IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 27, 2018, accepted August 24, 2018, date of publication September 13, 2018, date of current version October 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2868171

A Novel Semi-Supervised Learning Approach for Network Intrusion Detection on Cloud-Based Robotic System

YING GAO, YU LIU, YAQIA JIN, JUEQUAN CHEN, AND HONGRUI WU
Department of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China

Corresponding authors: Ying Gao (gaoying@scut.edu.cn) and Yu Liu (350268634@qq.com)

This work was supported in part by the Guangzhou People's Livelihood Science and Technology Project under Grant 201803010097 and in part by the Guangzhou Key Industrial Technology Project under Grant 201802020014 and Grant 201802010035.

ABSTRACT Although the cloud-based robotic system has provided the services in various industries, its data safety is continuously threatened, and the network intrusion detection system (NIDS) is considered as a necessary component to ensure its security. In recent years, many machine learning (ML) techniques have been applied for building a more intelligent NIDS. Most NIDSs based on the ML method and artificial intelligence techniques are either supervised or unsupervised. However, the supervised learning for NIDS depends much on the labeled data. This weakness makes it harder to detect the latest attack patterns. Meanwhile, the unsupervised learning for NIDS often fails to give the satisfactory results. Therefore, this paper proposed a novel fuzziness-based semi-supervised learning approach via ensemble learning for network intrusion detection on the cloud-based robotic system, which can address the above issues. First, due to the good generalization ability of ensemble learning, we construct an ensemble system trained by the labeled data. Moreover, for better utilizing the unlabeled data, a fuzziness-based method is adopted for data analysis. In this way, the noisy and redundant examples in the data set are removed. Finally, we use the same ensemble approach to combine both supervised and unsupervised parts. To verify the effectiveness and robustness of the NIDS, the proposed approach is tested on the NSL-KDD data set, which is a commonly used traffic data set. The experimental results show that the proposed approach achieves the accuracy 84.54% and 71.29% on the, respectively, "KDDTest+" and "KDDTest-21" data sets. When compared with the state-of-the-art method, the proposed method also delivers a promising result.

INDEX TERMS Intrusion detection, clouds, robotic system, artificial intelligence (AI), semi-supervised learning.

I. INTRODUCTION

Network intrusion detection system (NIDS) is a device or software that detects the abnormal use of the system by monitoring and analyzing the environment of network [1]. In 1987, Denning first introduced the concept of NIDS and proposed a detection model, which can explore a wide range of security violations on computer networks [2]. In recent years, since the unknown attacks are continuously increased, the traditional network protection tools, such as firewalls, access control or encryption, fail to protect the computer network against the novel attacks [3]. As a result, the efforts currently focused on establishing the more complex systems or network architectures, e.g., cyber physical systems [4], multi-dimensional context-aware social network architecture [5], quality-aware service access system [6], emotion-aware cognitive system [7] and NIDS etc. Among these security applications, the NIDS increasingly attracts attentions and has become the primary concern of the cloud-based robotic system. This is because the cloud-based robotic system needs a lot of data interaction and transmission. The transmission data consist of numerous privacy data. It is unavoidable that the security of privacy data is continuously threatened by the network intrusion. In order to ensure the safety of privacy data, the NIDS is necessary when constructing the cloud-based robotic system. Therefore in this paper, we proposed a novel NIDS to improve the security of cloud-based robotic system.

Despite the effectiveness of NIDS, establishing the NIDS is still challenging. This is because some issues, such as data collections and intrusion recognition, should be

taken into consideration [3]. Consequently, some public benchmark datasets (e.g., KDDCup99 and NSL-KDD [8]) are built and the sophisticated NIDSs are constructed to improve the performance of intrusion recognition. Since intrusion recognition can be considered as a classification task, numerous artificial intelligence (AI) techniques and machine learning (ML) techniques have been applied in the NIDSs [9], [10]. Generally, the ML-based techniques can be supervised or unsupervised. The task of supervised learning is to learn a mapping from the feature instances to the particular categories by merely using labeled data. Many supervised learning approaches, e.g., decision tree (DT) [11], deep neural network (DNN) [12], support vector machine (SVM) [13] and etc., have been successfully employed to recognize and detect the intrusions. The supervised learning approaches for NIDS have achieved high detection accuracy on many benchmark datasets. However, its shortcomings are obvious. First, the acquisition of labeled data needs extensive expertise and is often costly, and hence, the update of the detection model is expensive [14]. Second, since the training process depends on the obsolete labeled data, the detection model can hardly observe new types of attacks [15]. Unlike the supervised learning, the unsupervised learning approach trains the detection model without the labeled instances and only describes the hidden structure of unlabeled data. In unsupervised learning algorithm, different categories of network events are distinguished by estimating the distribution of unlabeled data. The examples with similar features are more likely to have the same class, and vice versa. Although the unsupervised learning has no need for the labeled data, it often results in a detection model with low accuracy and high false positive rate.

To overcome the above deficiencies, the semi-supervised learning is one of the implementations for NIDS. It combines both labeled and unlabeled data to establish the detection model [16]. On one hand, the semi-supervised learning reduces the dependency on the labeled data, and thus is considered to be more robust than supervised learning. On the other hand, since a small amount of labeled data is introduced, the semi-supervised learning usually performs better in accuracy and false positive alarm than unsupervised learning. However, the semi-supervised learning also shares the same disadvantages of both supervised and unsupervised learning approaches. Consequently, the semi-supervised learning approach for NIDS requires more sophisticated designs to reduce the negative impacts brought by both approaches [14].

In this paper, we propose a novel semi-supervised learning approach via ensemble learning for network intrusion detection. The ensemble-based system can reduce the variance in the classifier outputs [17]. In other words, its generalization capability outperforms the one in the single model-based system in most situations. Since there exists many attack types unseen in the train data, it is more suitable to choose the ensemble learning approach. For the labeled data, we first generate a group of basic classifiers as candidates and build an ensemble learning model based on the rankings of the classifiers. Moreover, to fully analyze the distribution of unlabeled data, a fuzziness-based method is adopted. Then according to the unsupervised learning results, a new ensemble learning system is built and extended to the previous one. Finally, the proposed algorithm will be tested on the NSL-KDD traffic dataset.

Our main contributions in this paper can be summarized as follows:

1) We present a novel ensemble learning approach to classify. Considering that the intrusion detection is a non-linear classification problem, we choose the classification and regression tree (CART) [18] as the basic learner of the ensemble system. In order to combines the outputs of CARTs, a 3-layers neural network is applied to decide their weight.
2) We adopt the fuzziness-based method to mine the hidden structure of unlabeled data. The method extracts the useful information and removes the redundant term from the unlabeled data, which improves the performance of the proposed approach.
3) We combine both supervised and unsupervised part via ensemble learning approach. By this means, the labeled data correct the classification of unlabeled data. In the meantime, due to the lack of labeled data, the utilization of unlabeled data completes the construction of classifier to make the detection process more robust and accurate.

This paper is organized as follows: Section II provides the background of ML-based approaches for NIDS and a brief description of the semi-supervised learning. Then, the proposed semi-supervised learning approach for NIDS on cloud-based robotic system is given in Section III. Experimental analysis and results are carried out in Section IV. Finally, the conclusions are drawn in Section V.

II. RELATED WORK

In this section, we mainly introduce some preliminaries related to this paper. Specifically, we first elucidate the basic framework for semi-supervised learning. Then we elaborate several ML-based approaches for NIDS.

