IMPLEMENTATION OF BACKPROPAGATION NEURAL NETWORK AND ADABOOST ALGORITHMS IN NETWORK INTRUSION DETECTION SYSTEM

Nowo Adisuryo[1], Ken Kinanti Purnamasari[2]

[1,2]Informatic Engineering – Indonesian Computer University
Jl. Dipatiukur 112 – 114 Bandung
E-mail : nowoadisuryo@gmail.com[1,] ken.kinanti@email.unikom.ac.id[2]

ABSTRACT

Network Intrusion Detection System (NIDS) is a software or hardware that can analyze traffic flowing to a network. NIDS will alert the Network Administrator if there is an attempted attack or intrusion into the network system. In NIDS, machine learning algorithms have been widely applied, for example, Decision Tree, SVM, Naive Bayes, Artificial Neural Network, and others. There are challenges in the use of machine learning for NIDS: the system must have a high degree of accuracy, low error rate, fast computing time, and less memory usage in detecting attacks on the network [1]. In this research we will use backpropagation neural network (BPNN) and adaptive boost (AdaBoost) algorithm on NIDS to classify network data packets into normal data packets or attacks. Based on the test results using the Kyoto 2006 + dataset consisting of 2000 data, the BPNN method obtains a lower accuracy when compared to the BPNN method with AdaBoost. The BPNN method obtains an accuracy of 99.75% while the BPNN with AdaBoost method receives 99.8% accuracy.

Keyword — backpropagation, adaboost, kyoto dataset, network intrusion detection system

1. INTRODUCTION

The rapid development and proliferation of botnets, large-scale attacks such as DDoS, email spam, and others on computer networks are becoming a more serious and dangerous threat than previous attacks. Based on this threat, therefore the Network Based Intrusion Detection System (NIDS), Intrusion Prevention System (IPS), firewalls became indispensable for maintaining valuable computer systems, networks and information [2].

NIDS is the deployment of real-time monitoring network sensors at vital locations in the network infrastructure. Network sensors analyze traffic and detect attack activity as well as suspicious activity [3]. Based on the techniques used NIDS can be classified into 2 different categories. One is anomaly detection and the other is a signature detection or misuse detection [4].

The use of machine learning for NIDS has been widely used in previous studies. Such as the use of Decision Tree method, SVM, Naive Bayes, Artificial Neural Network, and others. There are challenges in the use of machine learning for NIDS: the system must have a high degree of accuracy, low error rate, fast computing time, and less memory usage in detecting attacks on the network [1].

Research conducted by Harshal A. Sonawane and Tareek M. Pattewar [1] using Backpropagation Neural Network (BPNN) and Principal Component Analysis (PCA) methods in IDS, this study compares between the use of BPNN and PCA with the use of BPNN alone. The results of this study, the use of BPNN and PCA get an accuracy of 86.90% and the use of BPNN alone get an accuracy of 90.20%. Research conducted by Indraneel Mukhopadhyay and colleagues [5] using BPNN approach on IDS, from this study resulted in accuracy of 95.6%. Research conducted by Gang Wang et al. [6] using Artificial Neural Networks (ANN) and fuzzy clustering method on IDS, in this study also compared other methods, namely Decision Tree, Naive Bayes, BPNN, FC-ANN (Fuzzy Clustering - Artificial Neural Networks). The research yielded 96,71% accuracy for FC-ANN, 96,65% for BPNN, 96,11% for Naive Bayes, 96,75% for Decision Tree. Research conducted by Weiming Hu and colleagues [7] uses the AdaBoost algorithm and as a weak classifier using the Decision Stumps algorithm. This study yielded an accuracy of 90.04% - 90.88%. Research [1], [5], [6], [7] uses the 1999 KDD CUP dataset as training and test data. The 1999 KDD CUP dataset has been heavily criticized because it no longer represents the current network conditions [2]. In theory the accuracy of learning algorithms can be improved by using Boosting [8]. One of the boosting methods is Adaptive Boosting (AdaBoost). The AdaBoost method can improve the accuracy of binary classification cases. To boost with AdaBoost requires a basic algorithm as a weak learner algorithm whose output of the algorithm will be integrated with AdaBoost to create strong learner [9]. Research conducted by Jianfang Cao and colleagues [9] has implemented BPNN and AdaBoost to classify the emotional semantics contained in landscape images. The results of the research BPNN and AdaBoost algorithms get a higher accuracy rate

