IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ARTIFICIAL INTELLIGENCE IN CYBERSECURITY

Received May 9, 2019, accepted June 7, 2019, date of publication June 19, 2019, date of current version July 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2923640

An Adaptive Ensemble Machine Learning Model for Intrusion Detection

XIANWEI GAO, CHUN SHAN, CHANGZHEN HU, ZEQUN NIU, AND ZHEN LIU

Beijing Key Laboratory of Software Security Engineering Technology, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Chun Shan (sherryshan@bit.edu.cn)

This work was supported in part by the National Key R&D Program of China under Grant 2016YFB0800700, and in part by the National Natural Science Foundation of China under Grant U1636115.

ABSTRACT In recent years, advanced threat attacks are increasing, but the traditional network intrusion detection system based on feature filtering has some drawbacks which make it difficult to find new attacks in time. This paper takes NSL-KDD data set as the research object, analyses the latest progress and existing problems in the field of intrusion detection technology, and proposes an adaptive ensemble learning model. By adjusting the proportion of training data and setting up multiple decision trees, we construct a MultiTree algorithm. In order to improve the overall detection effect, we choose several base classifiers, including decision tree, random forest, kNN, DNN, and design an ensemble adaptive voting algorithm. We use NSL-KDD Test+ to verify our approach, the accuracy of the MultiTree algorithm is 84.2%, while the final accuracy of the adaptive voting algorithm reaches 85.2%. Compared with other research papers, it is proved that our ensemble model effectively improves detection accuracy. In addition, through the analysis of data, it is found that the quality of data features is an important factor to determine the detection effect. In the future, we should optimize the feature selection and preprocessing of intrusion detection data to achieve better results.

INDEX TERMS Intrusion detection, ensemble learning, deep neural network, voting, MultiTree, NSL-KDD.

I. INTRODUCTION

As the main approach to defend advanced threat attacks, network intrusion detection is facing more and more challenges. The traditional intrusion detection system based on feature detection has been used for a long time. Be limited by the scale and refresh rate of the database of predefined signatures, signature based intrusion detection system is not able to detect all types of attacks especially new attack variants. In order to solve this problem, researchers have paid much attention to introduce other techniques in intrusion detection, and one way is to use machine learning techniques.

In recent years, Decision tree, random forest, SVM, neural network and other machine learning algorithms have been used in the field of intrusion detection, and some improvements have been achieved.

However, as it is well known that there is no free lunch, each algorithm has its own advantages and disadvantages. Some algorithms may perform well on one type of attack, but show poor performance on other types. Through the analysis of some past research papers, no matter what the deep learning or feature selection methods is, there are still some disadvantages. In addition, many studies only focus on the overall detection accuracy, but the detection effect for small-scale data is often very low. The proportion of real attack events in all data is imbalanced, so we need to focus on the detection ability of malicious attack data with small proportion.

The present paper proposes an adaptive ensemble learning model, which can integrate the advantages of each algorithm for different types of data detection, and achieve optimal results through ensemble learning. The advantage of ensemble learning is to combine the predictions of several base estimators in order to improve generalizability and robustness over a single estimator. This paper uses NSL-KDD data set and some common algorithms such as decision tree, random forest and deep neural network to train our model. The MultiTree and adaptive voting algorithm are proposed which obviously improve the effect of intrusion detection. By comparison, they are superior to many previous research results and have good application prospects.

The associate editor coordinating the review of this manuscript and approving it for publication was Wei Quan.

II. RELATED WORK

In the field of intrusion detection, many scholars have tried machine learning algorithms and used public NSL-KDD data set for research in order to improve the detection effect [9]. Hodo *et al.* [4] reviewed machine learning techniques and their performance in detecting anomalies. Feature selection which influences the effectiveness of machine learning (ML) IDS is discussed to explain the role of feature selection in the classification and training phase of ML IDS. Nathan Shone presented a novel deep learning technique for intrusion detection [5], which addressed some concerns and proposed nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning. Tao *et al.* [7] proposed a novel approach called SCDNN, which combines spectral clustering (SC) and deep neural network (DNN) algorithms. KEHE WU *et al.* [8] used CNN to select traffic features from raw data set automatically, and set the cost function weight coefficient of each class based on its numbers to solve the imbalanced data set problem. Simone [23] designed IDS with a neural network ensemble method to classify the different attacks. The neural network ensemble method comprises autoencoder, deep belief neural network, deep neural network, and an extreme learning machine. To improve the performance of network intrusion detection systems, Xu C applied deep learning theory to intrusion detection and developed a deep network model with automatic feature extraction [14]. Deep learning techniques were used for training and achieved good performance.

Several studies have suggested that by selecting relevant features for intrusion detection system, it is possible to considerably improve the detection accuracy and performance of the detection engine [10]. Shrivas [13] proposed ANN-Bayesian Net-GR technique that means ensemble of Artificial Neural Network (ANN) and Bayesian Net with Gain Ratio (GR) feature selection technique. Ambusaidi *et al.* [17] proposed a mutual information based algorithm that analytically selects the optimal feature for classification. This mutual information based feature selection algorithm can handle linearly and nonlinearly dependent data features. MAJJED AL-QATF [18] proposed an effective deep learning approach, self-taught learning (STL)-IDS, based on the STL framework. The proposed approach is used for feature learning and dimensionality reduction. It reduces training and testing time considerably and effectively improves the prediction accuracy of support vector machines (SVM) with regard to attacks. Jaber *et al.* [30] used principal component analysis and linear discriminant analysis with a hybrid, nature-inspired metaheuristic algorithm called Ant Lion optimization for feature selection and artificial neural networks to classify and configure the cloud server, in order to prevent DDoS attack in cloud computing.