A. SEMI-SUPERVISED LEARNING

The semi-supervised learning approach is considered as the combination of supervised and unsupervised learning. Unlike the hybrid methods by using both approaches only on labeled data, the semi-supervised technique tries to learn from both labeled and unlabeled examples [22]. Specifically, the labeled data is represented as $S^l = (x_1^l, y_1^l), (x_2^l, y_2^l), \ldots, (x_{n_l}^l, y_{n_l}^l)\}$ and the unlabeled data is formulized as $S^u = \{x_1^u, x_2^u, \ldots, x_{n_u}^u\}$, where n_l and n_u are the number of labeled and unlabeled examples respectively. Generally, when training a semi-supervised model, the labeled examples are often in a small size and the unlabeled examples are available in a vast amount, which implies $n_l \ll n_u$. For better understanding the semi-supervised

learning methods, two commonly-used techniques, namely self-training method and co-training method, are introduced in the followings.

1) SELF-TRAINING

Self-training method [23] is a simple strategy to train both labeled and unlabeled data. It initially generates a fitted classifier with labeled examples. Then the classifier assigns the labels to the unlabeled examples. Meanwhile, the classifier is retrained with both labeled data and unlabeled data with predicted labels. The above processes are run iteratively until the terminal criterion is met. Self-training is also a wrapper algorithm [24]. It can be widely applied in many basic learning algorithms. Therefore, many efforts adopted the self-training to further develop their semi-supervised learning approaches. It has been reported that the self-training is outperformed in the natural language process tasks [25], [26], object detection [27] and etc. However, the self-training still suffers from two main disadvantages [28]. On one hand, the confident instances in self-labeled examples are often far from the decision boundary. On the other hand, the training process is easily trapped by the outliers. Consequently, there is a room for the development of self-training method.

2) CO-TRAINING

Co-training is also a well-known semi-supervised learning method. Unlike the self-training, it splits the feature into two disjoint views and then separately trains two classifiers in an iterative manner. For successfully training, one hypothesis should be satisfied, that is, two views should be conditionally independent given the categorical attributes [29]. As a result, a better splitting method for feature seems to be more important. However, this is not an easy work. Feger and Koprinska [30] have tried to find the optimal splitting by using conditional mutual information. Unfortunately, they failed to improve the performance as the random splitting was more outperformed. Nigam and Ghani [23] also showed that the random splitting method appears to be better in performance with sufficient redundancy in data. Salaheldin and El Gayar [31] proposed a new splitting features method and the best splitting point is obtained by using GA. They finally found that their method was competitive with the random splitting. Despite the difficulty of finding the best splitting, the co-training is still a popular approach to implement the semi-supervised learning.

B. ML-BASED APPROACHES FOR NIDS

Since the Denning first introduced the model for NIDS in 1987 [2], many efforts have been devoted to find more sophisticated and effective approaches to further extend the traditional method. Among these approaches, the ML-based technique is one of the most robust and intelligent implementations for the detection model. Lu *et al.* [19] proposed a rule-pruning method based on genetic algorithm (GA) to select some useful detection rules. They suggested using GA to explore the optimal representations of rule. The approach was performed better on the KDDCup99 traffic dataset. Ahmed and Mahmood [20] presented an unsupervised learning approach to detect a particular intrusion namely denial of service (DoS). The model distinguished the normal and anomaly based on the clustering method. Their approach was outperformed other clustering-based anomaly detection methods in terms of detection accuracy. To explore the more recognizable features, Cao *et al.* [21] used auto encoder (AE), which is a structure of neural network, to represent the intrusion features. Then based on the new representations, the density estimation method is employed for classification. Similarly, considering the numerous quantity of training data, Shone *et al.* [12] combined the deep and shallow learning for intrusion detection. This approach adopted a deep neural network for unsupervised feature extraction and a random forest for classification. They found it more efficient and effective than the deep brief network (DBN).

The findings from the above literature utilized either supervised or unsupervised approaches to recognize the anomaly events. Although these researches have achieved the satisfactory improvements in terms of detection accuracy, there still exist some problems to be solved. To be more detailed, the supervised learning approaches are mostly limited by the labeled data. When updating the detection model, the new labeled data should be provided, whereas labeling data is a tough and costly job. In addition, with numerous labeled data, the detection model tends to be easily affected by the noise. As to the unsupervised learning approach, since there is no prior information of category in unlabeled data, the classifier easily suffers from the low detection accuracy and high false alarm rate [20]. Considering the above weaknesses, we think that it is needed to reduce the dependency on labeled data and improve the utilization of unlabeled data. The semi-supervised learning is one of the choices that can achieve the target, whereas few literature applies it in the NIDS. Therefore, we proposed a novel semi-supervised learning approach for network intrusion detection, which combines both supervised and unsupervised learning approaches for processing the labeled and unlabeled data, respectively. Through a series of experiments, we will prove that the proposed algorithm is more efficient and effective for NIDS.

III. PROPOSED APPROACH FOR NIDS

A novel fuzziness-based semi-supervised learning approach via ensemble learning (FSSL-EL) is presented in this section, which is more effective to apply for the network intrusion detection. To be more detailed, for the labeled data, we present an ensemble learning method for training. For the unlabeled data, a fuzziness-based method is utilized to estimate their entropy and thus another ensemble model is generated. For better understanding our method, the flowchart of the proposed method is described in Fig. 1.

In the following subsections, we first introduce the structure of the ensemble learning approach. Then based on the trained classifier, we elaborate the fuzziness-based method. In addition, the entire approach for combining the above

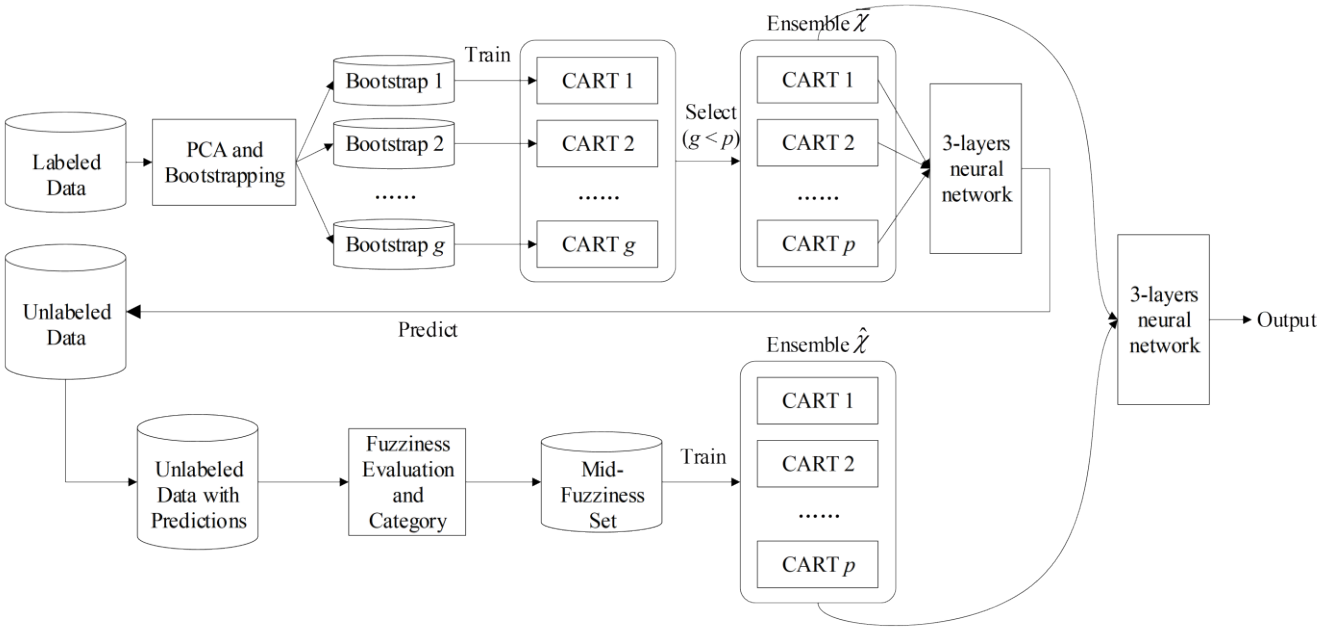


FIGURE 1. The training process of the proposed semi-supervised learning approach.