of about 15% compared to previously developed methods. The performance of BPNN and AdaBoost in NIDS case is not yet known, therefore in this research will use BPNN and AdaBoost algorithm for NIDS case.

2. CONTENTS OF RESEARCH

2.1 Preprocessing

Preprocessing consists of encoding and min-max normalization.

2.1.1 Encoding

In supervised learning there are generally various forms of data. The data can be in the form of numbers or words. If the data is in the form of numbers, then the algorithm can directly process it. If the data is in the form of a word, then the data must be transformed into a numerical form so that the algorithm can process it.

On the encoding label the way of transforming word-shaped data into numeric form is as follows. Suppose there is a feature with category-shaped data, {'audi', 'bmw', 'ford', 'toyota'}, the data will be transformed into numeric form by replacing index data to 0, ..., n, n is length of the array, with its index, {'0', '1', '2', '3'} [10].

2.1.2 Min-Max Normalization

Normalization min-max runs a linear transformation on the original data. Suppose minA and maxA are the minimum and maximum values of an attribute, A. Normalization min-max maps a value, xi, from A to x'i in the range [new_minA, new_maxA] by calculating

$$x'_i = \frac{x_i - min_j}{max_j - min_j}(new_max_j - new_min_j) + new_min_j \quad (2.1)$$

Normalization min-max maintains the relationship between the original data values. This normalization will find an "out-of-range" error if a future input case is outside the original data range for A [11].

2.2 Backpropagation

Backpropagation is one of the supervised learning algorithms used in artificial neural networks. Backpropagation looks for a combination of weights to minimize output errors to be considered to be the correct solution [12]. Backpropagation is simply done in two stages, namely:

1. Feedfoward

The pattern to be trained is set to each unit in the input layer, then the resulting output is transmitted to the next layer and continues until the output layer.

2. Backpropagation

Based on the expected output, each weight is adjusted to produce a minimum error ranging from the weights connected to the output of the neuron, then backtracking down to the input layer.

One of the popular activation functions for backpropagation networks is the binary sigmoid defined as (2.2). For the backpropagation stage use the derivative function that is (2.3).

$$f(X) = \frac{1}{1+e^{-X}} \quad (2.2)$$

$$f'(X) = f(X)(1 - f(X)) \quad (2.3)$$

The following stages in the training process backpropagation algorithm.

Feedfoward:

Step 1: Initialize weights and parameters

Step 2: Each input unit (X_i, i = 1, 2, 3, ..., n) sends the input signal to each input located on the hidden layer.

Step 3: Each unit in the hidden layer (Z_j, j = 1, 2, 3, ..., p) multiplied by its weight and sum and added normally.

$$z_in_j = V_{0,j} + \sum_{i=1}^{n} X_i V_{i,j} \quad (2.4)$$

calculating the hidden signal by using the activation function.

$$Z_j = f(z_in_j) \quad (2.5)$$

Step 3 : Each of output unit (y_k, k = 1, 2, 3, ..., m) multiplied by weight and summed up normally.

$$y_in_k = W_{0,k} + \sum_{j=1}^{p} Z_j W_{j,k} \quad (2.6)$$

calculate the output signal by using the activation function.

$$y_k = f(y_in_k) \quad (2.7)$$

Backward:

Step 4 : Each of output unit (y_k, k = 1, 2, 3, ..., m) receive target tk pattern according to input pattern during training and then error information of output layer (δ_k) calculated.