Majd *et al.* [12] proposed a 5-level hybrid classification system based on flow statistics in order to attain an improvement in the overall accuracy of the system. For the first level, they employ the k-Nearest Neighbor approach (kNN); for the second level, use the Extreme Learning Machine (ELM). To prevent the cyberattack irreversible damage, Zhou *et al.* [19] proposed a framework, called DFEL, to detect the internet intrusion in the IoT environment. Through the experimental results, authors presented that DFEL not only boosts classifiers' accuracy to predict cyber attack but also significantly reduce the detection time. Wahba *et al.* [22] applied adaptive boosting using naïve Bayes as the weak (base) classifier. The key point in the research is that they are able to improve the detection accuracy with a reduced number of features while precisely determining the attack. Zhao *et al.* [25] proposed a transfer learning-enabled framework and approach, called HeTL, which can find the common latent subspace of two different attacks and learn an optimized representation, which was invariant to attack behaviors' changes.

Generally speaking, previous studies mainly focus on the optimization of neural networks and some machine learning algorithms to improve the overall detection effect. The main optimization methods are feature selection and ensemble learning. However, there is still much room to improve the results of these studies.

III. PROPOSED APPROACH

A. ADAPTIVE ENSEMBLE LEARNING MODEL

The adaptive ensemble learning model designed in this paper chooses common machine learning algorithms such as decision tree, SVM(support vector machines), logical regression, kNN(k-nearest neighbors) [27], Adaboost, random forest and deep neural network as alternative classifiers. Five voting classifiers are selected through comparative tests. Then, by adjusting the proportion of samples, setting data weights, multi-layer detection and other combined method to boost the detection effect of each algorithm. Finally, the adaptive voting algorithm with different class-weights is used to obtain the optimal detection results.

The adaptive ensemble learning model showed in Figure 1 mainly includes the following processes:

1) Input the NSL-KDD training data set.

2) The preprocessing module converts the character-type features such as label and service into numbers, standardizes the data, and deletes unnecessary features.

3) Ensemble training of candidate algorithms using preprocessed data.

4) All the algorithms are trained by cross validation using training data, and the algorithm with better detection accuracy and operation performance is selected to be voted, and each algorithm is boosted to further improve the detection accuracy. The boosting methods include feature selection, unbalanced sampling, class weight, multi-layer detection, etc. In this paper, the decision tree algorithm is optimized and the MultiTree algorithm is proposed.

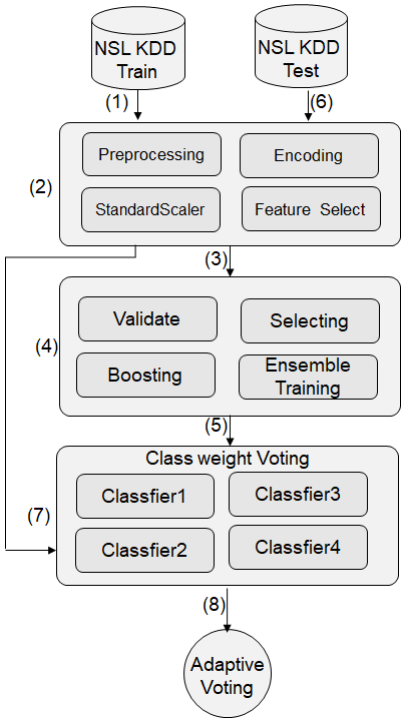

FIGURE 1. Adaptive ensemble learning model.

5) According to the training accuracy of each algorithm, the classification weights of each algorithm are set and the adaptive voting algorithm model is generated.

6) Input the data of the whole NSL-KDD test set and preprocess it in step 2.

7) Each algorithm selected is used to detect the test set, output the preliminary predict classification, and then calculate the final voting results using the adaptive voting algorithm.

B. DATASET INTRODUCTION

The famous public KDD99 [1] data set has two important issues which highly affect the performance of evaluated systems. One of the most important deficiencies is the huge number of redundant records [2] which causes the learning algorithms to be biased towards the frequent records, and thus preventing them from learning fewer records which are usually more harmful to networks such as U2R and R2L attacks. In addition, the existence of these repeated records in the test set often causes the evaluation results to be biased by the methods which have better detection rates on the frequent records. Tavallaee *et al.* [3] proposed a new data set NSL-KDD, which consists of selected records of the complete KDD data set but does not suffer from any of mentioned shortcomings. Table 1 shows the data set contains five types of data, including Normal, DOS, Probe, R2L, and U2R.

The analysis by Revathi [6] shows that NSL-KDD data set is very ideal for comparing different intrusion detection models. Therefore, this paper chooses the data set as the research object.

TABLE 1. NSL-KDD introduction.

Class	Subclass	Description
Normal	Normal	benign record
DOS	apache2,back,mailbomb,processtable,snmpgetattack,teardrop,smurf,land,Neptune,pod,udpstorm	denial-of-service, e.g. syn flood;
Probe	Nmap, Ipsweep, Portsweep, Satan, Mscan, Saint	surveillance and other probing, e.g., port scanning.
R2L	ftp_write,guess_passwd,snmpguess,imap,spy,warezclient,warezmaster,multihop,phf,imap,named,sendmail,xlock, xsnoop,worm,	unauthorized access from a remote machine, e.g. guessing password;
U2R	Ps,buffer_overflow,perl,rootkit,loadmodule,xterm,sqlattack,httptunnel,	unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks;

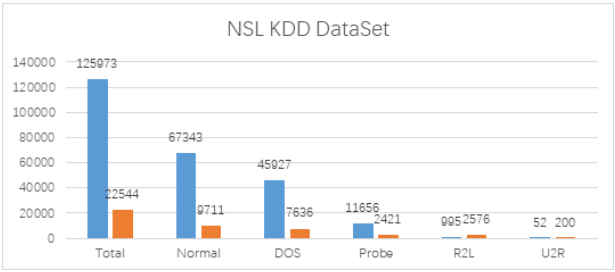

FIGURE 2. Proportional statistics of NSL-KDD train and test.

According to the statistical analysis of various types of data in Figure2, it is found that the distribution of various types of data is imbalanced. Normal has the highest proportion of total data, accounting for about 53% of the total data, while U2R type sample data is very few, which may lead to inadequate training and under-fitting. In fact, Probe, R2L and U2R are often used by hackers as advanced threat attack, so we should try our best to improve the classification accuracy of these types of data.