Algorithm 1 Ensemble Learning Approach

Input: The labeled examples S^l and the bootstrap sample rate r

1) Initialize the ensemble system $\bar{\chi} = \{\}$
2) Use PCA for dimensionality reduction on S^l
3) **for** $i = 1, 2, \ldots, g$
4) Generate bootstrap $\bar{B}_i$ by samplingS^l with sample rate r
5) Train a CART classifier $\bar{C}_i$ with $\bar{B}_i$
6) $\bar{\chi} = \bar{\chi} \bigcup \bar{C}_i$
7) **end for**
8) Select p $(p < g)$ CARTs $\bar{\chi} = \{\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_p\}$ based on results of the prediction accuracy on S^l
9) Train a 3-layers neural network $\bar{M}$ with labeled examples S^l
10) Form an ensemble system $\bar{\chi} = \{\bar{C}_1, \ldots, \bar{C}_p\} \bigcup \bar{M}$

Output: The ensemble system $\bar{\chi}$

processes is provided. Finally, a complexity analysis of the proposed approach is delivered.

A. PROPOSED ENSEMBLE LEARNING APPROACH

The Algorithm 1 shows the process of the proposed ensemble learning approach. Before training the ensemble model, we need the labeled data S^l and its size n_l. The i-th example S_i^l is formalized as (x_i^l, y_i^l), where $x_i^l \in \mathbb{R}^d$ contains d attributes and $y_i^l \in \{1, 2, \ldots, k\}$ is the class of x_i^l. Since we consider the intrusion detection as a multi-class problem, the class types should satisfy $k > 2$. With the labeled examples S_l given, the approach first uses the principal component analysis (PCA) algorithm [41] to extract the important features. The PCA retains the most interesting information from the traffic data and discards the nonvital components. Since the dataset contains some irrelevant features, the PCA is essential to be applied. After that, a bootstrapping method is adopted to repeatedly generate g dissimilar bootstraps $\{\bar{B}_1, \bar{B}_2, \ldots, \bar{B}_g\}$. The sampling rate is fixed to $r \in (0, 1)$ and the size of each bootstrap is rn_l. One reason for adopting the bootstrapping is that it ensures the diversity of basic classifiers, hence, the aggregation of the classifiers is meaningful. Another reason is that bootstrapping is kind of bagging techniques [32]. From the view of bias and variance, the bagging algorithm can reduce the expectation of output variance with the bias unchanged. The lower variance means the better generalization capability. It further indicates the less misclassifications on the unlabeled samples and the other unseen samples. As a result, the bootstrapping method is considered to be robust and stable to build an ensemble model.

After sampling, g CART classifiers $\{\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_g\}$ are generated by fitting their bootstraps $\{\bar{B}_1, \bar{B}_2, \ldots, \bar{B}_g\}$, respectively. CART is one of the most effective decision tree algorithms. It provides a simple and intuitive knowledge expression to alleviate linear and non-linear classification problems, also including the intrusion detection problem. Unlike the other tree-like classifiers (e.g., ID3, C4.5 and C5.0), it uses the Gini impurity to select the attribute. Given a group of examples X, the Gini impurity is formulized as:

$$I(X) = 1 - \sum_{i=1}^{k} p_i^2 \tag{1}$$

where p_i denotes to the probability of examples in A that belongs to class i. When the examples X split into

$(X_1, X_2, \ldots, X_m)$ according to the attribute A, the impurity value decreases as:

$$I(X, A) = \sum_{i=1}^{m} \frac{|X_i|}{|X|} I(X_i) \tag{2}$$

The best splitting attribute can be obtained by minimizing the $I(D, A)$. Overall, each CART $\bar{C}_i$ will fit its corresponding bootstrap $\bar{B}_i$ by iteratively finding the best split.

Among g CART classifiers, some with poor performance would degrade the performance of whole ensemble system. Therefore, we only select part of classifiers for ensemble and discard the others. The selection criterion is to choose the classifiers with higher detection accuracy on the labeled examples S^l. After filtering, only p ($p < g$) classifiers $\{\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_p\}$ are kept.

To combine the CART classifiers, a weighted voting method is performed. Instead of averaging the prediction results, the FSSL-EL utilizes a 3-layer neural network denoted as $\bar{M}$. The input of the network is the output of CARTs. Due to the representation of the multi-class task, we rewrite the output of CART in form of vector as follows:

$$\bar{C}_i(x) = (P(y = 1|x), \ldots, P(y = k|x)) \tag{3}$$

where $P(y = j|x), j = 1, 2, \ldots, k$ denotes to the probability that the example x belongs to class j. Here, the size of input layer is pk.

Suppose that the size of the hidden layer is d_H and the input example is x. From the input layer to the hidden layer, the calculation is formalized as follows:

$$\bar{H}(x) = \sigma\ (\bar{W}_H \bar{C}(x)^T + \bar{b}_H) \tag{4}$$

$$\bar{C}(x) = \begin{bmatrix} \bar{C}_1(x) \\ \vdots \\ \bar{C}_p(x) \end{bmatrix} \tag{5}$$

where $\bar{H}(x)$ is the output of hidden layer, σ is the sigmoid activation function $\sigma\ (t) = 1/(1 + e^{-t})$, $\bar{W}_H \in \mathbb{R}^{d_H \times pk}$ represent the hidden weights, and $\bar{b}_H \in \mathbb{R}^{d_H}$ denotes to the hidden biases.

Considering the multiple classification, we use softmax function [33] as the activation function rather than sigmoid. Therefore, from the hidden layer to output layer, the output is calculated as:

$$\bar{M}(x) = \mu(\bar{W}_M \bar{H}(x) + \bar{b}_M) \tag{6}$$

$$\mu(t) = \left(e^{t_1} / \sum_{i=1}^{k} e^{t_i}, \ldots, e^{t_k} / \sum_{i=1}^{k} e^{t_i} \right), t = (t_1, \ldots, t_k) \tag{7}$$

where $\bar{M}(x)$ is the output of the network, $\bar{W}_M \in \mathbb{R}^{k \times d_H}$ refers to the output weights, $\bar{b}_M \in \mathbb{R}^k$ refers to the output biases, and $\mu\ (t)$ is the softmax function.

For training the neural network $\bar{M}$, the loss function should be given. Here, the softmax loss function J for the network is formalized as:

$$J(\theta) = -\frac{1}{n_l} \sum_{i=1}^{n_l} \sum_{j=1}^{k} 1\{y_i^l = j\} \log \bar{M}_j(x_i^l) \tag{8}$$

where θ is the parameters of weights and biases of the network, x_i^l is the i-th labeled example, $\bar{M}_j(x_i^l)$ is the j-th component of the network output, and $1\{\cdot\}$ is the indicator function as eq. (9) shown:

$$1\{\Omega\} = \begin{cases} 1, & \text{if } \Omega \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

By using the backpropagation method [34] to minimize the loss function, the optimal parameters of the network are obtained. Finally, the CARTs and the neural network are combined into an ensemble system as $\bar{\chi} = \{\bar{C}_1, \ldots, \bar{C}_p, \bar{M}\}$. Our motivation to use the neural network can be concluded as two points. First, the neural network provides an adaptive way to learn the weight of basic classifier. Second, the neural network further improves the capability of non-linear transformation. Therefore, the ensemble system becomes more effective to solve a more complex classification problem.