$$\delta_k = (t_k - y_k) * ((y_k) * (1 - y_k)) \quad (2.8)$$

Then calculate the weight correction (which will be used later to improve the value $W_{j,k}$):

$$\Delta W_{j,k} = \alpha \delta_k Z_j \quad (2.9)$$

Calculate the bias correction to correct the value $W_{0,k}$:

$$\Delta W_{0,k} = \alpha \delta_k \quad (2.10)$$

Step 5 : On each hidden layer (Z_j, j = 1, 2, 3, …, p) sums the delta input (of the units in the previous layer)

$$\delta_in_j = \sum_{k=1}^{m} \delta_k W_{j,k} \quad (2.11)$$

$$\delta_j = (\delta_in_j)(z_in_j)(1 - z_in_j) \quad (2.12)$$

Then calculate the weight correction to improve the weight V.

$$\Delta V_{i,j} = \alpha \delta_k X_i \quad (2.13)$$

Calculate bias weight correction $V_{0,j}$.

$$\Delta V_{0,j} = \alpha \delta_j \quad (2.14)$$

Step 6 : Each of output unit (y_k, k = 1, 2, 3, ..., m) done to correct the bias and its weight (j = 0, 1, 2, 3, ..., p) resulting in new weight and bias.

$$W_{j,k}(baru) = W_{j,k}(lama) + \Delta W_{j,k} \quad (2.15)$$

Likewise for weights on the input layer to the hidden layer, repaired by using the weight correction that has been calculated before.

$$V_{i,j}(baru) = V_{i,j}(lama) + \Delta V_{i,j} \quad (2.16)$$

2.3 Backpropagation and Adaptive Boost

Referring to the AdaBoost algorithm, the efficiency and accuracy of the AdaBoost algorithm based on a weak classifiers series depends largely on the accuracy of the weak classifier itself. In this study Backpropagation Neural Network was chosen as a weak classifier and the end result of AdaBoost classifier required a method to combine the weak classifier results into an efficient AdaBoost classifier. In this study we will use the weighted voting method to combine the output of each weak classifier to obtain final outputs [13]. The algorithm for building hybrid models from AdaBoost and Backpropagation can be seen as follows.

Step 1 : Select samples for training randomly from the dataset using the 10-fold cross-validation method. Each sample training group will be assigned the same weight.

T = $\{(x_1, y_1), \dots, (x_i, y_i), \dots, (x_n, y_n)\}$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$.

The weights of this distribution in the training samples of the t round will be denoted by $D_t(i)$. Initially, all weights will be assigned the same value.

$$D1(i) = \frac{1}{n}, i = 1, 2, \dots, n, \quad (2.17)$$

where n is the total number of samples in the training dataset.

Step 2 : The parameters of BPNN, which include the structure of the network, the neurons at each layer, the expected target, will be organized according to the attributes of the input and output of the training samples. After that initialization of the weight and bias of BPNN.

Step 3 : Train using BPNN with weighted samples.

Step 4 : When the training epochs come to t round, a weak classifier $h_t(x)$ will be obtained by distribution weights D_t and after that calculate the weighted error ε_t from the model:

$$\varepsilon_t = \sum_{i=1}^{n} D_t(i) \text{ x } I(h_t(x_i)) \quad (2.18)$$

where $I(h_t(x_i))$ is an indicator function. The mathematical expression of the indicator function can be seen as follows.

$$I(h_t(x_i)) = \begin{cases} 1, jika\ h_t(x_i) \neq y_i \\ 0, jika\ h_t(x_i) = y_i. \end{cases} \quad (2.19)$$

Step 5 : Calculate the weights of the weak classifier

$$a_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right). \quad (2.20)$$

If the predicted error of the samples is less than 0.5, the weight of the weak classifier will be greater than zero, with reduced predictive errors will increase the weights of the weak classifier, which means the greater weighting of this weak classifier in the final output.

