Improving Accuracy and Speed of Network-based Intrusion Detection using Gradient Boosting Trees

Ryosuke Terado[1] and Morihiro Hayashida[2]
[1]*Planning and Sales Group, WORKS Co., Ltd, Masuda, Shimane, Japan*
[2]*Department of Electrical Engineering and Computer Science, National Institute of Technology, Matsue College, Matsue, Shimane, Japan*

Keywords: Network-based Intrusion Detection System, Gradient Boostring Tree, Neural Network.

Abstract: As computers become more widespread, they are exposed to threats such as cyber-attacks. In recent years, attacks have gradually changed, and security software's must be frequently updated. Network-based intrusion detection systems (NIDSs) have been developed for detecting such attacks. It, however, is difficult to detect unknown attacks by the signature-based NIDS that decides whether or not an access is malicious based on known attacks. We aim at developing a methodology to efficiently detect new unidentified attacks by constructing a model from latest access records. Kyoto 2016 dataset was constructed for the evaluation of such methods, and machine learning methods including random forest (RF) were applied to the dataset. In this paper, we examine a deep neural network and gradient boosting tree methods additionally for session data with twelve features excluding IP addresses and port numbers on Kyoto 2016 dataset. The average accuracy by a gradient boosting method XGBoost achieved 0.9622 more than five times faster than RF. The results suggest that XGBoost outperforms other machine learning classifiers, and the elapsed time for the classification is significantly shorter.

1 INTRODUCTION

In recent years, information technology has been promoted by technological advances, while the increase in the number of cyber-attacks has been regarded as a problem. In particular, attacks through computer networks on most important infrastructures such as power plants, large-scale factories, medical care, transportation, and government-related facilities can disrupt our power, water, and gas supply networks. One of the defense methods against such attacks is to detect them using a network-based intrusion detection system (NIDS), which is a system that detects in real time and informs us of malicious accesses in the network to be monitored. Currently, the signature-based NIDS that stores patterns of attacks and detects whether or not an observed access matches one of the stored patterns is mainly used.

Since the signature-based system is not effective to unknown attacks, various studies on the performance of NIDS have been conducted (Candora *et al.*, 2009), and especially NIDSs using machine learning methods have attracted attention. An effective feature selection method for an IDS was proposed, and the IDS was combined with least squares support vector machine (LSSVM) (Ambusaidi *et al.*, 2016). An IDS combined with three machine learning methods, k-means clustering (k-means), k-nearest neighbors (k-NN), and naïve Bayes (NB) was proposed (Om and Kundu, 2012). Support vector regression (SVR) optimized by combining the ant colony optimization and firefly algorithm was developed for detecting denial of service (DoS) attacks (Hosseini *et al.*, 2015). An intrusion detection method was proposed using anomaly detection using three unsupervised learning methods, cluster-based classification, k-NN, and one-class support vector machine (OCSVM) (Eskin *et al.*, 2002). It was reported that a network controlled by software-defined networking technology is vulnerable to distributed denial of service (DDoS) attacks, and an SVM-based DDoS attack detection method was proposed (Kokila *et al.*, 2014). IDSs based on neural network (NN) and SVM were constructed and compared (Mukkamala *et al.*, 2002). A random forest (RF)-based ensemble learning method for IDS was improved (Masarat *et al.*, 2016). An intrusion detection method combining a decision

Terado, R. and Hayashida, M.
Improving Accuracy and Speed of Network-based Intrusion Detection using Gradient Boosting Trees.
DOI: 10.5220/0008963504900497
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 490-497
ISBN: 978-989-758-399-5; ISSN: 2184-4356

tree (DT) with a genetic algorithm was proposed for feature selection (Stein *et al.*, 2005). The classification performance by DT and NB was compared (Amor *et al.*, 2004). Deep neural networks including recurrent neural networks have been examined for intrusion detection (Nadeem *et al.*, 2016; Potluri and Diedrich, 2016; Yin *et al.*, 2017).

In these studies, DARPA intrusion detection data sets (https://www.ll.mit.edu/r-d/datasets), KDD Cup 1999 data (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html), and Kyoto 2006+ dataset (Song *et al.*, 2011) have been used for evaluation of developed detection methods. These datasets are already old, and the latest attack tendency is not reflected and the period of data collection is too short. Hence, Kyoto 2016 dataset for enhancing the detection accuracy of NIDSs was constructed, the effectivity of machine learning methods, RF, DT, NB, SVM, k-NN, and OCSVM was confirmed for the dataset, and it was reported that the detection accuracy by RF was the best (Tada *et al.*, 2017).