C. DATA PREPROCESSING

There are 42 fields in the original data set, among which label, protocol and flag fields are character types. Those fields cannot be directly used as input of machine learning algorithm, so preprocessing operations are needed. Firstly, label tag columns in the original data are converted into five types, Normal: 0, DOS: 1, Probe: 2, R2L: 3, U2R: 4. The protocol_type field takes TCP, UDP and ICMP as its values. We use one-Hot-Encoding to process its text values, which converts all classification features into binary ones, such as [1,0,0] representing TCP protocol. After transformation, 122 data features are included. Through the analysis of training set data, it is found that num_outbound_cmds data value

is 0, so this feature is removed. In the original data, the range of values of many feature fields varies greatly, which has a great impact on the training results. Therefore, Standard Scaler method is used to standardize the data. Standardized data is transformed by subtracting the mean and dividing it by variance (or standard deviation). The normalized data conforms to the standard normal distribution, that is, the mean is 0 and the standard deviation is 1.

In order to understand the distribution of data intuitively, we reduce the dimensions of NSL-KDD from 42 to 2, and hope the 2 dimensions data set represents the original data set as much as possible. Principal component analysis (PCA) is used to find out the most important aspect of the data. We can see the distribution of data sets intuitively on a two-dimensional graph.

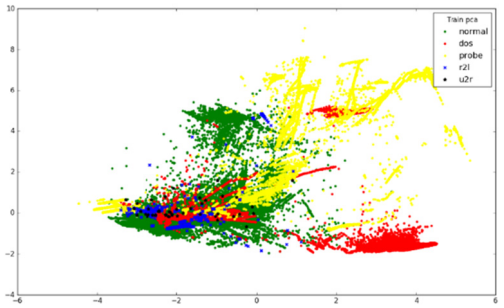

FIGURE 3. PCA analysis of NSL-KDD training data set.

As can be seen from the PCA chart of the training data set in Figure 3, the data of Normal (green), DOS (red), Probe (yellow) are relatively centralized, and the overlap with other types of data is not much, so they are easy to be classified. The R2L (blue) and U2R (black) data overlap with other data, and they are not centralized enough, so it is difficult to classify. However, the data of the two types are relatively small, and they are sensitive to the detection results, which can easily lead to a large number of false positives. After the training data set is preprocessed, machine learning algorithm can be used for training. The test set uses the data in the KDDTest+ file and uses the same pre-processing measurements as the training set. Classification algorithm should not only consider the overall detection accuracy, but also try to improve the accuracy of small ratio data.

D. MULTI TREE ALGORITHM

According to the previous research, we choose the CART (Classification and Regression Tree) to classify the sample data. In the classification problem, if there are K classes and the probability of sample points belonging to k class is p_k, then the Gini exponent of probability distribution is defined as follows: $\text{Gini(p)} = 1 - \sum_1^k p_k^2$, for a given sample set D, its Gini index is: $\text{Gini(D)} = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|}D|\right)^2$, C_k is the sample subset of class K in D, and K is the number of classes. Gini index is the difference between the sum of probability squares of class C_K and 1, which reflects the uncertainty of sample set. The sample set corresponding to the parent node

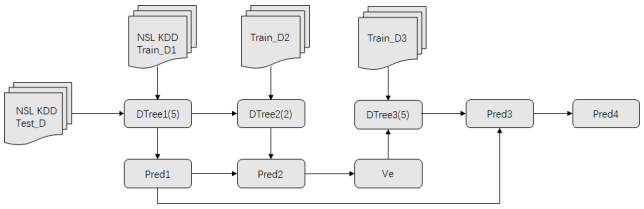

FIGURE 4. MultiTree algorithm process.

is D, and CART chooses feature A to split into two sub-nodes, corresponding set is D1 and D2. The split Gini index is defined as follows:

$$\text{Gini(D, A)} = \frac{|D1|}{|D|}Gini\,(D1) + \frac{|D2|}{|D|}Gini(D2)$$

The greater the Gini index, the higher the uncertainty of the sample set. The essence of classification learning process is the reduction of sample uncertainty (i.e. the process of entropy reduction), so the feature splitting of the minimum Gini index should be chosen.

Due to the serious imbalance in the proportion of various types of data in the data set, Normal and DOS types account for a higher proportion of data, resulting in higher accuracy of these types, while the accuracy of U2R type is lower. In this paper, an ensemble algorithm is designed to train several CART classifiers by adjusting the proportion of different types of samples to solve the imbalance of sample proportion.

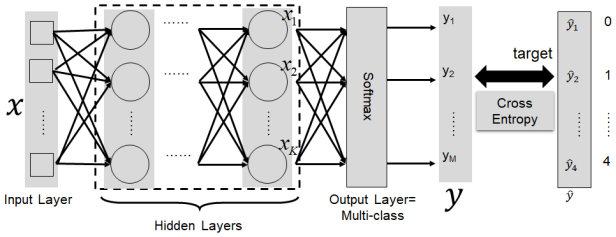

FIGURE 5. Deep neural network.

E. DEEP NEURAL NETWORK ALGORITHM

This paper also takes DNN as the base classifier algorithm and improves its detection effect. The structure of DNN designed in this paper is shown in Figure 5. The deep neural network consists of input layer, hidden layer and output layer. Layers are fully connected with each other. Any neuron in layer i must be connected with any neuron in layer i + 1. Although DNN seems complex, it is the same as perceptron in small local models, that is, a linear relationship $z = \sum w_i x_i +$ b with an activation function $\sigma(z)$. The forward propagation algorithm of DNN uses several weight coefficient matrices w, bias vector b and input value vector x to carry out a series of linear and activation operations, starting from the input layer, backward calculation from one layer to the output layer.