Step 6 : Update the weights on each sample on the training dataset for the next iteration.

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \exp(-a_t \text{ x } y_i \text{ x } h_t(x_i)) \quad (2.21)$$

where Z_t is the value of normalization constant for $D_{t+1}(i)$ whose mathematical expression can be seen as follows.

$$Z_t = \sum_{i=1}^{n} D_t(i) \text{ x } \exp(-a_t \text{ x } y_i \text{ x } h_t(x_i). \quad (2.22)$$

By doing so, weights on training data which are misclassified by the weak classifier will now increase in value or will decrease in value if samples are correctly classified.

Step 7 : As long as the training iteration continues, a series of weak classifiers will be obtained, then combine the weak classifiers during the training process with the following equation.

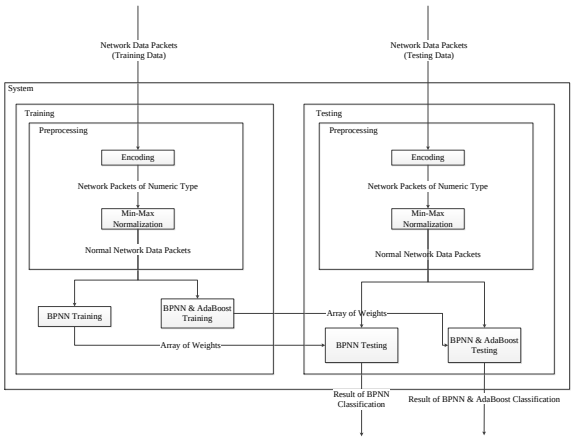
$$f(x) = \sum_{t=1}^{T} a_t \text{ x } h_t(x). \quad (2.23)$$

A series of weak classifiers with weights of diversity is collected by looping from step 2 to step 5. In the end, a AdaBoost model based on a trained weak classifier will be built with the following equation.

$$H(x) = \text{sign}(f(x)). \quad (2.24)$$

2.4 Discussion

The system includes two stages of training and testing process. The training process is done until the desired weight is met. If the desired conditions are met then the next can proceed to the testing process. In the test process, will be included test data to classify the results will be compared from the weight of the training done before. the system process flow can be seen in Figure 2.1.



Gambar 2.1 Alur Proses Sistem

The system process flow in Figure 2.1 begins by entering network data packets stored in comma separeted values (.csv) form on the system. In the preprocessing stage is divided into two processes namely the encoding process and data normalization. The encoding process is to transform the data in the form of a string into a numeric form to be processed by the algorithm. The process of normalizing the data is to normalize the data so that the data value is in the range -1.0 to 1.0 [11]. At the classification stage for training and testing using ANN with Multi Layer Fowards architecture and ANN integration with AdaBoost. ANN input neurons are normalized network data packets. Then the training process on

ANN and ANN integration with AdaBoost using the backpropagation algorithm resulting from this training generates the weights that are used for the stages in the testing process. After the testing process it will be obtained classification results that network data packets including normal or attack.

2.4.1 Preprocessing

Preprocessing stage is divided into two processes, namely encoding and normalization of data packet data set dataset. Normalization technique used is normalization min-max. The result of preprocessing will produce an encoded and normalized dataset which will be useful as input data for each classification stage.

2.4.1.1 Encoding

The data encoding process is done to convert the data of the type of string into data of numeric type, so that the data can be used for each step in classification.

The technique of encoding the data will be done by labeling the encoding. Block diagram for the encoding stage can be seen in Figure 2.2.

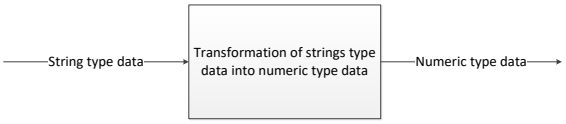


Figure 2.2 Block Diagram of Encoding Process

As we can see in Figure 2.2 data into input is data type string, then the data will be through the encoding process so that the data will be numeric type after through the encoding process.