In this paper, we consider the following problem: Given a network access behavior, obtained mainly from IP packets, discriminate whether or not the access is malicious. We first construct a deep neural network (DNN), and investigate whether or not attack detection tasks by neural networks are really practical. In addition, we examine two gradient boosting trees (GBTs), Friedman's GBT (Friedman, 2001) and XGBoost (Tianqi and Carlos, 2016), that are known to have higher regularization performance. In fact, it was reported that XGBoost was useful to the NSL-KDD dataset (Dhaliwal *et al.*, 2018). We perform computational experiments on Kyoto 2016 dataset. The results suggest that XGBoost outperforms other machine learning classifiers in the previous work. Furthermore, the elapsed time required for classification by GBTs was much shorter than those by other methods. It is important because an NIDS must classify a large amount of accesses.

2 METHODS

We propose a deep neural network for detecting attacks from IP packets, and briefly review two gradient boosting tree methods, Friedman's GBT (Friedman, 2001) and XGBoost (Tianqi and Carlos, 2016). Suppose that N data $(\boldsymbol{x}_i,\ y_i)_{i=1}^{N}$ with a vector $\boldsymbol{x}_i$ having features on the i-th session and a label y_i representing whether or not the session is malicious are given, where we expect features such as the amount of transmitted data, and the number of sessions having the same destination IP address, given in Kyoto 2016 dataset as elements of $\boldsymbol{x}_i$.

2.1 Deep Neural Network Model

Neural networks are often used in various research fields because it can respond flexibly to any problem. It, however, is not easy to build a highly accurate model in general. Figure 1 illustrates a fully connected neural network model. Given the value $x_i^{(j)}$ of the j-th neuron in the input layer, the value z_k of the k-th neuron in the next layer is calculated by

$$z_k = h\left(\sum_j w_{kj} x_i^{(j)} - b_k\right), \tag{1}$$

where h denotes an activation function, w_{kj} denotes the weight between the k-th neuron and j-th neuron, and b_k denotes a bias. The design of middle layers is very important to construct a model with high regularization performance. If the number of middle layers and the number of nodes are small, there is a high possibility that it will not be sufficiently trained. Hence, we construct a fully connected neural network with one hundred middle layers, where each middle layer has one hundred neurons, the output layer has two neurons corresponding to attack and normal accesses, the activation function from the input layer and a middle layer to another middle layer are the hyperbolic tangent function, and the activation function from the last middle layer to the output layer is the softmax function.

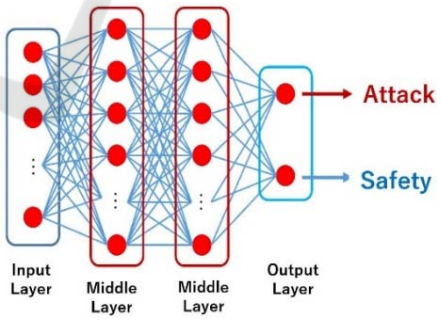

Figure 1: Illustration on a fully connected neural network.

2.2 Friedman's GBT Algorithm

Friedman's Gradient Boosting Algorithm is the first GBT algorithm, which is one of the ensemble learning methods that combine multiple classifiers with low classification performance, called weak learners. Ensemble learning includes three types of learning, bagging, boosting, and stacking. A GBT

belongs to boosting. In boosting, the first weak learner learns one dataset, and the next weak learner learns so that the example misclassified by the previous weak learner can be preferentially classified correctly. Thus, parallel processing is impossible, but classification accuracy tends to be high. The Friedman's GBT tries to construct learners with parameters that minimizes the error function $\Psi(y, f(\boldsymbol{x}))$ for N input data. It, however, is difficult to find the minimum solution. Hence, the following gradient $g_t(\boldsymbol{x})$ for the training data is calculated.

$$g_t(\boldsymbol{x}) = E_y\left(\frac{\partial\Psi(y,\ f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})}\middle|\boldsymbol{x}\right)_{f(\boldsymbol{x})=\hat{f}^{t-1}(\boldsymbol{x})} \quad (2)$$

Then, the parameters ρ_t, θ_t of the weak learner at the *t*-th step are obtained by minimizing the following function.

$$\sum_{i=1}^{N}\left(-g_t(\boldsymbol{x}_i) + \rho_t h(\boldsymbol{x}_i, \theta_t)\right)^2, \quad (3)$$

where *h* denotes a base-learner model.

