% training data and 20% testing data

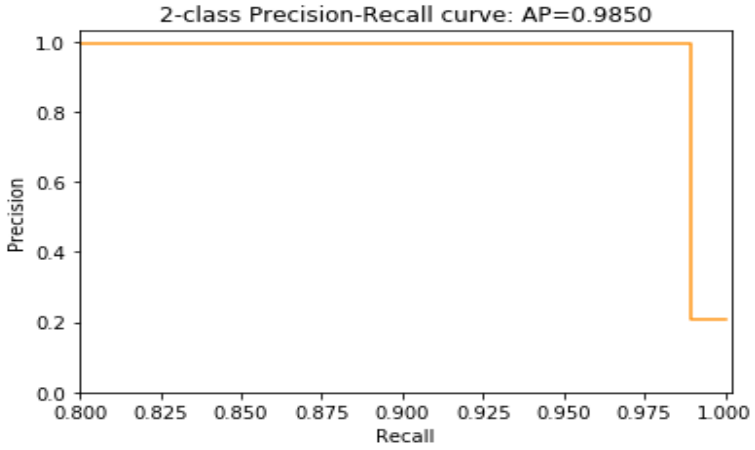

Fig. 24. Precision-Recall of classification with 80% training data and 20% testing data

ROC curve in **Fig. 23** and the Precision-Recall curve in the **Fig. 24** describe the parameter values contained in the Confusion matrix, where the TP value is 39166, the FP value is 68, the FN value is 108 and the TN value is 10082. The FN value has decreased from the previous experiment. However, the resulting FN and FP values are considered still be understated.

Lastly, the validation results of the experiment with 90% training data and 10% testing data are presented. **Fig. 25** and **Fig. 26** show the Accuracy and Loss comparison during the training and testing phase with 90% training data and 10% testing data. It can be seen in **Fig 25** that the Loss experiences significant increase in the 69th epoch. While in **Fig. 26**, the Accuracy again decreases in the 42nd and 70th epoch. From the two figures, it still shows non-convergent graph, so the performance of the resulting classification model is still not good.