In multi-classification scenarios, we use ReLU (Rectified Linear Unit) as the activation function:

$$\sigma\,(x) = \max(0, W^T + b)$$

Algorithm 1 MultiTree Algorithm

Input: Training Set Data Train_D, Test_D
Output: Output integrated test results
Process:
 # step 1 Training the first decision tree
rus = RandomUnderSampler(ratio = {1/16∗r0,r1,r2,r3,r4})
 # Reduce the ratio of Normal types to 1/16 to solve the problem of sample imbalance.
Train_rus = rus.fit_sample(Train_D)
Dtree1.fit(Train_rus)
pred_y1 = Dtree1.predict(Test_D)
 # step 2 Training Normal and Abnormal Decision Trees
ConvertBinary(Train_D, Test_D)
 # Converting training set and test set into two categories.
rus = RandomUnderSampler(ratio = {1/8∗r0,r1})
 # Adjusting normal/abnormal sample ratio.
Train_rus2 = rus.fit_sample(Train_D)
DTree2.fit(Train_rus2)
y_normal = DTree2.predict(Test_D)
error_index = list(pred_y1 == 0).isin (list(y_normal == 1))
 # Records with inconsistent statistical classifications. Records identified as 0 in the first step and 1 in the second step.
# step 3:*The data identified as Normal in Step 1 and abnormal in Step 2 are further classified into (1,2,3,4) categories.*
rus = RandomUnderSampler(ratio = {r1,r2,r3,r4})
 # Training attack type decision tree only
Train_rus3 = rus.fit_sample(Train_D)
DTree3.fit(Train_rus3)
error_y_ = clf_evil.predict(Test_D[error_index])
pred_y1[error_index] = error_y_
 # Re-assignment of appraisal results
return pred_y1

A loss function should be used to measure the output loss of training samples, and then the loss function is optimized to minimize the extreme value. We use cross-entropy as loss function:

$$C = -\frac{1}{n}\sum_{x}[ylny + (1-y)\ln(1-y)]$$

A series of linear coefficient matrices W and bias vector b are our final training results. After designing the DNN model, we adjust the weights of the training data, set class_weight = 0:1, 1:2, 2:20, 3:40, 4:200}, and increase the weight of the type data with small sample proportion, so as to improve the detection effect of this type. The class weight will be helpful to the detection effect. Finally, we use the softmax function

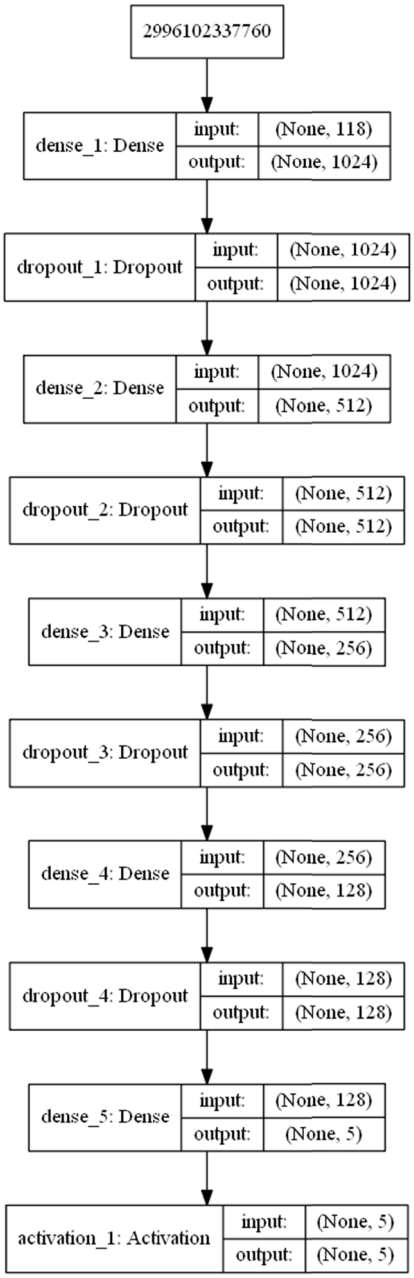

FIGURE 6. DNN model parameter.

to decide which class the record should be. The network structure and parameters of DNN can be seen in Figure 6.

F. ADAPTIVE VOTING

In order to synthesize the advantages of each algorithm, we propose an adaptive voting algorithm. The algorithm showed in Figure 7 trains different classifiers (weak classifiers) for the same training set, and then assemble these weak classifiers to form a stronger final classifier. The core thoughts of the algorithm is to determine the weight w_{ij} of a classifier algorithm for a certain type of data which expresses the credibility (possibility) of obtaining the detection value in this scenario. Because the characteristics of each data set

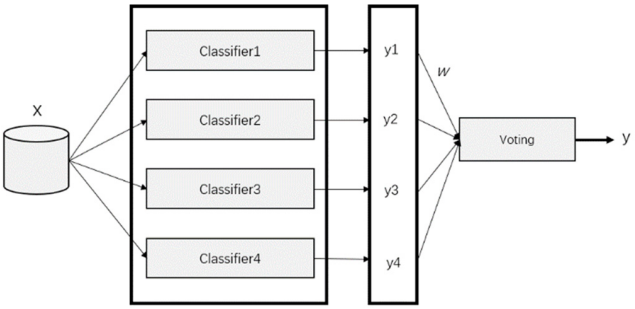

FIGURE 7. Adaptive voting algorithm.

are different, the voting weight can also be set manually by referring to the weight value to achieve the best results.

The adaptive voting algorithm is as follows:

Algorithm 2 Adaptive Voting Algorithm

Input:

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}, x_i \in X, y_i \in Y$$
$$Y = \{1, .., c\}; F = \{f_1, f_2, \cdots, f_m\};$$

Out:H(x)

1) Initialize the class weights:

$$w = \begin{bmatrix} w_{11} & \cdots & w_{1c} \\ \cdots & \cdots & \cdots \\ w_{m1} & \cdots & w_{mc} \end{bmatrix}$$

$w_{ij} = 1; i = \{1,\ldots,m\}; j = \{1,..,c\}$

2) For i from 1 to m

(a) Fit a classifier $f_i(x)$ to the training data

(b) Calculate possibility for f_i $(x|c = j)$, $w_{ij} = \frac{\sum_0^c (f_i(x)==\hat{y})\&\&(y==j)}{N}$

3) Using all classifiers to predict: classifier.predict(Test_x),

4) Calculate possibility for one record to belong class c, $p_i = \sum_1^m w_{ij}(f_i(x) = c)$

5) Output:

$$H(x) = argmax\left(\sum_{i=1}^{i=c} p_i(f_i(x) == \hat{y}\right)$$

Algorithmic description:

1) Optimize the machine learning algorithms (classifiers) in F, then use training sets and verification sets to train and evaluate them.