Algorithm 2 Fuzziness-Based Method

Input: The unlabeled examples S^u and the ensemble system $\bar{\chi}$

1) Use the PCA to reduce the dimension of dataset
2) Assign the labels to S^u by using $\bar{\chi}$ and the dataset with labels is rewritten as S^{sl}
3) Use the entropy estimation in eq. (10) to evaluate the fuzziness of S^{sl}
4) Categorize the S^{sl} into three parts: FS^{low}, FS^{mid} and FS^{high}
5) Initialize the ensemble system $\hat{\chi} = \{\}$
6) **for** $i = 1, 2, \ldots, g$
7) Generate bootstrap $\hat{B}_i$ by sampling FS^{mid} with the ratio $rn_l/\lfloor n_u/3 \rfloor$
8) Train a CART classifier $\hat{C}_i$ with $\hat{B}_i$
9) $\hat{\chi} = \hat{\chi} \bigcup \hat{C}_i$
10) **end for**

Output: The ensemble system $\hat{\chi}$, the mid-fuzziness set FS^{mid}

B. FUZZINESS-BASED METHOD

To fully analyze the structure of unlabeled data, a fuzziness-based algorithm is applied and the method is shown in Algorithm 2. Fuzziness is considered as a type of uncertainty with the value between 0 and 1 [36]. It has been widely applied in many application fields, e.g., classification [37], data mining [38] and etc. In this paper, our motivation to adopt the fuzziness can be concluded as two points. First, the fuzziness-based method can provide estimation of the importance of each example, thus eliminates the irrelevant terms. Second, it has been proven that a proper fuzziness-based method related to the classifier will strengthen the

generalization capability [39]. In other words, the fuzziness-based method can enhance the detection ability for the new malicious events.

Before starting the fuzziness-based method, we first define the unlabeled data $S^u = \{x_1^u, x_2^u, \ldots, x_{n_u}^u\}$ with the size n_u. Also, we need PCA for feature extraction and the trained ensemble system $\bar{\chi}$ for prediction. By using $\bar{\chi}$, the examples in S^u are assigned with prediction labels. We rewrite the unlabeled examples with prediction labels as $S^{sl} = \{(x_1^u, \bar{\chi}(x_1^u)), \ldots, (x_{n_u}^u, \bar{\chi}(x_{n_u}^u))\}$, namely as self-labeled examples. Next, the Shannon's information entropy estimation is applied to measure the fuzziness of the classifier output [40], which can be calculated as:

$$F(x) = -\frac{1}{k}\sum_{i=1}^{k}(\bar{\chi}_i(x)\log_2\bar{\chi}_i(x) + (1-\bar{\chi}_i(x))\log_2(1-\bar{\chi}_i(x))) \quad (10)$$

where $\bar{\chi}_i(x)$ denotes to the output value of i-th node in ensemble system $\bar{\chi}$, i.e., the probability that the example x belongs to the i-th category.

After evaluating the fuzziness, the self-labeled samples S^{sl} will be categorized according to rank of the fuzziness value. Specifically, the approach divides the examples S^{sl} into three parts: the low-fuzziness set FS^{low}, mid-fuzziness set FS^{mid} and high-fuzziness set FS^{high}. These three fuzzy sets have the same size $\lfloor n_u/3 \rfloor$. Through the experiments, we found that FS^{mid} offered a better performance improvement in NIDS. Therefore, the approach builds a new ensemble system based on FS^{mid} and discards FS^{high} and FS^{low}. Similar to the training process of $\bar{\chi}$, the new ensemble adopts the bootstrapping method to take sample and further generates the corresponding CART classifiers. Each CART classifiers are trained by FS^{mid} with the prediction labels. To ensure the homogeneity with the CARTs in $\bar{\chi}$, the sampling rate of bootstrapping is set to $rn_l/\lfloor n_u/3 \rfloor$ and the number of CARTs is same as $\bar{\chi}$. The new ensemble is formalized as $\hat{\chi} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_p\}$.

C. ENTIRE SEMI-SUPERVISED LEARNING APPROACHE FOR NIDS

The proposed FSSL-EL combines the ensemble learning method and the fuzziness-based method for intrusion detection. The approach first generates an ensemble learning models $\bar{\chi}$ with the labeled data S^l. Moreover, given the prediction on unlabeled data S^u by using $\bar{\chi}$, the fuzziness-based method is adopted to evaluate fuzziness value and further groups the samples into FS^{high}, FS^{mid} and FS^{low}. Then another ensemble $\hat{\chi}$ is obtained by fitting the mid-fuzziness set FS^{mid}.

Since the input of the previous network $\bar{M}$ is fixed, it is not supportable to extend additional CART classifiers. Therefore, after two ensemble models are built properly, a new 3-layers neural network, namely M, is initialized to combine $\bar{\chi}$ and $\hat{\chi}$. The outputs of CARTs in both $\bar{\chi}$ and $\hat{\chi}$ are merged as the inputs of the network. The activation function of the hidden layer and output layer are sigmoid and softmax, respectively. The expressions are given as follows:

$$H(x) = \sigma\,(W_H C(x)^T + b_H) \quad (11)$$

$$C(x) = \begin{bmatrix} \bar{C}_1(x) \\ \vdots \\ \bar{C}_p(x) \\ \hat{C}_1(x) \\ \vdots \\ \hat{C}_p(x) \end{bmatrix} \quad (12)$$

$$M(x) = \mu(W_M H(x) + b_M) \quad (13)$$

where W_H, b_H, W_M and b_M are the parameters of the network. To train the new network M, the backpropagation method is used. The labeled examples S^l is incorporated with the mid fuzziness set FS^{mid} as the training data. Finally, the ensemble system that combines both supervised and unsupervised learning results is represented as $\chi = \{\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_p, \hat{C}_1, \hat{C}_2, \ldots, \hat{C}_p, M\}$.

Overall, the FSSL-EL uses the principle of bagging algorithm to build the classification model. It mainly consists of two parts: the supervised and the unsupervised part. For the supervised part, an ensemble learning approach based on CART is adopted to learn from the labeled data. The advantage of ensemble learning is its better generalization. This indicates the stronger ability to detect the novel attack patterns. For the unsupervised part, a fuzziness-based method helps to explore the inner structure of the unlabeled data, and hence another ensemble system is built. As a result, the fuzziness-based method makes the unlabeled data available for classification, which increases the utilization of data. In the meantime, the redundant terms are removed to increase the efficiency of the whole system. With the combination of supervised and the unsupervised part, the entire detection model becomes more robust and its performance is improved.