2.4.1.2 Min-Max Normalization

Normalization of min-max is a process of normalization of numeric data so that data can be in the range of values between -1.0 to 1.0 [11]. The data to be normalized is in all features. Normalization of data using equation (2.19). Block diagram for normalization phase of data can be seen in Figure 2.3.

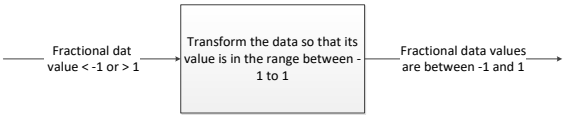


Figure 2.3 Block Diagram Min-Max Normalization Process

As we can see in Figure 2.3 all fractional data whose value < -1 or > 1 will be transformed so that the data will be in the range between -1 and 1.

2.4.2 Backpropagation

The process that runs on this training and testing uses backpropagation algorithm. The number of neurons in the backpropagation architecture used in this study can be seen in Figure 2.4.

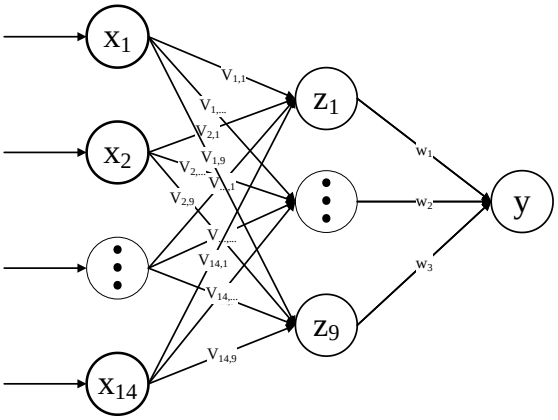


Figure 2.4 Backpropagation Architecture

In Figure 2.4 there is a neuron on the input layer (X) with a total of 14 neurons, this is determined based on the number of features used in this study are 14 features, X_j where j = 1, 2, ..., 14, will contain data on the j-feature. While on the hidden layer (Z) there are nine neurons, this amount is based on comparisons using 5 and 13 neurons. Neurons in the output layer amount to 1 neuron which is a normal class or attack. Among the input, hidden, and output neurons are the connecting weights of the weights between the hidden-input layers (V) and the hidden-ouput layer (W) and the bias weight (V_0) and (W_0).

2.4.2.1 Backpropagation Training

The process done in the training phase using backprpagation is the initialization phase of weights, feedfoward process, backpropagate, and weight update.



Figure 2.5 BPNN Training Block Diagram

Can be seen on the block diagram of BPNN training, the input is a normal network data packet, the first stage is the initialization of neural networks and training parameters, the second stage is the initialization of weights where in this study the value for the weights is in the range -0.5 to 0.5, the third stage is training with BPNN which will be stage feedfoward and backward until the training conditions stopped met, after the training will get the weight of training results.

2.4.2.2 Backpropagation Testing

Test phase using backpropagation algorithm using feedfoward process. The weights generated during the training phase will be used for the testing process.

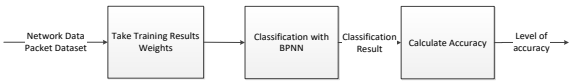


Figure 2.6 Block Test Process Diagram with BPNN

As we can see in Figure 2.6 input from the system in the form of network data packets of normal value, then the system will take the weight of previous training results, after that will be classified with BPNN, and the next process the system will calculate the accuracy of the classification. The output of this process is the result of the classification and accuracy of the BPNN algorithm.

2.4.3 Backpropagation and Adaptive Boost

The process that runs on this training and testing uses backpropagation algorithms as a weak classifier and AdaBoost to build strong classifier. The number of neurons in the backpropagation architecture used in this study can be seen in Figure 2.4. The number of weak classifiers used will be adjusted during the training process, the parameters of the number of neurons in the input layer, the hidden layer, and the output layer will be adjusted based on the backpropagation architecture in Figure 2.4.