2) Calculate the training accuracy of each algorithm for different attack types as the weight cardinality w_{ij}.

3) For each test record, the predictive results of each classifier are calculated according to the [0-4] type.

4) Choose the class with the max voting result as the final predict result of the record.

5) Output full five-category test results.

An example is given to illustrate the working principle of the weighted voting algorithm.

As can be seen in Table 2, Classifier 1 has the best detection effect for Type 3, while Classifier 3 has the best detection effect for Type 1. After setting the weights of each classifier, we can start voting for test records.

By adaptive voting, the highest possible classification test results in Table 3 were calculated.

TABLE 2. The class-weights of three classifier.

Class Weight	Classifier1	Classifier2	Classifier3
Class1	w11=0.8	w12=0.5	w13=0.7
Class2	w21=0.7	w22=0.8	w23=0.9
Class3	w31=0.9	w32=0.6	w33=0.5

TABLE 3. Examples of voting results.

Predict	Classifier1	Classifier2	Classifier3	Voting	Result
Record1	Class2	Class 2	Class1	0.7+0.8 > 0.7	Class2
Record2	Class1	Class1	Class1	0.8+0.5+0.7	Class1
Record3	Class3	Class2	Class2	0.9<0.8 + 0.9	Class2
Record4	Class3	Class2	Class1	0.7<0.8<0.9	Class3

TABLE 4. Confusion matrix.

Attack		Predicted Class	
		Yes	No
True Class	Yes	TP	FN
	No	FP	TN

IV. EVALUATION

A. EVALUATION METRICS

All these evaluation metrics [5] are basically derived from the four basic attributes of the confusion matrix depicting the actual and predicted classes in Table 4.

1) True Positive (TP) - Attack data that is correctly classified as an attack.

2) False Positive (FP) - Normal data that is incorrectly classified as an attack.

3) True Negative (TN) - Normal data that is correctly classified as normal.

4) False Negative (FN) - Attack data that is incorrectly classified as normal.

We will use the following measures to evaluate the performance of our proposed solution:

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

The accuracy measures the proportion of the total number of correct classifications.

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

The precision measures the number of correct classifications penalized by the number of incorrect classifications.

$$\text{Recall} = \frac{TP}{TP + FN}$$

The recall(also called detection rate) measures the number of correct classifications penalized by the number of missed entries.

$$\mathrm{F1}-\mathrm{score}=\frac{2\cdot\mathrm{Precision}\cdot\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}$$

The F1-score measures the harmonic mean of precision and recall, which serves as a derived effectiveness measurement.

B. TEST EVALUATION

This section lists the experimentation process and results obtained. All the tasks are performed using the Python with scikit-learn and tensorflow library on Win10 system. The test computer is equipped with Intel(R) Core i7 CPU 1.8GHz and 8.0GB RAM.

1) TEST EVALUATION

Firstly, several selected machine learning algorithms are used to cross-validate and evaluate the indicators of each algorithm. The original training set is divided into training set and verification set according to the ratio of 50:50.

TABLE 5. Cross-validate on training set.

Algorithms	Accuracy	Precision	Recall	F1	Time(s)
DeciTree	99.63%	99.62%	99.62%	99.62%	0.33
RanForest	99.8%	99.79%	99.8%	99.79%	0.7
kNN	99.59%	99.57%	99.59%	99.58%	33
LR	97.73%	97.72%	97.73%	97.71%	13.7
SVM	99.53%	99.52%	99.53%	99.52%	220.7
DNN	98.4%	99.09%	98.4%	98.64%	245.2
Adaboost	99.9%	99.9%	99.9%	99.89%	112.3

According to the results of cross-validation in Table 5, the training effect of Adaboost algorithm is the best. Then the test data set is used to validate the algorithms and evaluate the generalization effect in Table 6.

TABLE 6. Result of each algorithm on KDDTest+.

Algorithms	Accuracy	Precision	Recall	F1	Time(S)
DeciTree	79.71%	83.51%	79.72%	77.31%	6.34
RanForest	76.64%	81.85%	76.64%	72.17%	1.86
kNN	75.51%	80.97%	75.51%	71.41%	86.49
LR	73.58%	74.65%	73.58%	69.13%	43.77
SVM	74.09%	80.91%	74.09%	70.38%	1785.2
DNN	81.6%	84%	81.6%	80.18%	227.8
Adaboost	76.02%	81.82%	76.02	72.12%	265.1

In the field of intrusion detection, there is a high real-time requirement for the analysis of big data of security logs, which requires not only high accuracy, but also as short detection time as possible. Through comparison, it is found that the overall accuracy of DNN and decision tree is higher, and the running time of decision tree is shorter, which belongs to the best cost-effective learning algorithm. We also need to evaluate the detection effect of each algorithm for different types of data.

TABLE 7. Recall of algorithms detects on each class of KDDTest+.

Algorithms	Normal	Dos	Probe	R2L	U2R
DeciTree	97.21%	84.77%	66.25%	21.37%	7.00%
R.Forest	97.37%	81.47%	68.86%	2.73%	1.50%
kNN	97.27%	79.38%	62.21%	5.16%	4.50%
LR	92.81%	79.81%	64.93%	1.56%	3.00%
SVM	97.65%	77.40%	48.95%	9.38%	2.00%
DNN	95.92%	84.56%	77.53%	30.63%	26.50%

By comparison in Table 7, the detection effect of R2L and U2R data is poor, which is closely related to the imbalance of sample proportion in the previous analysis. If we need to improve the overall detection effect, we must find ways to overcome these problems. Although the overall detection effect of DNN is better, it is worse than other types in Normal type detection. As can be seen in Table 7, some other algorithms perform well on Normal type, but not on the others. Therefore, various classification algorithms do not have advantages in all types of data, but each has its own advantages. In the future, we will optimize the combination of the algorithms and make use of the advantages of various algorithms to improve the overall detection effect.