D. COMPLEXITY OF THE FSSL-EL APPROACH

We will analyze the time cost and the space complexity of the proposed algorithm in this subsection. The overall approach mainly contains three parts and the time complexity T can be represented as follows:

$$T = T_l + T_u + T_c \quad (14)$$

where T_l, T_u and T_c denote to the time cost of ensemble learning approach on labeled data, fuzziness-based method on unlabeled data and combination of above two methods, respectively. T_l is relevant to the size of labeled data n_l, the data dimensionality d, the class number k, the size of hidden layer d_H, the size of ensemble model g and p. Therefore, the calculation is as follows:

$$T_l = O(gdn_l \log n_l + n_l(pkd_H + d_H k)\omega) \quad (15)$$

where $dn_l \log n_l$ denotes to the complexity to generate a CART and ω is the iteration to train the neural network. T_u is associated with the size of labeled data n_u, the data

TABLE 1. The details of features in NSL-KDD dataset.

No.	Feature	Data Type	No.	Feature	Data Type	No.	Feature	Data Type
1	duration	numerical	15	su_attempted	numerical	29	same_srv_rate	numerical
2	protocol_type	symbolic	16	num_root	numerical	30	diff_srv_rate	numerical
3	service	symbolic	17	num_file_creations	numerical	31	srv_diff_host_rate	numerical
4	flag	symbolic	18	num_shells	numerical	32	dst_host_count	numerical
5	src_bytes	numerical	19	num_access_files	numerical	33	dst_host_srv_count	numerical
6	dst_bytes	numerical	20	num_outbound_cmds	numerical	34	dst_host_same_srv_rate	numerical
7	land	symbolic	21	is_hot_login	symbolic	35	dst_host_diff_srv_rate	numerical
8	wrong_fragment	numerical	22	is_guest_login	symbolic	36	dst_host_same_src_port_rate	numerical
9	urgent	numerical	23	count	numerical	37	dst_host_srv_diff_host_rate	numerical
10	hot	numerical	24	srv_count	numerical	38	dst_host_serror_rate	numerical
11	num_failed_logins	numerical	25	serror_rate	numerical	39	dst_host_srv_serror_rate	numerical
12	logged_in	symbolic	26	srv_error_rate	numerical	40	dst_host_rerror_rate	numerical
13	num_compromised	numerical	27	rerror_rate	numerical	41	dst_host_srv_rerror_rate	numerical
14	root_shell	numerical	28	srv_rerror_rate	numerical			

TABLE 2. The distribution of classes in NSL-KDD dataset.

	Training Dataset		Testing Dataset	
	KDD_Train	KDD_Train_20percent	KDDTest+	KDDTest-21
Normal	67343	13449	9711	2152
Probe	11656	2289	2421	2402
DOS	45927	9234	7458	4342
U2R	52	11	200	200
R2L	995	209	2754	2754
Total	125973	25192	22544	11850

dimensionality d and the size of the ensemble p as follows:

$$T_u = O(n_u p + p d n_u \log n_u) \tag{16}$$

Finally, the model combines the two ensembles by using neural network. Therefore, T_c is related to the samples size of both labeled and unlabeled data, the ensemble size and the structure of neural network as follows:

$$T_c = O(2p(n_u + n_l) + (n_u + n_l)(2pkd_H + d_H k)\omega) \tag{17}$$

The class number k is much smaller. Thus, the total time cost of the proposed method can be simplified as follows:

$$T = O(p(n_u + n_l)d_H\omega + pdn_u \log n_u + gdn_l \log n_l) \tag{18}$$

The space consumption contains the cost of data and model. Therefore, the space complexity S is formalized as follows:

$$S = S_d + S_m \tag{19}$$

The space complexity of data S_d and the model S_m are represented as:

$$S_d = O((n_l + n_u) \cdot (d + k)) \tag{20}$$

$$S_m = O(pN + pkd_H + d_H k) \tag{21}$$

where N is the number of the nodes in CART. The space complexity through the whole training process is $O((n_l + n_u)(d + k) + p(N + kd_H))$.

IV. EXPERIMENT AND ANALYSIS

In this section, we introduce the NSL-KDD dataset and some experimental settings related to the proposed algorithm. In order to validate the performance of the proposed method, several comparison experiments are made.

A. NSL-KDD TRAFFIC DATASET

The NSL-KDD dataset was first presented by Travallaee *et al.* [8]. They found that the KDDCup99 dataset brought some issues, such as the redundant records, duplicate records and the unreasonable distribution of records. Considering that the above inherent problems would have adverse effects on the performance of detection system, they presented an advanced benchmark dataset, namely NSL-KDD traffic dataset. Generally, the NSL-KDD dataset has the similar structure as KDDCup99 dataset. The NSL-KDD includes both training sets ('KDDTrain' and 'KDDTrain_20percent') and testing sets ('KDDTest+' and 'KDDTest-21'). The features of the dataset describe the basic contents and statistical information of network connection. The total size of features is 41. The labels of NSL-KDD dataset record five typical network events, i.e., normal, probe, denial of service (DOS), user to root (U2R) and remote to local (R2L). To have a more comprehensive understanding, the details of NSL-KDD dataset are provided in Table 1 and Table 2.

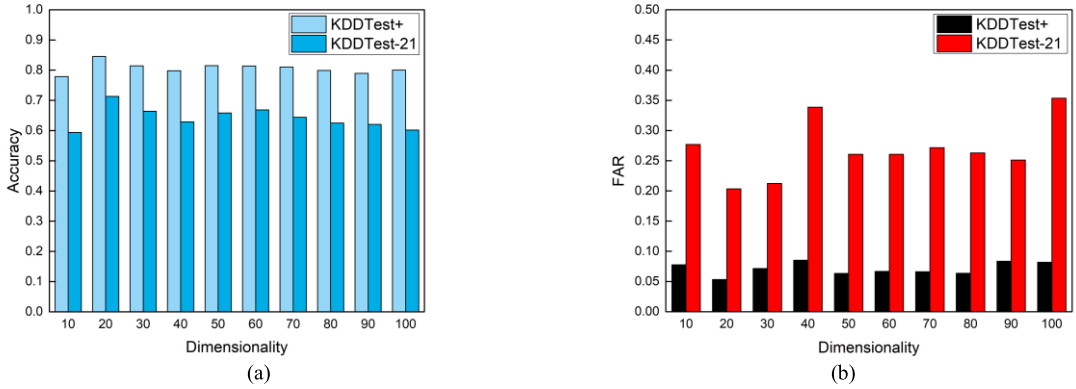


FIGURE 2. The effects of the PCA algorithm.

Nowadays, although some other newer traffic datasets are available for evaluation, many researches still consider the NSL-KDD dataset as one of the most authoritative benchmarks in field of intrusion detection. Therefore, in this paper, we utilize the NSL-KDD dataset for evaluating the proposed semi-supervised learning approach. Specifically, we use the 'KDDTrain_20percent' dataset for training, and 'KDDTest+' and 'KDDTest-21' for testing. Since these two testing sets, especially 'KDDTest-21', contain lots of novel

attack patterns, they are considered more suitable to validate the generalization capability of a model. Moreover, from the training dataset, we randomly select 2000 examples as the labeled data S_l and the remaining examples are used as the unlabeled data S_u. Besides, we treat the intrusion detection as a multi-class problem in consideration of five categories: normal, probe, DOS, U2R and R2L.

B. EXPERIMENTAL SETTINGS

1) EXPERIMENTAL ENVIRONMENT

In this research, the detection model is implemented by using the scikit-learn library, which is one of the most efficient machine learning tools. All the experiments are performed on a personal computer. The configurations of the computer are listed as follows: the Window 10 operation system, Intel i5-7400 CPU @ 3.00GHz and 8GB main memory.