2.3 XGBoost Algorithm

XGBoost is also a gradient boosting tree method. Let a function f_t represent the t-th decision tree, $\hat{y}_i$ be the predicted value of the i-th decision tree after i steps, that is, $\hat{y}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i)$, y_i be the i-th actual value, Ψ be an error function, and $\Omega(f_t)$ be a penalty function written by $\gamma T + \frac{1}{2}\lambda||\boldsymbol{w}||^2$, where T denotes the size of a decision tree, γ denotes a penalty for T, λ denotes the contribution ratio of the previous tree in constructing a new tree, and $\boldsymbol{w}$ denotes a vector returned by the decision tree f_t. Then, the objective function $L^{(t)}$ to be minimized is defined using $f_t(\boldsymbol{x_i})$ by

$$L^{(t)}(f_t) = \sum_{i=1}^{n}\Psi\left(y_i, \hat{y}_i{}^{(t-1)} + f_t(\boldsymbol{x}_i)\right) + \Omega(f_t). \quad (4)$$

The quadratic Taylor expansion around 0 with respect to f_t is taken as follows.

$$L^{(t)}(f_t) \approx \sum_{i=1}^{n}\left(\Psi\left(y_i, \hat{y}_i{}^{(t-1)}\right) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2}h_i {f_t}^2(\boldsymbol{x}_i)\right) + \Omega(f_t), \quad (5)$$

where $g_i = \partial_{\hat{y}_i{}^{(t-1)}}\Psi\left(y_i, \hat{y}_i{}^{(t-1)}\right)$ and $h_i = \partial^2{}_{\hat{y}_i{}^{(t-1)}}\Psi\left(y_i, \hat{y}_i{}^{(t-1)}\right)$.

By removing terms that are not related to the optimization, that is, terms not related to f_t, from Eq.(5), we have

$$\tilde{L}^{(t)}(f_t) = \sum_{j=1}^{T}\left(\sum_{i\in I_j} g_i w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_i + \lambda\right){w_j}^2\right) + \gamma T. \quad (6)$$

Then, the optimal solution ${w_j}^*$ that is returned by the final node *j* of the t-th decision tree minimizing the objective function is represented by

$${w_j}^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}. \quad (7)$$

In this formula, the first and second gradient of the t -1 decision trees is used to construct t -th decision tree. By substituting ${w_j}^*$ to Eq.(6), we have

$$\tilde{L}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T. \quad (8)$$

Since it is difficult to find the optimal tree structure, the greedy algorithm tries to iteratively add branches to the tree. Suppose that an instance set *I* is splitted to I_L and I_R. Then, if $\tilde{L}^{(t)}(q)$ decreases, the branch is adopted.

The major difference from the Friedman's algorithm is that XGBoost uses not only the first-order gradient but also the second-order gradient to minimize the loss function. Therefore, the convergence of the loss function is fast, the number of trees can be reduced compared to the Friedman's algorithm, and the time required for training can be reduced. If the number of trees can be reduced, the model can be simplified and the time required for classification of test data can be reduced. It is more practical to be able to discriminate in a short time even if a large number of cyber-attacks are received.

3 COMPUTATIONAL EXPERIMENTS

3.1 Kyoto 2016 Dataset

It is important that datasets used for evaluation of attack detection performance contain sufficient long period of data collection and the latest attack tendency. Kyoto 2016 dataset (Tada *et al.*, 2017) was constructed to deal with more shrewd attacks and includes the following features: (1) the latest attack trends included (2) constructed from actual traffic (3)

long period of data collection (approximately 10 years from November 2006 to December 2015).

The number of sessions observed in about 10 years is 806,095,624, of which 160,873,849 were normal sessions and 640,618,555 sessions were malignant sessions. Each session has the following twelve features: (1) session length (2) the amount of transmitted data (3) the amount of received data (4) among the sessions in the past 2 seconds, the number of sessions having the same destination IP address as the current session (5) among (4), the percentage of sessions having the same service type as the current session (6) among (4), the percentage of sessions that caused a SYN error (7) among the sessions in the past 2 seconds, the percentage of sessions that caused a SYN error and the service type is the same as the current session (8) among the past 100 sessions with the same destination port, the number of sessions having the same source IP address as the current session (9) among the past 100 sessions with the same destination port, the number of sessions having the same destination IP address as the current session (10) among (8), the percentage of sessions having the same source port as the current session (11) among (8), the percentage of sessions that caused a SYN error (12) among (9), the percentage of sessions that caused a SYN error. We also used these twelve features that were the same as the features used in the previous work by Tada *et al*.