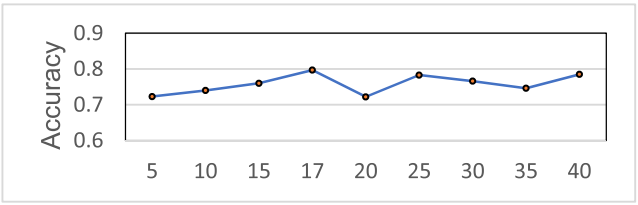

FIGURE 8. Accuracy of decision tree using different number of features.

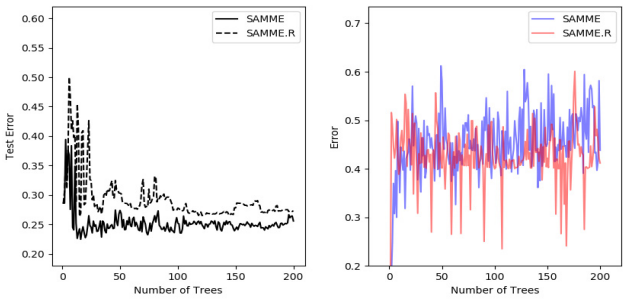

FIGURE 9. Adaboost SAMME.R test results.

2) MULTITREE RESULT

Many researchers use feature selection method to improve the effect of decision tree. The feature with the maximum value of Gini is selected as the partitioning feature. We use CART (Classification and Regression Tree) algorithm to test the accuracy of different number of features in Figure 8, and find that 17 features get the best accuracy of 79.7% on NSL-KDD Test+. We tested the four-level decision tree and

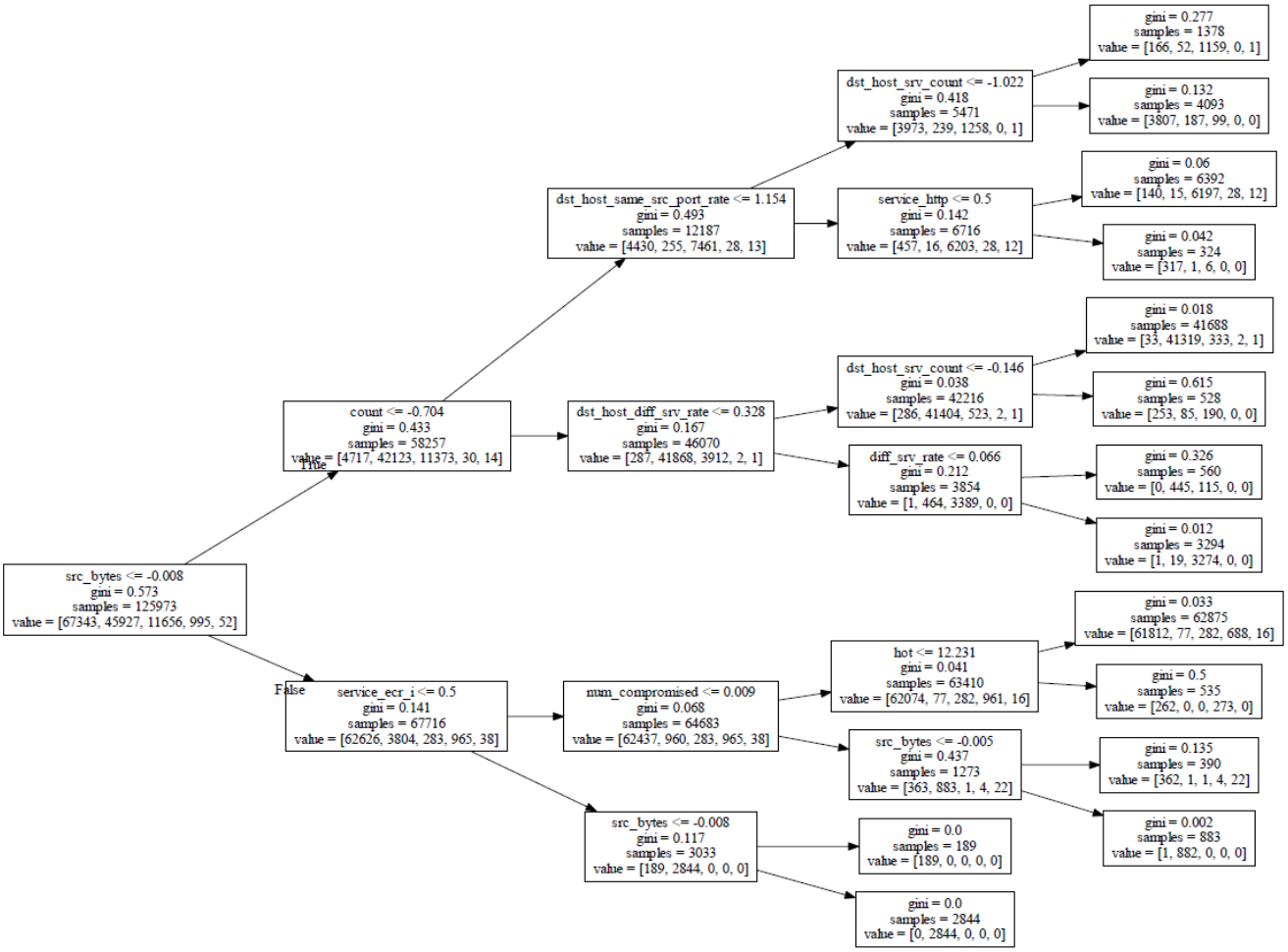

FIGURE 10. 4-level decision tree generated from NSL-KDD training data set.

found that src_bytes feature has the biggest Gini value and is the best choice for root node in Figure 10. Experiments show that the number of features and which features to select have a greater impact on the detection results. In the follow-up algorithm, we will also select 17 main features for the training of decision tree.

Adaboost SAMME.R algorithm trains several weak classifiers to form a strong classifier [16], the detection effect is not as good as the accuracy of using decision tree algorithm, and the accuracy of using 200 estimators is not significantly improved listed in Figure 9, which shows that the Adaboost algorithm is not always effective.