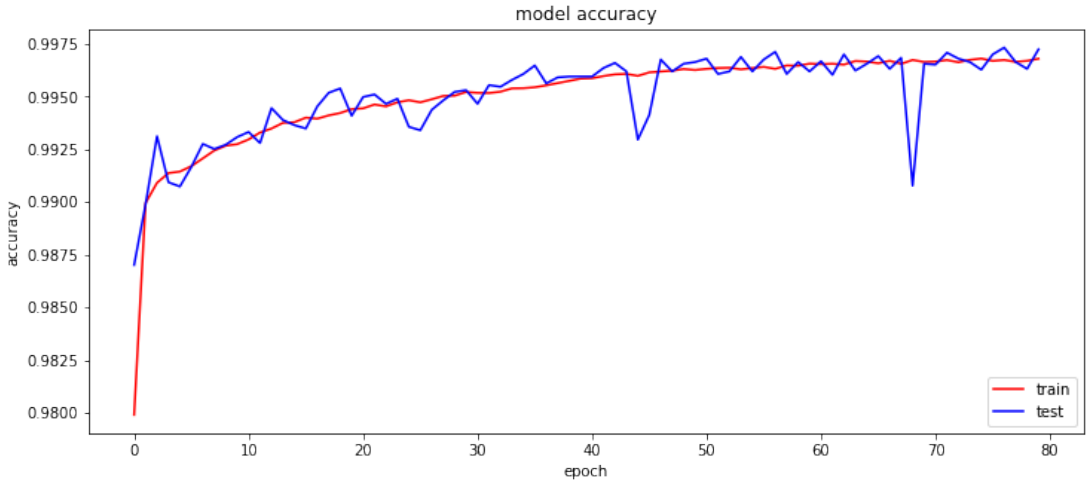

Fig. 25. Accuracy of classification with 90% training data and 10% testing data

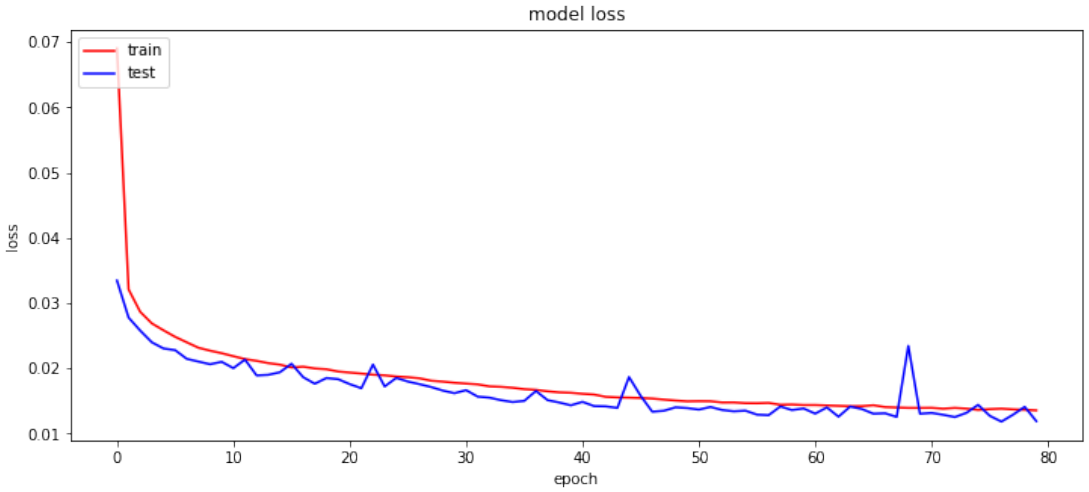

Fig. 26. Loss of classification with 90% training data and 10% testing data

To describe further details on the classification performance, the ROC curve and the Precision-Recall curve are shown in **Fig. 27** and **Fig. 28**, respectively.

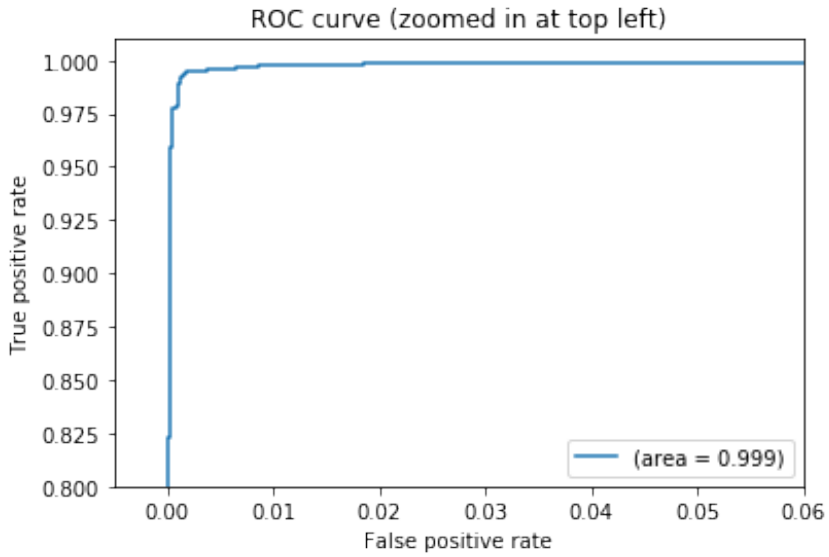

Fig. 27. ROC of classification testing with 90% training data and 10% testing data