The dataset was divided into two-month units from November 2006 to December 2008 and from November 2013 to December 2015 as shown in Table 1. Then, we evaluated the classifiers as follows. First, 10,000 sessions were randomly selected from each period. After sessions selected in period A were used for training, sessions selected in period B were used for test (called A-B). Similarly, after sessions in period B were used for training, sessions in period C were used for test (called B-C), and the same procedure was repeated until period M in the previous work. In addition to A, ..., M, we evaluated the classifiers for newer periods, N, ..., Z to investigate whether or not gradient boosting tree algorithms and random forests can deal with tendency of newer attacks.

Table 1: Periods from Nov. 2006 to Dec. 2008 and from Nov. 2013 to Dec. 2015, and the corresponding symbols.

Period	Symbol
Nov. 2006 ～ Dec. 2006	A
Jan. 2007 ～ Feb. 2007	B
Mar. 2007 ～ Apr. 2007	C
May 2007 ～ Jun. 2007	D
Jul. 2007 ～ Aug. 2007	E
Sep. 2007 ～ Oct. 2007	F
Nov. 2007 ～ Dec. 2007	G
Jan. 2008 ～ Feb. 2008	H
Mar. 2008 ～ Apr. 2008	I
May 2008 ～ Jun. 2008	J
Jul. 2008 ～ Aug. 2008	K
Sep. 2008 ～ Oct. 2008	L
Nov. 2008 ～ Dec. 2008	M
Nov. 2013 ～ Dec. 2013	N
Jan.2014 ～ Feb. 2014	O
Mar. 2014 ～ Apr. 2014	P
May 2014 ～ Jun. 2014	Q
Jul. 2014 ～ Aug. 2014	R
Sep. 2014 ～ Oct. 2014	S
Nov. 2014 ～ Dec. 2014	T
Jan. 2015 ～ Feb. 2015	U
Mar. 2015 ～ Apr. 2015	V
May 2015 ～ Jun. 2015	W
Jul. 2015 ～ Aug. 2015	X
Sep. 2015 ～ Oct. 2015	Y
Nov. 2015 ～ Dec. 2015	Z

3.2 Implementation

For optimization of our neural network, the number of epochs was set to 100, the softmax cross entropy function and stochastic gradient descent in tensorflow package (https://www.tensorflow.org/) were used. For Friedman's GBT and XGBoost, we used GradientBoostingClassifier in scikit-learn package (http://scikit-learn.org/), and XGBoost python package, respectively. We used fixed parameters as shown in Table 2 during all our experiments for Friedman's GBT and XGBoost, where parameters were previously tuned using grid search for 120,000 sessions randomly selected from period A to M. For random forests, the number of decision trees to be weak learners was fixed to 100, the depth of the decision tree was not fixed, and leaves were expanded until the number of terminal node data was less than two.

Table 2: Parameters, the number of decision trees, depth of trees, and learning rate, in our experiments for Friedman's GBT and XGBoost.

	Friedman's GBT	XGBoost
# decision trees	400	90
Depth of trees	3	6
Learning rate	0.3	0.45

We used a computer with intel core i9 processor 3.6GHz and 64G bytes memory under linux OS in all experiments. Since cyber-attacks often force a large amount of accesses at one time, an NIDS must classify the accesses as accurately and quickly as possible. Our purpose is not only to improve the accuracy but also to reduce the time required for classification.

We calculated accuracy, precision, true positive rate (TPR), and false positive rate (FPR) from classification results as follows.

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

$$Precision = \frac{TP}{TP+FP},$$

$$TPR = \frac{TP}{TP+FN},$$

$$FPR = \frac{FP}{TN+FP},$$

where *TP* denotes the number of sessions that were actually attacks and predicted to be attacks, *FP* denotes the number of sessions that were predicted to be attacks but were actually normal (type 1 error), *TN* denotes the number of sessions that were actually normal and predicted to be normal, and *FN* denotes the number of sessions that were predicted to be normal but were actually attacks (type 2 error).