We use NSL-KDD Test + dataset to compare decision tree, Adaboost and MultiTree algorithm. The result in Table 8 shows MultiTree we proposed achieves the best effect, and its accuracy is 84.23%.

3) ADAPTIVE VOTING RESULT

After the test and evaluation above, SVM algorithm takes a long time and has no advantage in accuracy, while Adaboost algorithm is not ideal, and the precision of logistic regression

TABLE 8. Comparison of Adaboost and MultiTree.

Algorithms	Accuracy	Precision	Recall	F1
DecisionTree	79.71%	83.51%	79.72%	77.31%
Adaboost	76.02%	81.82%	76.02	72.12%
MultiTree	84.23%	86.4%	84.23%	83.6%

algorithm is not high, so abandon these three algorithms. Considering the detection accuracy and operation performance, Decision Tree, Random Forest, kNN, DNN and MultiTree are selected as ensemble learning algorithms.

We give some examples in Table 9 to illustrate how the adaptive voting algorithm works, and the numbers in the table mean the type of samples. After summing up the output results of each algorithm, the weighted voting algorithm is used to calculate the final prediction results. It can be seen that the voting algorithm has better accuracy than the single algorithm.

After training with various algorithms, the adaptive voting algorithm is used to validate the NSL-KDD test set, and good results are obtained calculated from the Table10 and

TABLE 9. Adaptive voting sample on KDDTest+.

DeciTree	R.forest	kNN	DNN	Multi Tree	**Voting**	**True**
3	0	0	3	0	**0**	**0**
1	1	1	1	1	**1**	**1**
4	0	4	4	3	**4**	**4**
3	4	0	4	4	**4**	**4**
4	0	4	4	2	**4**	**4**
0	0	0	0	2	**2**	**2**
3	0	0	4	2	**4**	**3**
0	0	0	4	3	**3**	**3**
2	2	2	3	2	**2**	**2**
0	0	0	0	3	**3**	**3**
0	0	0	4	3	**3**	**3**
2	0	2	2	0	**0**	**0**

TABLE 10. Result of voting.

Method	Normal	Dos	Probe	R2L	U2R
Voting	94.93%	84.37%	87.11%	55.27%	25%

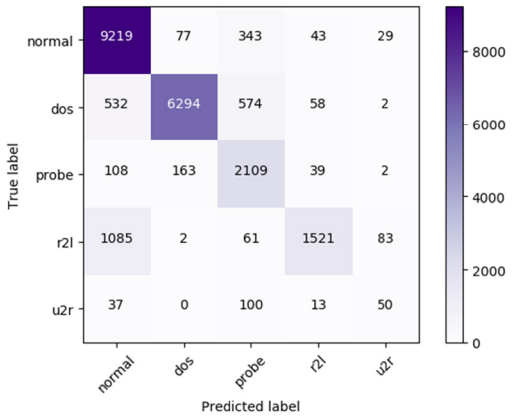

FIGURE 11. Confusion matrix of adaptive voting(85.2%).

Figure 11. The metrics are Accuracy: 0.852, Precision:0.865, Recall: 0.852, F1: 0.849.

The PCA principal component analysis method is used to analyze the test result shown in Figure12.

It is found that the data distributed in the test set has some similarities with training data. Among them, R2L and U2R data overlap with other data, and their distribution is not uniform, which makes classification difficult. After using the boosted algorithm to classify the test data, the erroneous classified data are displayed by PCA method. Most of the data points in the test data have been successfully classified and deleted from the voting results figure. It is found that Normal and Probe data are well separated, but some DOS (red) data are not classified successfully. But it also shows a certain linear regular distribution that help us to continue optimizing our work in the future.

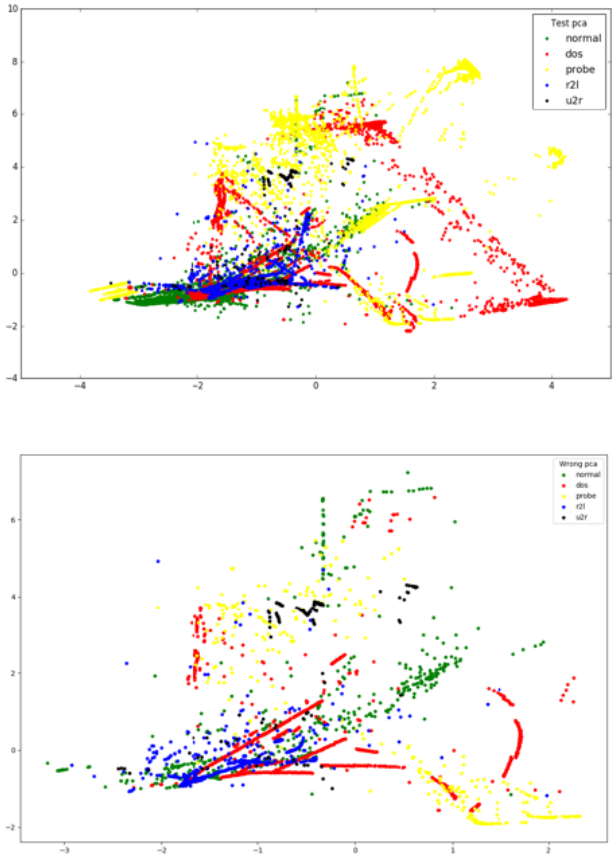

FIGURE 12. PCA Compare test set with voting wrong result (14.8%).

TABLE 11. Comparison of the accuracy of our model with other method.

Author	algorithm	Classes	Data Set	Accuracy
Our Model	**Ensemble Voting**	**5**	**KDDTest+**	**85.2%**
Our Model	**Multi Tree**	**5**	**KDDTest+**	**84.23%**
Our Model	**DNN**	**5**	**KDDTest+**	**81.61%**
Majd Latah[12]	kNN+ ELM	5	KDDTest+	84.29%
KEHE WU[8]	CNN	5	KDDTest+	79.48%
Tavallaee[3]	NB Tree	5	KDDTest-21	66.16%
Ingre B.[24]	ANN	2	KDDTest+	81.2%
Aggarwal P[16]	Random Tree	5	KDDTest+	83.04%
Ambusaidi [17]	LSSVM-IDS	5	Corrected KDD 99	78.86%
Al-Qatf M [18]	SAE_SVM	2	KDDTest+	84.96%

C. PERFORMANCE COMPARISON

In order to objectively evaluate the effect of our algorithm, we compare the test results with the data of other papers.