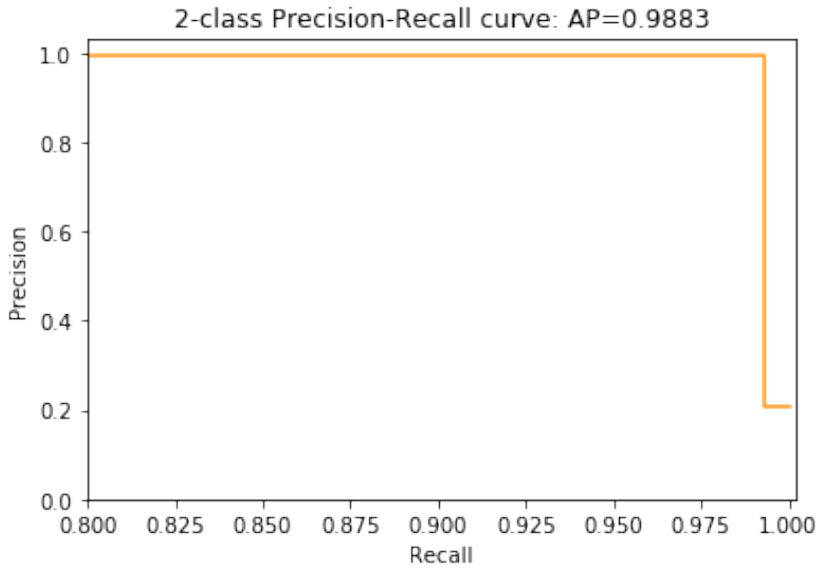

Fig. 28. Precision-Recall of classification testing with 90% training data and 10% testing data

ROC curve in **Fig. 27** and Precision-Recall curve in **Fig. 28** describe the parameter values contained in the Confusion matrix, where the TP value is 19585, FP value is 32, FN value is 35, and TN value is 5060. It is observed that the values of FN and FP have decreased significantly from previous experiments. The use of 90% training data and 10% testing scenario achieved the best performance in term of Confusion matrix parameters.

Based on observation results presented in **Table 15**. Accuracy, Precision, Sensitivity, Specificity Value and F-Score values are calculated and presented in in **Table 16**. The validation result of experiment with 50% training data and 50% testing data provided accuracy value of 99.50% in the training phase. While the validation result during the testing phase provided Accuracy value of 99.55%, Specificity value of 99.36%, Sensitivity value of 99.61%, F1-score value 99.72%, and Precision value 99.83%. The Accuracy value is relatively good,

however there is still a room for further improvement. These results are considered still not good enough for classification. Then, the validation results of experiment with 60% training data and 40% testing data provided Accuracy value 99.65% during the training phase. As for the testing phase, it generated Accuracy value of 99.66%, Specificity value of 99.15%, Sensitivity value of 99.79%, F1-score value of 99.78%, and Precision value of 99.78%. The Accuracy value has increased if compared to the previous experiment. However, the Accuracy value is yet considered to be improved, so the results are still not good in classifying.

Next, the validation results of experiment with training data 70% and testing data 30% provided Accuracy value of 99.64% during the training phase. As for the testing phase, it produced Accuracy value of 99.65%, Specificity value of 98.73%, Sensitivity value of 99.89%, F1-score value of 99.67%, and Precision value of 99.78%. The regained Accuracy value has increased when compared to the previous accuracy value. However, the accuracy value is still considered able to be improved, so the results are still not good in classifying.

The validation results of experiment with 80% training data and 20% testing data provided Accuracy value of 99.65% during the training phase. As for the testing phase, it generated Accuracy value of 99.64%, Specificity value of 99.94%, Sensitivity value of 99.82%, F1-score value of 99.77% and Precision value of 99.72%. The regained Accuracy value has increased from the previous experiment. However, the accuracy value is considered able to be improved, so the results are still not good in classifying. Finally, validation results of experiment with 90% training data and 10% testing data resulted in Accuracy value of 99.71% during training phase. As for the testing phase, it generated Accuracy value of 99.72%, Specificity value of 99.31%, Sensitivity value of 99.83%, F1-score value of 99.82% and Precision value of 99.82%. The regained accuracy value has increased from the previous experiment. From overall experiment results, this 90% training data and the 10% testing data separation scenario achieved the best performance.