2) DATA PREPROCESSING

In the NSL-KDD dataset, there are mainly two feature types: symbolic and numerical. Although the proposed algorithm can handle the symbolic features, the feature values are still not distributed uniformly [14] and it may trigger a negative effect on the learning process. To overcome the problem, the one-hot encoding method [35] and data normalization are adopted before starting to learn. The one-hot encoding method is an effective approach to convert the symbolic or discrete feature into the numerical one. After transformation, the new feature value is encoded as a sequence only with 0 and 1. Its dimensionality increases according to the distinct value in the corresponding symbolic feature. In the NSL-KDD dataset, the symbolic features like 'protocol_type', 'service' and 'flag' are encoded by using one-hot encoding method as their distinct values are more than 2. The remaining symbolic features can be treat as the Boolean type with the value either 0 or 1. After encoding, the dimensionality of data will increase from 41 to 122. In addition, a data normalization method is used. The aim of the data normalization is to scale the feature values within the interval [0, 1] as follows:

$$x_{i,j}^{normal} = \frac{x_{i,j} - \min(x_{:,j})}{\max(x_{:,j}) - \min(x_{:,j})} \tag{22}$$

where $x_{i,j}$ is the j-th attribute value for the example x_i, $x_{:,j}$ represents the j-th feature.

3) EVALUATION METRICS

To evaluate the performance of the prediction results, several metrics are performed and their representations are given as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{23}$$

$$\text{False Alarm Rate (FAR)} = \frac{FP}{FP + TN} \tag{24}$$

where true positives (TP) denotes to the number of attack samples correctly classified as an attack, true negative (TN) refer to the number of attack samples incorrectly classified as normal, false positive (FP) represents the number of normal samples correctly classified as normal and false negative (FN) is the number of normal samples incorrectly classified as an attack. The detection accuracy describes the ability of model to make correct prediction. The FAR can reflect the ratio that the normal traffics are recognized as the attacks in an incorrect manner. Therefore, a well-performed detection model should have a high accuracy and a low FAR.

C. EXPERIMENTAL RESULTS

1) EFFICIENCY OF THE PCA METHOD

Since the proposed approach uses the PCA to eliminate the redundant features, it is necessary to investigate the impact of the PCA. The Fig. 2 has shown the results in the form

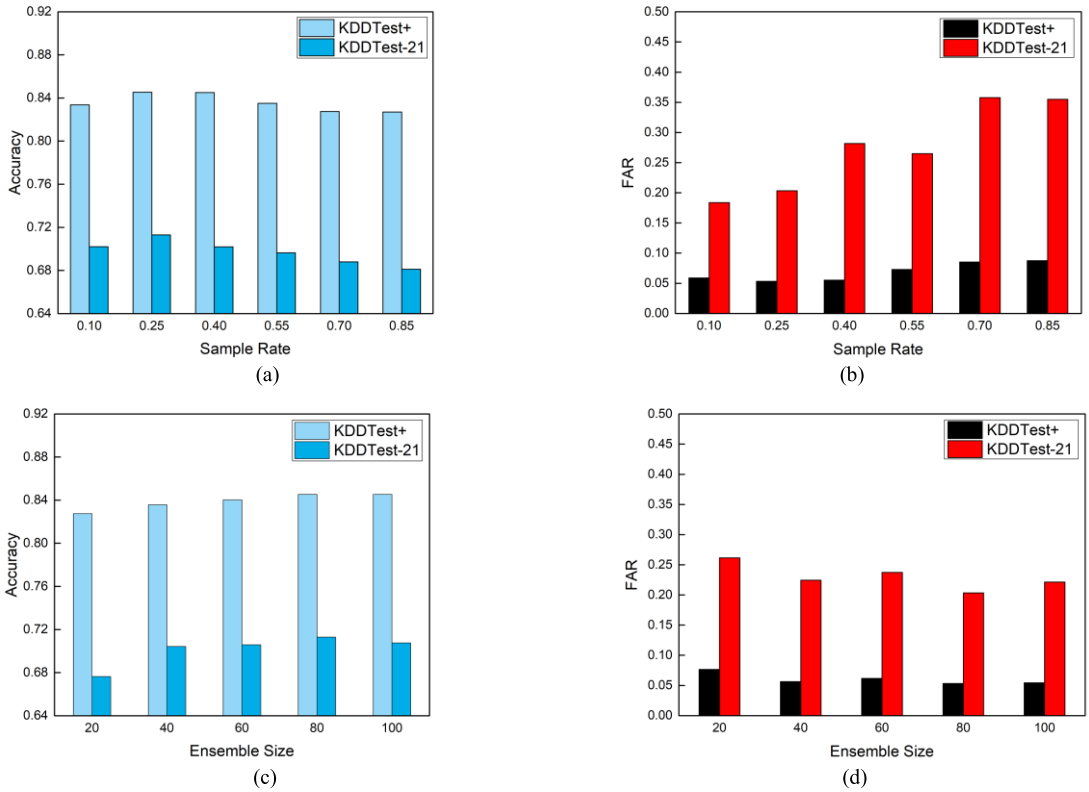


FIGURE 3. The effects of the bootstrapping method.

of graphs. From the Fig. 2, it can be observed that when the dimensionality reduces to 20, the approach obtains the result with the highest accuracy and lowest FAR. When the dimensionality is higher or lower than 20, the performance becomes worst. The experimental result indicates that the more compressed dataset loses more useful information. Meantime, the data with noise and relevant term adversely affect the detection performance. As a result, it is proven that the PCA is an essential process before the classification begins.

2) EFFICIENCY OF THE BOOTSTRAPPING METHOD

To study the efficiency of the bootstrapping method, we test the effect of two parameters related to the bootstrapping, i.e., the sample rate $r = \{0.1, 0.25, 0.4, 0.55, 0.7, 0.85\}$ and the ensemble size $p = \{20, 40, 60, 80, 100\}$. The Fig. 3 shows the experimental results. It can be observed that when the sample rate and the ensemble rate are set to 0.25 and 80 respectively, the FSSL-EL gets a better performance. As the sample rate rises or declines from 0.25, the accuracy drops and the FAR increases. This is because when the sample rate becomes larger, the bootstraps contain more noisy and redundant samples, and thus results in a poorer performance. Meanwhile, if the sample rate is too small, the bootstraps cannot fully describe the distribution of the original dataset and the whole ensemble model becomes under-fitting. Similarly, when the ensemble size is less than 80, the model is weaker in learning and fails to make the correct prediction. When the ensemble size is greater than 80, the performance is slightly decreased mainly because of the over-fitting learning toward the dataset. Overall, the settings of sample rate and ensemble size should be set properly. When the corresponding values are initialized to 0.25 and 80, the model delivers a promising performance

3) EFFICIENCY OF THE FUZZINESS-BASED METHOD

In this part, we mainly investigate the effect of the fuzziness-based method. The proposed method will be compared with the one without using fuzziness-based method. In addition, since the unlabeled data are assigned by the classification model, it is unavoidable that some of the unlabeled examples would be misclassified. Therefore, in the following experiments, we also study the contributions of three fuzzy sets and see which fuzzy set causes less misclassification on intrusion detection.

The Table 3 shows the experimental results and the accuracy of each class are provided. It is obvious that the fuzziness-based method plays an important role in improving the detection performance in comparison with the one without using the method. Among three fuzzy sets, the mid-fuzziness set offers the best performance in 'KDDTest+' and 'KDDTest-21' dataset, which has achieved the accuracy with the value 84.54% and 71.29%, respectively. In addition, the FSSL-EL with mid-fuzziness set also obtains a satisfactory result in FAR when compared with most of other results. It proves that the mid-fuzziness set produces

TABLE 3. The effects of the fuzziness-based method.