2.4.3.1 Backpropagation and AdaBoost Training

The process performed at the training stage using backprpagation and AdaBoost is the initialization phase of sample weights ($D_{i,1}$) for each sample, where i is the row index, backpropagation training using weighted samples, calculation of weak classifier error (ε), calculation of the weight of weak classifier (a), and update the weight matrix D.



Figure 2.7 Block Diagram of Training Process BPNN and AdaBoost

As we can see in Figure 2.7, the input is a normal network data packet, the first stage is the initialization of neural networks and training parameters, the second stage is the initialization of sample weights, neural netowork weights in which the value for neural network weights is in the range -0 , 5 to 0.5, the third stage is the training with BPNN and AdaBoost where will be the stage feedfoward, backward, get a weak classifier, calculate the error classifier error, and update the sample weights, this stage is done until the training conditions stopped met, after training will get the weight of training outcomes.

2.4.3.2 Backpropagation and AdaBoost Testing

The testing phase of BPNN and AdaBoost uses the feedfoward process on BPNN for each round c. After the rotation value c = C, it will be the weighted voting process of the classification class weakness classification to obtain the classification of BPNN and AdaBoost.

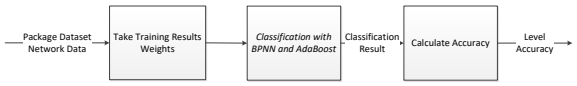


Figure 2.8 Block Diagram Testing Process BPNN and AdaBoost

As we can see in Figure 2.8 input from the system in the form of network data packets of normal value, then the system will take the weight of previous training results, after that will be classified with BPNN and AdaBoost, and the next process the system will calculate the accuracy of the classification. The output of this process is the classification and accuracy of the BPNN and AdaBoost algorithms.

2.4.4 Testing Result

The test was performed to classify 2 classes of normal and attack, using a total of 2000 data tested by the 10-fold cross validation method.

2.4.4.1 Test Scenario 1

In test scenario 1, testing of BPNN and BPNN methods with AdaBoost is done by changing the value of the hidden layer number to 1, 3, 6, 8, 10, while the other variable is filled with the same value. Here are the other variable values used in test scenario 1, the value of learning rate is 0.1, Epoch 1000, Target Error 0.05, Weak Classifier 3.

From the test scenario 1 we get the overall results that can be seen in Table 2.1.

Table 2.1 Data Result Scenario Testing 1

Neuron	Accuracy of BPNN	MSE BPNN	Accuracy of BPNN with *AdaBoost*	MSE BPNN with *AdaBoost*
1	93,15%	0,490	93,2%	0,108
3	99,25%	0,490	99,4%	0,128
6	99,2%	0,543	99,5%	0,139
8	99,4%	0,494	99,65%	0,136
10	99,25%	0.511	99,5%	0.131

From the result data of test scenario 1 it can be concluded that the number of neurons that have the highest accuracy either on BPNN or BPNN with AdaBoost in this case are 8 neurons with 99.4% accuracy for BPNN and 99.65% for BPNN with AdaBoost.

2.4.4.2 Test Scenario 2

In testing scenario 2, testing of BPNN and BPNN methods with AdaBoost is done by changing the value of the learning rate to 0.05, 0.1, 0.15, 0.2, 0.25,

while other variables are filled with the same value. Here are the other variable values used in the test scenario 2. The number of neurons in the hidden layer is 8, Epoch 1000, Target Error 0.05, Weak Classifier 3.

From the test scenario 2 we get the overall result which can be seen in Table 2.2.