Table 15. Comparison of validation results of different training data and testing data separation

Data Ratio	Accuracy of Training	Testing				
		Accuracy	Specificity	Sensitivity	F1-Score	Precision
50:50	99.50%	99.55%	99.36%	99.61%	99.72%	99.83%
60:40	99.65%	99.65%	99.15%	99.79%	99.78%	99.78%
70:30	99.64%	99.65%	98.73%	99.89%	99.67%	99.78%
80:20	99.65%	99.64%	98.94%	99.82%	99.77%	99.72%
90:10	99.71%	**99.72%**	**99.31%**	**99.83%**	**99.82%**	**99.82%**

The validation results of BACC and MCC also showed that 5-layer Autoencoder model with 90% training data and 10% testing data separation scenario achieved the best performance as displayed in **Table 16**.

Table 16. BACC and MCC validation result

Data Separation Scenario	BACC	MCC
50:50	99.48%	98.65%
60:40	99.47%	98.97%
70:30	99.40%	98.71%
80:20	99.38%	98.91%
90:10	**99.57%**	**99.17%**

With the aim to measure the robustness of the selected model, it was experimented using different dataset that has more records of data, in this case using the N_BaIoT dataset. This data has a number of different classes. The MedBIoT dataset used in the initial test has the Normal and the Attack labels, while the N_BaIoT dataset has Benign, Mirai, and Bahslite labels. Comparison of the selected model performance on the two datasets is presented in **Table 17**. The results show that the selected model has very good performance on the N_BaIoT dataset with an accuracy rate of up to 99.99%.

Table 17. Comparison of validation results on the MedBIoT and N_BaIoT datasets

Dataset	Data Ratio	Accuracy of Training	Testing				
			Accuracy	Specificity	Sensitivity	F1-Score	Precision
MedBIoT	90:10	99.71%	99.72%	99.31%	99.83%	99.82%	99.82%
N_BaIoT	90:10	99.99%	99.99%	99.99%	99.99%	99.96%	99.94%

4.3 Comparison with previous research works

After completing series of experiments, now the experimental result is compared with previous studies that used various types of datasets, both using machine learning and deep learning classifiers. Besides, we also perform comparison with those studies that consider dimensionality reduction methods or not. The comparison results showed that the proposed method of this research work achieved higher level of accuracy as can be seen in **Table 18**.

Table 18. Comparison of experimental result with previous research works

Ref & (Year)	Method	Dataset	Accuracy
[32] (2017)	PCA + Softmax	KDD CUP 99	84.99%
[16] (2018)	Fisher score + k-NN	N-BaIoT	97.24%
[20] (2019)	Entropy + SVM	N-BaIoT	93.15%
[22] (2020)	Random Forest	MedBIoT	97.66%
[33] (2021)	DNN	NSL-KDD	83.05%
[33] (2021)	DNN	CSE-CIC-IDS2018	95.78%
[19] (2021)	k-NN	Bot IoT	99.60%
[21] (2021)	One class SVM	IOT Security Dataset	99%
[34] (2022)	Decision Tree	MedBIoT	99.41%
This Work	**Autoencoder + ANN**	MedBIoT	**99.64%**

4.4 Discussion

Nowadays, DDOS attacks have become a very serious problem. All networks connected to the Internet are very vulnerable to DDOS attacks. The IoT network has recently developed very rapidly. Thus, it has also become the target of DDOS attacks as revealed by research works in [35]–[37]; DDOS attacks are carried out on cloud computing IoT networks. Besides, DDOS attacks are now carried out on edge computing IoT networks [38].