4 RESULTS

Table 3 shows the results on the accuracy by our deep neural network for each period, where the standard deviation of the accuracy was 0.0458. Table 4 shows the results on the accuracy, precision, TPR, and FPR by random forests, which was best in the previous work by Tada *et al*.

Table 3: Results on the accuracy by our deep neural network for each period, A-B, ..., L-M.

	Acc.
A-B	0.8814
B-C	0.8853
C-D	0.9301
D-E	0.9165
E-F	0.8998
F-G	0.905
G-H	0.8394
H-I	0.8996
I-J	0.7585
J-K	0.9031
K-L	0.9385
L-M	0.8932
Average	0.8875

From the tables, we can see that the accuracy by fully connected neural networks with one hundred middle layers was lower than that by random forests with one hundred decision trees although the accuracy by our neural network was not so low.

Table 4: Results on the accuracy, precision, TPR, and FPR for each period, A-B, ..., L-M, by random forests, which was best in the previous work by Tada *et al*.

	Acc.	Pre.	TPR	FPR
A-B	0.9399	0.9801	0.898	0.0182
B-C	0.9647	0.9416	0.991	0.0615
C-D	0.9083	0.9722	0.8405	0.0239
D-E	0.9253	0.9469	0.901	0.0504
E-F	0.9747	0.9635	0.9867	0.0374
F-G	0.9848	0.9811	0.9886	0.019
G-H	0.9221	0.8794	0.9794	0.1352
H-I	0.9762	0.9846	0.9676	0.0151
I-J	0.8145	0.7365	0.9892	0.3602
J-K	0.9874	0.9857	0.9891	0.0144
K-L	0.9885	0.9891	0.988	0.0109
L-M	0.9752	0.9747	0.9757	0.0253
Ave.	0.9468	0.9446	0.9579	0.0643

Tables 5 and 6 show the results on the accuracy, precision, TPR, and FPR by Friedman's GBT and XGBoost, respectively, for each period, A-B, ..., L-M. The average accuracy, precision, and TPR by the GBT methods were higher than those by RF, which had the highest classification accuracy in the previous study by Tada *et al*. In addition, the average false positive rate (FPR) by the GBT methods was smaller than that by RF. Especially, although FPRs in G-H and I-J were relatively high, XGBoost could reduce FPRs in both periods. In period C-D, the accuracy 0.9852 by Friedman's GBT increased from that by RF. The accuracy 0.9898 by XGBoost in period K-L was the highest, and the accuracy by Friedman's GBT and RF in the same period was also high. It is considered that features in sessions between periods K and L were very similar. In another experiment, by adding source and destination port numbers as features of a session, the average accuracy by XGBoost achieved 0.9879.

Moreover, we measured the elapsed time for classification of one session in A-B,..., L-M. The average elapsed time by RF, Friedman's GBT, and XGBoost, was 0.113 ± 0.0201, 0.0671 ± 0.0106, and 0.0192 ± 0.00369 seconds, respectively (also shown in Fig. 2). The average elapsed time by XGBoost was the shortest, and XGBoost could do the classification more than five times faster than RF. These results mean that XGBoost can improve the detection

accuracy of cyber-attacks and reduce the time required for classification.

Table 5: Results on the accuracy, precision, TPR, and FPR by Friedman's GBT for each period, A-B, ..., L-M.

	Acc.	Pre.	TPR	FPR
A-B	0.9399	0.9801	0.898	0.0182
B-C	0.9651	0.9907	0.9912	0.0609
C-D	0.9852	0.9862	0.9852	0.0158
D-E	0.9394	0.9335	0.9323	0.0526
E-F	0.9755	0.9859	0.9862	0.0352
F-G	0.9868	0.9913	0.9924	0.0178
G-H	0.9187	0.9916	0.9929	0.1555
H-I	0.9772	0.9697	0.9682	0.0148
I-J	0.8603	0.8658	0.9896	0.269
J-K	0.9886	0.9895	0.988	0.0123
K-L	0.9896	0.9886	0.9901	0.0094
L-M	0.9779	0.974	0.9738	0.0179
Ave.	0.9597	0.9661	0.9761	0.0566

Table 6: Results on the accuracy, precision, TPR, and FPR by XGBoost for each period, A-B, ..., L-M.