	'KDDTest+'							'KDDTest-21'						
	Accuracy						FAR	Accuracy						FAR
	Normal	Probe	DOS	U2R	R2L	Total		Normal	Probe	DOS	U2R	R2L	Total	
No Fuzzy Set	93.35%	74.96%	82.74%	7.05%	64.95%	83.63%	6.65%	76.71%	72.16%	70.41%	6.60%	65.25%	69.63%	23.29%
High Fuzziness	92.17%	70.92%	82.46%	7.50%	60.87%	82.10%	7.83%	73.44%	76.41%	70.34%	8.10%	**66.45%**	70.18%	26.56%
Mid Fuzziness	**94.69%**	76.44%	**82.75%**	6.85%	**66.38%**	**84.54%**	**5.31%**	**79.65%**	76.97%	**70.87%**	**8.60%**	65.04%	**71.29%**	**20.35%**
Low Fuzziness	92.96%	**78.64%**	81.93%	7.30%	64.59%	83.55%	7.04%	75.03%	**77.60%**	69.04%	6.80%	64.63%	69.79%	24.97%

less misclassification and is more helpful to detect anomaly. However, since the numbers of 'U2R' and 'R2L' in the training samples are far smaller than the other classes, their corresponding accuracies are relatively lower. Despite of this, the proposed method performs better and more stable when predicting the classes in a larger size.

Overall, the mid-fuzziness set produces most of the best results. It is evident that the fuzziness-based technique is an effective approach to handle the intrusion detection problem.

4) EFFICIENCY OF THE BASIC CLASSIFIER

In order to explore the impact of basic classifier in the ensemble system, we replace the basic classifier with different ML algorithms. Specifically, the origin FSSL-EL (temporarily denoted as FSSL-EL-CART) is compared with the one based on J48 (denoted as FSSL-EL-J48), k-nearest neighbor (denoted as FSSL-EL-KNN), logistic regression algorithm (denoted as FSSL-EL-LR) and Gaussian Naïve Bayes (denoted as FSSL-EL-GNB). Since the logistic regression classifier can merely handle the binary class problem, we use the one-vs-rest (OvR) scheme to make it supportable for the multi-class case.

TABLE 4. The comparison among different basic classifiers.

	'KDDTest+'		'KDDTest-21'	
	Accuracy	FAR	Accuracy	FAR
FSSL-EL-CART	**84.54%**	**5.31%**	**71.29%**	**20.35%**
FSSL-EL-J48	83.14%	6.40%	69.69%	24.22%
FSSL-EL-KNN	80.78%	6.54%	65.07%	21.28%
FSSL-EL-LR	77.57%	9.06%	58.22%	36.79%
FSSL-EL-GNB	75.76%	11.85%	55.57%	49.19%

The experimental result is provided in Table 4. It can be found that the tree-like classifiers, i.e., FSSL-EL-CART and FSSL-EL-J48, outperform the other algorithms. They obtain the accuracy 84.54% and 83.14% on 'KDDTest+' dataset, and meanwhile 71.29% and 69.69% on 'KDDTest-21' dataset. The other approaches like FSSL-EL-KNN, FSSL-EL-LR and FSSL-EL-GNB, receive the unsatisfactory results with the accuracy below 81% on 'KDDTest+' and below 66% on 'KDDTest-21'. The reason is possibly that the decision tree classifier produces a stronger non-linear representation. The representations are further enhanced by the ensemble and become more capable in a complex detection environment. Therefore, For the improvement of the performance, it is more suitable to select the tree-like classifier as the basic learner, especially CART.

TABLE 5. The comparison among the traditional ML and state-of-art approaches.

	'KDDTest+'		'KDDTest-21'	
	Accuracy	FAR	Accuracy	FAR
J48	81.05%	N/A	63.97%	N/A
Naïve Bayes	76.56%	N/A	55.77%	N/A
NB tree	82.02%	N/A	66.16%	N/A
Random forest	80.67%	N/A	63.25%	N/A
Random tree	81.59%	N/A	58.51%	N/A
Multi-layer perceptron	77.41%	N/A	57.34%	N/A
SVM	69.52%	N/A	42.29%	N/A
SMR[43]	75.23%	N/A	N/A	N/A
STL[43]	79.10%	N/A	N/A	N/A
RNN-IDS[44]	81.29%	N/A	64.67%	N/A
DBN[22]	80.58%	19.42%	N/A	N/A
Experiment-1[15]	82.41%	N/A	67.06%	N/A
Experiment-2[15]	84.12%	N/A	68.82%	N/A
FSSL-EL	**84.54%**	**5.31%**	**71.29%**	20.35%

5) COMPARISON WITH OTHER ML-BASED APPROACHES

In order to further validate the effectiveness of the proposed approach, we compare it with the state-of-art intrusion detection methods. In addition, several results obtained by the traditional ML methods (e.g., J48, SVM and etc.) [8] are provided as well.

The comparison results are shown in Table 5. It is worth mentioning that the 'Experiment-1' and 'Experiment-2' are also the semi-supervised learning approach. From Table 5,

it is observed that the proposed method is competitive with the traditional ML classifiers, including the ensemble learning method like random forest. Among the traditional ML methods, the tree-like methods outperform the other ML methods (i.e., SVM and Multi-layer perceptron). This result further indicates our motivation to choose CART as the basic classifier. Moreover, when compared with the algorithms with the structure of neural network (i.e., SMR, STL, RNN-IDS and DBN), the proposed approach still offers a better performance and seem to be more efficient. In addition, the FSSL-EL algorithm also outperforms the other two semi-supervised learning approaches. Specifically, it outperforms 'Experimental-1' by 2.13% and 4.23%, meanwhile, defeats 'Experimental-2' by 0.38% and 2.47% on 'KDDTest+' and 'KDDTest-21' respectively. Overall, the proposed method provides an effective way to detect intrusion and outperforms most of the state-of-art approaches for NIDS.

V. CONCLUSIONS AND OUR FUTURE WORK

In this paper, a novel semi-supervised learning approach for NIDS on cloud-based robotic system has been presented.

Through a series of experiments, the proposed algorithm is proved to be helpful in intrusion detection and improve the security of cloud-based robotic system.

Overall, our main contribution can be concluded as three points. First, a novel ensemble learning approach provides a reliable detection on the network intrusion. It enhances the ability to recognize the new traffic patterns. Second, the fuzziness-based method makes full use of the numerous unlabeled data and further increases the robust and detection accuracy of the whole system. Third, the FSSL-EL algorithm is proven to be more effective for solving the intrusion detection problem. With the comparison of many state-of-art approaches, the proposed method has delivered a promising performance. The experimental results also verify that the semi-supervised learning method is suitable to apply in the intrusion detection. For future studies, it would be necessary to improve the detection performance and model generalization. To achieve the target, we still keep interest in studying a more effective and efficient semi-supervised learning method for network security. In addition, we will test our method in more public benchmarks.

REFERENCES

[1] G. B. White, E. A. Fisch, and U. W. Pooch, "Cooperating security managers: A peer-based intrusion detection system," *IEEE Netw.*, vol. 10, no. 1, pp. 20–23, Jan. 1996.

[2] D. E. Denning, "An intrusion-detection model," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987.

[3] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, pp. 1–35, Jan. 2010.

[4] Y. Guo, X. Hu, B. Hu, J. Cheng, M. Zhou, and R. Y. K. Kwok, "Mobile cyber physical systems: Current challenges and future networking applications," *IEEE Access*, vol. 6, pp. 12360–12368, 2018.