In detecting cyberattacks, one of the most crucial requirements is to have high detection accuracy. Therefore, researchers attempt to come out with ideas to achieve the requirement. There are two aspects we may consider, i.e.: 1) reducing the data attributes' dimensions and 2) better classifier algorithms. This research work has chosen Autoencoder as dimensionality reduction method that managed very well in reducing the data dimensionality significantly, even with low number of layers of the Autoencoder. It can be verified by the decreasing of the

dataset file size from 152 Megabyte (MB) into 12.6 MB (equivalent to 91.2% reduction).

At the same time, the classification process still provided acceptable accuracy level that can be seen from the accuracy level of the experiment results with different scenario of training and testing dataset separation. The selected features are used as the basis of the ANN classifier to perform normal traffic vs. attack traffic classification. We conducted 5 different dataset separation scenarios to investigate the behavior of the classifier as well as to study the effects of the training data's size on the detection accuracy. We found out that the dataset separation very much affects the detection performance, i.e. the more data portion for the training phase, the better accuracy performance of the ANN classifier. The experiment result with the data separation scenario of 90% training data vs. 10% testing data provided the highest accuracy, i.e.: 99.72%. Moreover, we also measured the other performance metrics to validate the accuracy results. The performance metrics for the best scenario are: Precision value of 99.82%, Specificity value of 99.31%, Sensitivity value of 99.83% and F1-score value of 99.82%. The metrics indicated a good classification result achieved by the proposed method, which combine the 5-layer Autoencoder dimensionality reduction method with ANN classifier.

Similar research works on botnet attacks detection on IoT network have been carried out, such as research work conducted by Zhao et al. [32], where the PCA method was used in dimensionality reduction on KDD CUP 99 dataset and combined with Softmax classifier and obtained detection accuracy rate of 84.99%. Then Bahsi et al. [16] used the Fisher score method in reducing the dimensionality of the N-BaIoT dataset then used k-NN classifier and obtained detection accuracy rate of 97.24%. Another research work conducted by Nomm and Bahsi [20] used a method of reducing the entropy dimensions on N-BaIoT dataset and were further classified using SVM until achieved detection accuracy value of 93.15%.

Furthermore, Kunang et al. [33] used an optimized Deep Neural Networks (DNNs) as classifier, resulting detection accuracy rate of 83.05% on NSL-KDD dataset and 95.78% on CSE-CIC-IDS2018 dataset. On research works using the MedBIoT dataset, Kalakoti et al. [34] implemented the Decision Tree classification method and reported result of 99.41% detection accuracy rate. Other research work by Guerra-Manzanares et al. [22] implemented the Random Forest classification method and reported that the proposed method can detect attacks with an accuracy rate of 97.66%%

Meanwhile, the proposed work achieved 99.72% detection accuracy as the best accuracy. The strength of the proposed work in this paper is the use of dataset that represent medium-sized real IoT network and contains actual IoT botnet malicious network traffic. Therefore, the achieved performance metrics are significant.

5. Conclusion and Future Work

An IoT botnet attack detection system has been developed through a combination of deep Autoencoder and ANN. Experimental results showed that the use of deep Autoencoder as dimensionality reduction method highly affects the results obtained at classification stage. The classification process becomes more efficient, because the data is relatively smaller than the original data so that it can save memory usage. The classification result by applying the ANN algorithm, obtained accuracy value of 99.72%, precision of 99.82%, sensitivity of 99.82%, specificity of 99.31%, and F1-score value of 99.82%. Based on these results, the ANN algorithm has succeeded in increasing the values of Accuracy, Precision, Specificity, F1-score, and Sensitivity for binary cases.

For further research, the authors plan to conduct experiments on classification with more data as well as the implementation of deep learning algorithms. Referring to the research trends on cyber security that will emerge in the future [39], the author also consider the use of

artificial intelligence techniques for detecting, identifying, and responding to various forms of IoT cyber-attacks automatically. Furthermore, preventing IoT data breaches in cloud computing, information degradation, identity disclosure, playback, and denial of service IoT attacks are interesting topics. Thus, leveraging Artificial Intelligence techniques in preventing cyber-attacks on IoT data services on edge computing is considered as future research. Lastly, the use of quantum computing in detecting, identifying, and responding to various forms of IoT cyber-attacks is also become one of the authors focus in the future.