	Acc.	Pre.	TPR	FPR
A-B	0.9518	0.9885	0.9094	0.0105
B-C	0.965	0.9413	0.992	0.0619
C-D	0.9875	0.9856	0.9872	0.0144
D-E	0.9395	0.9546	0.9187	0.0436
E-F	0.9771	0.9672	0.9884	0.0335
F-G	0.9865	0.9829	0.9902	0.0172
G-H	0.935	0.8901	0.9935	0.1236
H-I	0.9786	0.9847	0.9724	0.0152
I-J	0.8686	0.7982	0.9899	0.2526
J-K	0.9892	0.9873	0.9911	0.0127
K-L	0.9898	0.991	0.9886	0.0089
L-M	0.9782	0.9824	0.9739	0.0175
Ave.	0.9622	0.9545	0.9746	0.051

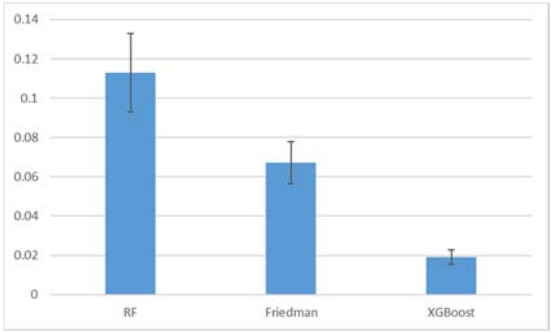

Figure 2: The average elapsed time (seconds) for classification of one session by random forest (RF), Friedman's GBT, and XGBoost.

Tables 7, 8, and 9 show the results on the accuracy, precision, TPR, and FPR by RF, Friedman's GBT, and XGBoost, respectively, for each period, N-O, ..., Y-Z. Although the average accuracy by XGBoost was slightly higher than the others, the difference of the average accuracy was very small. The average precision, TPR, and FPR, respectively, were almost the same in the classifiers.

The average elapsed time for classification of one session in N-O, ..., Y-Z by RF, Friedman's GBT, and XGBoost was 0.130 ± 0.0245, 0.0634 ± 0.0103, and 0.0188 ± 0.00356 seconds, respectively. Similarly to periods, A-B, ..., L-M, the elapsed time by XGBoost was the shortest also in N-O, ..., Y-Z.

Table 7: Results on the accuracy, precision, TPR, and FPR by random forests for each period, N-O, ..., Y-Z.

	Acc.	Pre.	TPR	FPR
N-O	0.9606	0.9619	0.9594	0.0381
O-P	0.9374	0.916	0.9634	0.0885
P-Q	0.9486	0.9539	0.9427	0.0455
Q-R	0.708	0.6399	0.9548	0.5372
R-S	0.9244	0.9062	0.9469	0.0982
S-T	0.9293	0.9214	0.9387	0.0801
T-U	0.9303	0.9224	0.9396	0.0789
U-V	0.8165	0.7425	0.9692	0.3361
V-W	0.9072	0.8961	0.9212	0.1067
W-X	0.8915	0.8648	0.9281	0.1451
X-Y	0.8424	0.7832	0.9472	0.2622
Y-Z	0.9049	0.8813	0.936	0.1261
Ave.	0.8918	0.8658	0.9456	0.1619

Table 8: Results on the accuracy, precision, TPR, and FPR by Friedman's GBT for each period, N-O, ..., Y-Z.

	Acc.	Pre.	TPR	FPR
N-O	0.9586	0.9611	0.956	0.0387
O-P	0.9366	0.913	0.9653	0.092
P-Q	0.9479	0.9556	0.9395	0.0437
Q-R	0.7098	0.6409	0.9543	0.5348
R-S	0.9267	0.9231	0.931	0.0776
S-T	0.9312	0.9291	0.9338	0.0712
T-U	0.9311	0.9257	0.9376	0.0752
U-V	0.8178	0.7431	0.9715	0.3358
V-W	0.8965	0.8724	0.9291	0.1361
W-X	0.8866	0.9498	0.9393	0.1661
X-Y	0.8379	0.7785	0.9449	0.2689
Y-Z	0.9022	0.8777	0.935	0.1305
Ave.	0.8902	0.8725	0.9448	0.1642

Table 9: Results on the accuracy, precision, TPR, and FPR by XGBoost for each period, N-O, ..., Y-Z.