[5] X. Hu, X. Li, E. Ngai, V. Leung, and P. Kruchten, "Multidimensional context-aware social network architecture for mobile crowdsensing," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 78–87, Jun. 2014.

[6] Z. Ning *et al.*, "A cooperative quality-aware service access system for social Internet of vehicles," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2506–2517, Aug. 2018.

[7] X. Hu *et al.*, "Emotion-aware cognitive system in multi-channel cognitive radio ad hoc networks," *IEEE Commun. Mag.*, vol. 56, no. 4, pp. 180–187, Apr. 2018.

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

[9] S. M. H. Bamakan, H. Wang, and Y. Shi, "Ramp loss K-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem," *Knowl.-Based Syst.*, vol. 126, pp. 113–126, Jun. 2017.

[10] K. Kalegele, K. Sasai, H. Takahashi, G. Kitagata, and T. Kinoshita, "Four decades of data mining in network and systems management," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, pp. 2700–2716, Oct. 2015.

[11] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[12] 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.

[13] Y. Li, J. Xia, S. Zhang, J. Yan, X. Ai, and K. Dai, "An efficient intrusion detection system based on support vector machines and gradually feature removal method," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 424–430, 2012.

[14] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for DDoS detection," in *Applied Intelligence*, no. 3. 2018, pp. 1–16.

[15] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.

[16] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Introduction Semi-Supervised Learn.*, vol. 3, no. 1, pp. 1–130, 2009.

[17] R. Polikar, "Ensemble learning. Ensemble machine learning," *Scholarpedia*, vol. 4, pp. 1–34, 2012.

[18] L. Breiman, J. H. Friedman, R. Olshen, and C. J. Stone, "Classification and regression trees," *Encyclopedia Ecol.*, vol. 40, no. 3, pp. 582–588, 1984.

[19] N. Lu, S. Mabu, T. Wang, and K. Hirasawa, "An efficient class association rule-pruning method for unified intrusion detection system using genetic algorithm," *IEEJ Trans. Elect. Electron. Eng.*, vol. 8, no. 2, pp. 164–172, 2013.

[20] M. Ahmed and A. N. Mahmood, "Novel approach for network traffic pattern analysis using clustering-based collective anomaly detection," *Ann. Data Sci.*, vol. 2, no. 1, pp. 111–130, Mar. 2015.

[21] V. L. Cao, M. Nicolau, and J. Mcdermott, "A hybrid autoencoder and density estimation model for anomaly detection," in *Proc. Int. Conf. Parallel Problem Solving Nature*, 2016, pp. 717–726.

[22] J. Levatić, D. Kocev, M. Ceci, and S. Džeroski, "Semi-supervised trees for multi-target regression," *Inf. Sci.*, vol. 450, pp. 109–127, Jun. 2018.

[23] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proc. 9th Int. Conf. Inf. Knowl. Manage.*, vol. 33, 2000, pp. 86–93.

[24] J. Tanha, M. van Someren, and H. Afsarmanesh, "Semi-supervised self-training for decision tree classifiers," *Int. J. Mach. Learn. Cybern.*, vol. 8, no. 1, pp. 355–370, 2017.

[25] B. Maeireizo, D. Litman, and R. Hwa, "Co-training for predicting emotions with spoken dialogue data," in *Proc. ACL Interact. Poster Demonstration Sessions*, 2004, p. 28.

[26] B. Wang, B. Spencer, C. X. Ling, and H. Zhang, "Semi-supervised self-training for sentence subjectivity classification," in *Proc. Conf. Can. Soc. Comput. Stud. Intell.*, 2008, pp. 344–355.

[27] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc. 7th IEEE Workshops Appl. Comput. Vis.*, vol. 1, Jan. 2005, pp. 29–36.

[28] M. F. A. Hady and F. Schwenker, "Semi-supervised learning," *J. Roy. Stat. Soc.*, vol. 172, no. 2, p. 530, 2013.

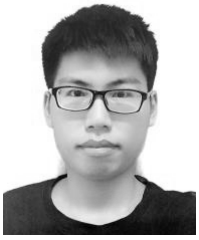
[29] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proc. 11th Annu. Conf. Comput. Learn. Theory*, 1998, pp. 92–100.

[30] F. Feger and I. Koprinska, "Co-training using RBF nets and different feature splits," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2006, pp. 1878–1885.

[31] A. Salaheldin and N. El Gayar, "New feature splitting criteria for co-training using genetic algorithm optimization," in *Multiple Classifier Systems*. 2010, pp. 22–32.
[32] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[33] N. M. Nasrabadi, "Pattern recognition and machine learning," *J. Electron. Imag.*, vol. 16, no. 4, p. 049901, 2007.
[34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 323, pp. 533–536, 1986.
[35] W. A. Chren, Jr., "One-hot residue coding for low delay-power product CMOS design," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 3, pp. 303–313, Mar. 1998.
[36] A. De Luca and S. Termini, "A definition of a nonprobabilistic entropy in the setting of fuzzy sets theory," *Inf. Control*, vol. 20, pp. 301–312, May 1972.
[37] H. Ishibuchi, S. Mihara, and Y. Nojima, "Parallel distributed hybrid fuzzy GBML models with rule set migration and training data rotation," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 2, pp. 355–368, Apr. 2013.
[38] J. Alcalá-Fdez, R. Alcalá, M. J. Gacto, and F. Herrera, "Learning the membership function contexts for mining fuzzy association rules by using genetic algorithms," *Fuzzy Sets Syst.*, vol. 160, no. 7, pp. 905–921, 2009.
[39] X.-Z. Wang, H.-J. Xing, Y. Li, Q. Hua, C.-R. Dong, and W. Pedrycz, "A study on relationship between generalization abilities and fuzziness of base classifiers in ensemble learning," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 5, pp. 1638–1654, Oct. 2015.
[40] X. Z. Wang, R. A. R. Ashfaq, and A. M. Fu, "Fuzziness based sample categorization for classifier performance improvement," *J. Intell. Fuzzy Syst.*, vol. 29, no. 3, pp. 1185–1196, 2015.
[41] S. Mika, B. Schölkopf, A. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 11. 1999, pp. 536–542.
[42] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun.*, Oct. 2016, pp. 258–263.
[43] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

YING GAO received the bachelor's and master's degrees from the Central South University of China and the Ph.D. degree from the South China University of Technology, China, in 1997, 2000, and 2006, respectively. She is currently a Professor with the School of Computer Science and Engineering, South China University of Technology. She has published more than 30 papers in international journals and conferences. Her current research interests include service-oriented computing technology, software architecture, and network security.

YU LIU was born in Guangdong, China, in 1994. He received the B.S. degree in computer science and engineering from the South China University of Technology in 2017, where he is currently pursuing the M.S. degree in computer science and engineering. His research interests include machine learning, deep learning, and evolutionary computation.



YAQIA JIN received the B.S. degree in computer science and engineering from the South China University of Technology in 2017, where he is currently pursuing the M.S. degree in computer science and engineering. His major research interests are big data analyses and cloud computing.

JUEQUAN CHEN received the B.S. degree in applied mathematics from the Guangdong University of Technology, China, in 2010, and the M.S. degree in computer science from the South China Normal University, China, in 2018. He is currently pursuing the Ph.D. degree with the South China University of Technology. His research interests include machine learning, pattern recognition, evolutionary computation, and cyberspace security.

HONGRUI WU received the B.S. degree from the South China University of Technology, China, in 2018, where he is currently pursuing the M.S. degree in computer science and engineering.