Notations and Acronyms

Notations/acronyms	Definition
ANN	Artificial Neural Network
BACC	Balance Accuracy
CICFlowMeter	A network traffic flow generator and analyser (use for generating 83 network traffic features)
DNN	Deep Neural Network
DDOS	Distributed Denial of Service
FP	False Positive
FN	False Negative
Fig.	Figure
GB	*Gigabyte*
Gen	*Generation*
HD	*High Definition*
IoTID20	Dataset that purposely designed to detect IoT Botnet attacks and it families like DoS, MITM, Scan which includes packet traces of IoT attacks
IoT	Internet of Things
k-NN	K-Nearest Neighbors
MB	Megabyte
MCC	Matthew's Correlation Coefficient
MedBIoT	Medium Botnet IoT dataset
N-BaIoT	Network-Based IoT attacks dataset
NSL-KDD	Network Security Laboratory - Knowledge Discovery in Databases
PCA	*Principal Component Analysis*
RAM	Random Access Memory
ROC	*Receiver Operating Characteristic*
SMOTE	Synthetic Minority Over-Sampling Technique
SSD	*Solid-State Drive*
SVM	Support Vector Machine
TP	True Positive
TN	True Negative
VGA	Video Graphic Adapter
XGBoost	Xtreme Gradient Boosting

References

[1] Y. Perwej, M. Ahmed, B. Kerim, and H. Ali, "An Extended Review on Internet of Things (IoT) and its Promising Applications," *Commun. Appl. Electron.*, vol. 7, no. 26, pp. 8–22, 2019. Article (CrossRef Link).

[2] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu, "The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1606–1616, 2019. Article (CrossRef Link).

[3] U. Kumar, S. Navaneet, N. Kumar, and D. S. Pandey, "Isolation of DDoS Attack in IoT: A New Perspective," *Wirel. Pers. Commun.*, vol. 114, pp. 2493-2510, Oct. 2020. Article (CrossRef Link).

[4] K. Kumari and M. Mrunalini, "Detecting Denial of Service attacks using machine learning algorithms," *J. Big Data*, vol. 9, no. 1, 2022. Article (CrossRef Link).

[5] I. Winkler and A. T. Gomes, "Adversary Infrastructure," *Adv. Persistent Secur.*, pp. 67–79, 2017. Article (CrossRef Link).

[6] A. Khan and C. Cotton, "Detecting attacks on iot devices using featureless 1D-CNN," in *Proc. of 2021 IEEE Int. Conf. Cyber Secur. Resilience, CSR 2021*, pp. 461–466, 2021. Article (CrossRef Link).

[7] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "Machine Learning and Deep Learning Approaches for CyberSecurity: A Review," *IEEE Access*, vol. 10, no. March. pp. 19572 - 19585, 2022. Article (CrossRef Link)

[8] P. S. Saini, S. Behal, and S. Bhatia, "Detection of DDoS Attacks using Machine Learning Algorithms," in *Proc. of International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 16–21, 2020. Article (CrossRef Link)

[9] K. D. R. N. Behera, "A Survey on Machine Learning: Concept, Algorithms and Applications," *Int. J. Innov. Res. Comput. Commun. Eng.*, vol. 5, no. 2, pp. 8198–8205, 2017. Article (CrossRef Link).

[10] E. Hodo et al., "Threat analysis of IoT networks using artificial neural network intrusion detection system," in *Proc. of 2016 Int. Symp. Networks, Comput. Commun. ISNCC 2016*, pp. 1-6, 2016. Article (CrossRef Link).

[11] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks," *Neurocomputing*, vol. 172, pp. 385–393, 2016. Article (CrossRef Link).