	Acc.	Pre.	TPR	FPR
N-O	0.9573	0.96	0.9544	0.0399
O-P	0.9372	0.9146	0.9644	0.0901
P-Q	0.9485	0.9545	0.942	0.0449
Q-R	0.7134	0.6435	0.9568	0.5301
R-S	0.9265	0.9198	0.9344	0.0816
S-T	0.9321	0.933	0.9309	0.0668
T-U	0.9306	0.9224	0.9402	0.079
U-V	0.8196	0.7448	0.9726	0.3334
V-W	0.8992	0.8761	0.9299	0.1316
W-X	0.8802	0.8407	0.9384	0.178
X-Y	0.8487	0.7929	0.9438	0.2464
Y-Z	0.9093	0.8933	0.9296	0.1111
Ave.	0.8919	0.8663	0.9448	0.1611

5 CONCLUSIONS

We first constructed a fully connected neural network with one hundred middle layers for detecting cyber-attacks. Although it is considered that the neural network had a sufficient number of parameters and significant flexibility, the classification accuracy by the neural network was not higher than that by RF, which was the best in the previous work. We examined two gradient boosting tree (GBT) methods, Friedman's GBT and XGBoost, that are known to have higher regularization performance, and we performed computational experiments on newly constructed Kyoto 2016 dataset. The results suggest that XGBoost outperforms other machine learning classifiers including RF. Furthermore, the elapsed time required for classification by GBTs was much shorter than those by RF. It is important because an NIDS must classify a large amount of accesses immediately. As future work, we would like to model normal accesses for automatically recognizing new unknown threats.

ACKNOWLEDGEMENTS

This work was partially supported by National Institute of Technology, Japan.

REFERENCES

Ambusaidi, M. A., He, X., Nanda, P. and Tan, Z. (2016). Building an intrusion detection system using a filter-based feature selection algorithm. *IEEE Trans. Computers*, 65(10): 2986-2998.

Amor, N. B., Benfarhat, S. and Elouedi, Z. (2004). Naïve Bayes vs decision trees in intrusion detection systems. Proc. 2004 ACM Symposium on Applied Computing, 420-424.

Candora, V., Banerjee, A. and Kumar, V. (2009). Anomaly detection: a survey. ACM Comput. Surv., 41(3):15:1-15:58.

Dhaliwal, S. S., Nahid, A. A. and Abbas, R. (2018). Effective intrusion detection system using XGBoost. Information, 9(7):149.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. Application of Data Mining in Computer Security, 77-101.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. The Annuals of Statistics, 29(5):1189-1232.

Hosseini, Z. S., Chabok, S. J. S. M. and Kamel, S. R. (2015). DOS intrusion attack detection by using of improved SVR. Proc. 2015 International Congress on Technology, Communication and Knowledge, 159-164.

Kokila, R. T., Selvi, S. T. and Govindarajan, K. (2014). DDoS detection and analysis in SDN-based environment using support vector machine classifier. Proc. 6th International Conference on Advanced Computing, 205-210.

Masarat, S., Sharifian, S., and Taheri, H. (2016). Modified parallel random forest for intrusion detection systems. J. Supercomput. 72(6): 2235-2258.

Mukkamala, S., Janoski, G. and Sung, A. (2002). Intrusion detection using neural networks and support vector machines. Proc. 2002 International Joint Conference on Neural Networks, 2:1702-1707.

Nadeem, M., Marshall, O., Singh, S., Fang, X. and Yuan X. (2016). Semi-supervised deep neural network for network intrusion detection. KSU Proc. Cybersecurity Education, Research and Practice, 2.

Om, H. and Kundu, A. (2012). A hybrid system for reducing the false alarm rate of anomaly intrusion detection system. Proc. 1st International Conference on Recent Advances in Information Technology, 131-136.

Potluri, S. and Diedrich, C. (2016). Accelerated deep neural networks for enhanced intrusion detection system. Proc. 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation, 1-8.

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D. and Nakao, K. (2011). Statistical analysis of honeypot data and building of Kyoto 2006+dataset for NIDS evaluation. Proc. 1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, 29-36.

Stein, G., Chen, B., Wn, A. S. and Hua, K. A. (2005). Decision tree classifier for network intrusion detection with GA-based feature selection. Proc. 43rd Annual Southeast Regional Conference, 136-141.

Tada, R., Kobayashi, R., Shimada, H. and Takakura, H. (2017). Generating Kyoto 2016 dataset for NIDS

evaluation. Trans. Inf. Process. Soc. Japan, 58(9): 1450-1463.

Tianqi, C. & Carlos, G. (2016). XGBoost: A scalable tree boosting system. Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.

Yin, C., Zhu, Y., Fei, J., and He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 5:21954-21961.