The results of comparison in Table 11 show that our adaptive ensemble machine learning model is an efficient approach for intrusion detection, and our ensemble model gives the best attack classification on the KDDTest+ dataset.

V. CONCLUSION

According to the theory that there is no free lunch, no learning algorithm is the best learner in any scenario. In the detection effect of single classification algorithm, the performance difference of each algorithm is not prominent. No matter what learning algorithm is adopted, a series of methods can be used to improve the detection effect. In this paper, we proposed an adaptive ensemble learning model. The key idea of our model is to use ensemble learning to gather the advantages of different algorithms. We use the method of ensemble learning to improve the detection effect. Compared with other research papers, it is proved that our ensemble model effectively improves the detection accuracy. The accuracy of the adaptive voting algorithm we proposed is 85.2%, and the precision 86.5%, the recall 85.2%, the F1 84.9%, better than algorithms in Table 6. Compared with other algorithms of the same kind, the effect of the algorithm is obviously improved, and it has great practical value. Although deep neural network has some advantages in detection effect, it takes long time in our comparative experiment, which means it will lead to a long detection delay in the practical application scenario of broadband network which will affect the response time of attack detection. Although the effect of a decision tree is not as good as that of DNN, the result of our MultiTree algorithm is better than that of DNN algorithm. For imbalanced classification scenarios, adjusting the proportion of samples, setting different class-weights and choosing appropriate features can improve the accuracy of machine learning algorithm. In the subsequent practical applications in the field of intrusion detection, the primary goal is to improve the quality of training data as much as possible, optimize feature extraction and preprocessing methods, and make the data more separable. In addition, for a small number of types of attacks such as U2R, separate optimization methods should be considered to improve the detection capability of such high-level threat attacks. Ensemble machine learning has a good generalization effect, which is worthy of continuous promotion and optimization in the field of network security research and application.

REFERENCES

[1] *KDD Cup Data*. Accessed: 1999. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[2] *NSL-KDD Dataset*. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html

[3] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 1–6.

[4] E. Hodo, X. Bellekens, A. Hamilton, C. Tachtatzis, and R. Atkinson, "Shallow and deep networks intrusion detection system: A taxonomy and survey," 2017, *arXiv:1701.02145*. [Online]. Available: https://arxiv.org/abs/1701.02145

[5] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[6] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Int. J. Eng. Res. Technol.*, vol. 2, no. 12, pp. 1848–1853, 2013.

[7] M. Tao, W. Fen, and C. Jianjun, Y. Yang, and C. Xiaoyun, "A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks," *Sensors*, vol. 16, no. 10, p. 1701, 2016.

[8] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018.

[9] L. Dhanabal and S. P. Shantharajah, "A study on NSL-KDD dataset for intrusion detection system based on classification algorithms," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 4, no. 6, pp. 446–452, 2015.

[10] H. Nkiama, S. Z. M. Said, and M. Saidu, "A subset feature elimination mechanism for intrusion detection system," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 3, pp. 148–157, 2016.

[11] Y. Lecun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," 2018, *arXiv:1806.03875*. [Online]. Available: https://arxiv.org/abs/1806.03875

[13] A. K. Shrivas and A. K. Dewangan, "An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set," *Int. J. Comput. Appl.*, vol. 99, no. 3, pp. 8–13, 2014.

[14] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[15] M. Albayati and B. Issac, "Analysis of intelligent classifiers and enhancing the detection accuracy for intrusion detection system," *Int. J. Comput. Intell. Syst.*, vol. 8, no. 3, pp. 841–853, 2015.

[16] P. Aggarwal and S. K. Sharma, "Analysis of KDD dataset attributes—Class wise for intrusion detection," *Procedia Comput. Sci.*, vol. 57, pp. 842–851, 2015.

[17] M. A. Ambusaidi, X. He, P. Nanda, and Z. Tan, "Building an intrusion detection system using a filter-based feature selection algorithm," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 2986–2998, Oct. 2016.

[18] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[19] Y. Zhou, M. Han, L. Liu, J. S. He, and Y. Wang, "Deep learning approach for cyberattack detection," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2018, pp. 262–267.

[20] T. Omrani, A. Dallali, B. C. Rhaimi, and J. Fattahi, "Fusion of ANN and SVM classifiers for network attack detection," in *Proc. 18th Int. Conf. Sci. Techn. Autom. Control Comput. Eng. (STA)*, Dec. 2017, pp. 374–377.

[21] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[22] Y. Wahba, E. ElSalamouny, and G. ElTaweel, "Improving the performance of multi-class intrusion detection systems using feature reduction," 2015, *arXiv:1507.06692*. [Online]. Available: https://arxiv.org/abs/1507.06692

[23] S. A. Ludwig, "Intrusion detection of multiple attack classes using a deep neural net ensemble," in *Proc. Symp. Series Comput. Intell. (SSCI)*, Nov./Dec. 2017, pp. 1–7.

[24] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Jan. 2015, pp. 92–96.

[25] J. Zhao, S. Shetty, J. W. Pan, C. Kamhoua, and K. Kwiat, "Transfer learning for detecting unknown network attacks," *EURASIP J. Inf. Secur.*, vol. 2019, no. 1, p. 1, 2019.

[26] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[27] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[28] M. A. Nielsen, *Neural Networks and Deep Learning*. San Francisco, CA, USA: Determination Press, 2015.

[29] C. Chio and D. Freeman, *Machine Learning and Security: Protecting Systems with Data and Algorithms*. Newton, MA, USA: O'Reilly Media, 2018.

[30] A. N. Jaber, M. F. Zolkipli, H. A. Shakir, and M. R. Jassim, "Host based intrusion detection and prevention model against DDoS attack in cloud computing," in *Proc. Int. Conf. P2P Parallel, Grid, Cloud Internet Comput.* Cham, Switzerland: Springer, 2017.