[12] Susanto, D. Stiawan, M. A. S. Arifin, J. Rejito, M. Y. Idris, and R. Budiarto, "A Dimensionality Reduction Approach for Machine Learning Based IoT Botnet Detection," in *Proc. of Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2021–Oct, no. October, pp. 26–30, 2021. Article (CrossRef Link)

[13] J. Wang, H. He, and D. V. Prokhorov, "A folded neural network autoencoder for dimensionality reduction," *Procedia Comput. Sci.*, vol. 13, pp. 120–127, 2012, Article (CrossRef Link).

[14] S. LANDER, "An Evolutionary Method For Training Autoencoders For Deep Learning Networks," 2014. Article (CrossRef Link)

[15] Y. Song and S. Hyun, "Analysis of Autoencoders for Network Intrusion Detection," *Sensors*, vol. 21, no. 4294, pp. 1–23, 2021. Article (CrossRef Link)

[16] H. Bahsi, S. Nomm, and F. B. La Torre, "Dimensionality Reduction for Machine Learning Based IoT Botnet Detection," in *Proc. of 2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV*, pp. 1857–1862, 2018. Article (CrossRef Link)

[17] M. Alqahtani, H. Mathkour, and M. M. Ben Ismail, "IoT botnet attack detection based on optimized extreme gradient boosting and feature selection," *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–21, 2020. Article (CrossRef Link) .

[18] A. Bijalwan, "Botnet Forensic Analysis Using Machine Learning," *Secur. Commun. Networks*, vol. 2020. 2020. Article (CrossRef Link).

[19] S. Pokhrel, R. Abbas, and B. Aryal, "IoT Security: Botnet detection in IoT using Machine learning," *arXiv*, pp. 1–11, 2021. Article (CrossRef Link)

[20] S. Nomm and H. Bahsi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," in *Proc. of 17th IEEE International Conference on Machine Learning and Applications, ICMLA*, pp. 1048–1053, 2019. Article (CrossRef Link)

[21] M. H. Raj, A. N. M. A. Rahman, U. H. Akter, K. N. Riya, A. T. Nijhum, and R. M. Rahman, "IoT Botnet Detection Using Various One-Class Classifiers," *Vietnam J. Comput. Sci.*, vol. 8, no. 2, pp. 291–310, 2021. Article (CrossRef Link).

[22] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network," in *Proc. of ICISSP 2020 - Proc. 6th Int. Conf. Inf. Syst. Secur. Priv.*, pp. 207–218, 2020. Article (CrossRef Link).

[23] Y. Meidan et al., "N-BaIoT-Network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Sep. 2018. Article (CrossRef Link).

[24] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic using Time based Features," in *Proc. of ICISSP*, pp. 253–262, 2017. Article (CrossRef Link)

[25] J. Alsamiri and K. Alsubhi, "Internet of things cyber attacks detection using machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 12, pp. 627–634, 2019, Article (CrossRef Link).

[26] M. M. Rasheed, A. K. Faieq, and A. A. Hashim, "Android Botnet Detection Using Machine Learning," *Ingénierie des systèmes d Inf.*, vol. 25, no. 1, pp. 127–130, 2020. Article (CrossRef Link).

[27] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, and F. Jasmir, "Automatic Features Extraction Using Autoencoder in Intrusion Detection System," in *Proc. of 2018 Int. Conf. Electr. Eng. Comput. Sci. ICECOS 2018*, vol. 17, pp. 219–224, 2019, Article (CrossRef Link).

[28] M. Inthachot, V. Boonjing, and S. Intakosum, "Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend," *Comput. Intell. Neurosci.*, vol. 2016. 2016. Article (CrossRef Link).

[29] C. Pascariu and I. D. Barbu, "Dynamic analysis of malware using artificial neural networks : Applying machine learning to identify malicious behavior based on parent process hierarchy," in *Proc. of the 9th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2017*, vol. 2017–Janua, pp. 1–8, 2017. Article (CrossRef Link).