Table 2.2 Data on Scenario Results Testing 2

Learning Rate	Accuracy of BPNN	MSE BPNN	Accuracy of BPNN with *AdaBoost*	MSE BPNN with *AdaBoost*
0,05	99,15%	0.490	99,5%	0.136
0,1	99,25%	0.490	99,4%	0.128
0,15	93,3%	0.543	93,3%	0.107
0,2	99,65%	0.494	99,55%	0.144
0,25	99,75%	0.518	99,8%	0.112

From the result data of test scenario 2 it can be concluded that the learning rate that has the highest accuracy either on BPNN or BPNN with AdaBoost in this case is 0.25 with accuracy of 99.75% for BPNN and 99.8% for BPNN with AdaBoost .

3. CLOSING

Based on the results of tests that have been done, it can be concluded that the method of BPNN and BPNN with AdaBoost which is implemented into the network packet classification system has the highest accuracy of 99.75% for the BPNN and 99.8% for the BPNN with AdaBoost. This accuracy is derived from a variable arrangement that has a number of neurons of 8 neurons, a learning rate of 0.25, an epoch limit of 1000, a target error of 0.05, and a weak classifier number of 3.

The proposed suggestion is the addition of dataset used in the study, because degan the more varied dataset used then the better also method of BPNN and BPNN with AdaBoost in its classification.

BIBLIOGRAPHY

[1] Harshal A. Sonawane dan Tareek M. Pattewar, "A Comparative Performance Evaluation of Intrusion Detection based on Neural Network and PCA," *IEEE ICCSP 2015 conference*, 2015.

[2] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Masashi Eto, Daisuke Inoue, dan Koji Nakao, "Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation," *Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security - BADGERS*, 2011.

[3] Gert Schauwers dan Gert DeLaet, *Network Security Fundamentals*. California: Cisco Press, 2004.

[4] Nutan Farah Haq, Abdur Rahman Onik, Avishek Khan Hridoy, Musharrat Rafni, Faisal Muhammad Shah, dan Dewan Farid, "Application of Machine Learning Approaches in Intrusion Detection System: A Survey," *International Journal of Advanced Research in Artificial Intelligence*, vol. 4, no. 3, hal. 9–18, 2015.

[5] Indraneel Mukhopadhyay, Mohuya Chakraborty, S. Chakrabarti, dan Tanusree Chatterjee, "Back propagation neural network approach to Intrusion Detection System," *2011 International Conference on Recent Trends in Information Systems*, 2011.

[6] Gang Wang, Jinxing Hao, Jian Ma, dan Lihua Huang, "A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering," *Expert Systems with Applications*, 2010.

[7] Weiming Hu, Wei Hu, dan Stephen J. Maybank, "AdaBoost-based algorithm for network intrusion detection," *IEEE Transactions on Systems, Man, and Cybernetics*, 2008.

[8] Holger Schwenk dan Yoshua Bengio, "Boosting Neural Networks," *Neural Computation*, 2000.

[9] Jianfang Cao, Junjie Chen, dan Haifang Li, "An AdaBoost-backpropagation neural network for automated image sentiment classification," *Scientific World Journal*, 2014.

[10] Prateek Joshi, *Python Machine Learning Cookbook*. Birmingham: Packt Publishing, 2016.

[11] Jiawei Han, Micheline Kamber, dan Jian Pei, *Data Mining: Concepts and Techniques*. Massachusetts: Morgan Kaufmann, 2012.

[12] Raul Rojas, "Neural Networks A Systematic Introduction," *Neural Networks*, 1996.

[13] Feng Shen, Xingchao Zhao, Dao Lan, dan Limei Ou, "A Hybrid Model of AdaBoost and Back-Propagation Neural Network for Credit Scoring," *Proceedings of the Eleventh International Conference on Management Science and Engineering Management*, 2018.

[14] S. Sahu dan B. M. Mehtre, "Network Intrusion Detection System Using J48 Decision Tree," *Advances in Computing, Communications and Informatics (ICACCI), 2015 International Conference on*, 2015.