[30] A. Luque, A. Carrasco, A. Martín, and A. de las Heras, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix," *Pattern Recognit.*, vol. 91, pp. 216–231, 2019, Article (CrossRef Link).

[31] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *Proc. of Int. Conf. Pattern Recognit.*, pp. 3121–3124, 2010. Article (CrossRef Link).

[32] S. Zhao, W. Li, T. Zia, and A. Y. Zomaya, "A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things," in *Proc. of 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing*, vol. 2017–Janua, pp. 836–843, 2017. Article (CrossRef Link)

[33] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *J. Inf. Secur. Appl.*, vol. 58, 2021. Article (CrossRef Link).

[34] R. Kalakoti, S. Nomm, and H. Bahsi, "In-Depth Feature Selection for the Statistical Machine Learning-Based Botnet Detection in IoT Networks," *IEEE Access*, vol. 10, no. July, pp. 94518–94535, 2022. Article (CrossRef Link).

[35] B. B. Gupta, A. Gaurav, and D. Peraković, "A Big Data and Deep Learning based Approach for DDoS Detection in Cloud Computing Environment," in *Proc. of 2021 IEEE 10th Global Conference on Consumer Electronics (GCCE)*, pp. 287–290, 2021. Article (CrossRef Link).

[36] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal Load Distribution for the Detection of VM-Based DDoS Attacks in the Cloud," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 114–129, 2020. Article (CrossRef Link).

[37] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and N. Kara, "FoGMatch: An Intelligent Multi-Criteria IoT-Fog Scheduling Approach Using Game Theory," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1779–1789, 2020. Article (CrossRef Link).

[38] R. Islambouli, Z. Sweidan, A. Mourad, and C. Abou-Rjeily, "Towards Trust-Aware IoT Hashing Offloading in Mobile Edge Computing," in *Proc. of 2020 Int. Wirel. Commun. Mob. Comput. IWCMC 2020*. pp. 2216–2221, 2020. Article (CrossRef Link).

[39] S. S. Gill et al., "AI for next generation computing: Emerging trends and future directions," *Internet of Things*, vol. 19, p. 100514, 2022. Article (CrossRef Link).

Deris Stiawan Received the PhD degree in Computer Engineering from Universiti Teknologi Malaysia, Malaysia. He is currently an Associate Professor at Department of Computer Engineering, Faculty of Computer Science, Universitas Sriwijaya. His research interests include computer network, Intrusion Detection/ Prevention System, and heterogeneous network

Susanto Received his PhD degree in Computer Science from Universitas Sriwijaya Palembang, South Sumatera, Indonesia. He is currently a senior lecturer at Faculty of Computer, Universitas Bina Insan, Indonesia. His research interests include cryptography, information technology, information security, and network security.

Abdi Bimantara Received bachelor degree from Department of Computer Engineering, Universitas Sriwijaya, Indonesia, in 2021. His main interest includes both theoretical and practical aspect of Internet of Things and Computer Network under Computer Network & Information Security (COMNETS) Research Group at Faculty of Computer Science, Universitas Sriwijaya since 2017.

Mohd Yazid Idris An Associate Professor at School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia. He obtained his M.Sc and Ph.D. in the area of Software Engineering, and Information Technology (IT) Security in 1998 and 2008 respectively. In software engineering, he focuses on the research of designing and development of mobile and telecommunication software. His main research activity in IT security is in the area of Intrusion Prevention and Detection (IPD).

Rahmat Budiarto Received B.Sc. degree from Bandung Institute of Technology in 1986, M.Eng. and Dr.Eng. in Computer Science from Nagoya Institute of Technology in 1995 and 1998, respectively. Currently, he is a full Professor at College of Computer Science and IT, Albaha University, Saudi Arabia. His research interests include intelligent systems, brain modeling, IPv6, network security, Wireless sensor networks, and MANETs.