IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 7, 2017, accepted December 20, 2017, date of publication December 27, 2017, date of current version March 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2787719

An Effective Two-Step Intrusion Detection Approach Based on Binary Classification and *k*-NN

LONGJIE LI[1], YANG YU[1], SHENSHEN BAI[1,2], YING HOU[1], AND XIAOYUN CHEN[1]
[1]School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China
[2]Department of Electronic and Information Engineering, Lanzhou Vocational Technical College, Lanzhou 730070, China

Corresponding author: Longjie Li (ljli@lzu.edu.cn).

This work was supported in part by the National Natural Science Foundation of China under Grant 61602225 and in part by the Fundamental Research Funds for the Central Universities under Grant lzujbky-2017-192.

ABSTRACT Intrusion detection has been an important countermeasure to secure computing infrastructures from malicious attacks. To improve detection performance and reduce bias towards frequent attacks, this paper proposes a two-step hybrid method based on binary classification and *k*-NN technique. Step 1 employs several binary classifiers and one aggregation module to effectively detect the exact classes of network connections. After step 1, the connections whose classes are uncertain are sent to step 2 to further determine their classes by the *k*-NN algorithm. Step 2 is based on the outcomes of step 1 and yields a beneficial supplement to step 1. By combining the two steps, the proposed method achieves reliable results on the NSL-KDD data set. The effectiveness of the proposed method is evaluated in comparison with five supervised learning techniques. Experimental results demonstrate that the proposed method outperforms baselines with respect to various evaluation criteria. In particular, for U2R and R2L attacks, the F1-scores of the proposed method are much higher than those of baselines. Furthermore, comparisons with some recent hybrid approaches are also listed. The results illustrate that the proposed method is competitive.

INDEX TERMS Intrusion detection, hybrid method, binary classification, C4.5, *k*-nearest neighbors

I. INTRODUCTION

The prompt development of computer networks, especially the Internet, has brought considerable convenience to people in their daily lives, enterprises in their business dealings, organizations in their provision of services, etc. At the same time, various network security threats have become critically serious due to the continuous appearance of new vulnerabilities, and attack methods. Therefore, security mechanisms that can defend against these threats and maintain the confidentiality, integrity, and availability of computational resources have been indispensable.

An intrusion detection system (IDS) that is able to identify and prevent malicious network traffic has become an important security countermeasure [1], [2]. It monitors network events and collects network packets in a computing infrastructure. By analyzing the packets, an IDS detects abnormal behaviors and blocks malicious connections from attackers or intruders. In the last decade, the study of intrusion detection has captured increasing attention from security researchers [2]–[4].

In general, intrusion detection approaches are categorized as *misuse-based detection* and *anomaly-based detection* depending upon the fashion of analysis [5]–[7]. A misuse-based detection system identifies an intrusion by matching it with predefined signatures. Thus, profiles of attacks are required when building a misuse-based detection system. It is able to reliably detect known network attacks with a low false alarm rate, but new attacks slip through because their signatures are unknown. Alternatively, anomaly-based detection systems identify an attack by capturing the deviation from normal activity. Unlike misuse-based systems, anomaly-based systems are likely to recognize unknown intrusion behaviors. Because new attack methods keep emerging, anomaly-based detection systems have become increasingly important in protecting network security, notwithstanding the fact that they may suffer from a high false alarm rate [7]. In recent years, with the great efforts of researchers, anomaly-based detection systems based on machine learning and data mining techniques have been proposed to provide reliable detection results [3], [6], [8], [9].

In essence, anomaly-based intrusion detection can be considered as a classification problem, one that determines network attacks by classifying network traffic into normal and abnormal connections [10]–[13]. Accordingly, supervised learning techniques, such as Bayesian methods, Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), *k*-nearest neighbors (*k*-NN), decision trees, are promising methods for facilitating the development of IDSs [11], [14]–[16]. In the studies of IDS, hybrid approaches, such as ensemble or hybrid classifiers, have become the mainstream, since they are superior to single classification technique in terms of accuracy [9], [17], [18]. The intuition behind a hybrid approach is to enhance the performance of an IDS by combining several machine learning and data mining techniques.

However, there are several limitations in some existing studies. First, exact intrusion information is not reported. Some intrusion detection methods only determine the occurrence of attacks, but do not provide their types. Actually, exact intrusion information is very important for network administrators to take relevant security actions. The second limitation is low detection performance for low-frequency attacks. The reason is that the intrusion detection dataset is very imbalanced (see Table 2). Compared with high-frequency attacks, low-frequency attacks have few instances and may be considered as outliers. Low-frequency attacks, e.g., user to root (U2R) attacks, may have more serious threats than high-frequency ones, e.g., Probe attacks. Thus, detecting low-frequency attacks with high performance is critical for an IDS. The last limitation is too many parameters. Some intrusion detection models, especially hybrid models, have many parameters. Setting values for those parameters is not easy. Some studies search for the best values by means of an optimization algorithm, such as the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) algorithm. However, this policy will increase the training time, and the obtained values are not necessarily optimal. Unoptimized values may affect detection performance negatively. Therefore, reducing the number of parameters in intrusion detection models is necessary.

In this work, we propose an effective hybrid approach based on binary classification and *k*-NN [19] techniques to detect network intrusion. The detecting procedure of the proposed approach is composed of two steps. First, step 1 makes use of several binary classifiers (BCs) to identify abnormal connections and detect their types. In step 1, one BC is in charge of distinguishing normal and abnormal behaviors, and other BCs are responsible for classifying abnormal behaviors. Due to the working mechanism of the proposed method, there may be a group of connections whose classes are still uncertain after step 1. Next, those connections are classified by means of *k*-NN in step 2. Afterwards, abnormal connections will be reported to network administrators with their attack types.

In the proposed method, we consider intrusion detection as a binary classification problem in step 1. A classification problem with only two classes is known as a *binary classification problem*. In contrast, when the class number is greater than two, the classification problem is referred to as a *multiclass classification problem*. Essentially, intrusion detection is a multiclass classification problem. However, in this paper, we employ several independent BCs to take over this job. By converting intrusion detection into a binary classification problem, we can reduce the negative impact caused by the imbalance of the intrusion detection dataset. In the proposed method, one BC concerns one class. Therefore, it can address classes with very few representative examples. In this paper, we adopt the C4.5 algorithm [20], a non-parametric decision tree algorithm (see Subsection II-B), to learn those BCs. In consequence, there is only one parameter in our model, i.e., k in *k*-NN. In Subsection V-A, we will show reasonable values of k.

The performance of our hybrid method is evaluated by conducting experiments on the NSL-KDD benchmark dataset [21]. First, we analyze the results of each step of our method. Then, the detection performances of our method and five supervised learning methods are compared in terms of accuracy, precision, detection rate, F1-score, and false alarm rate. The experimental results demonstrate that the proposed method has the ability to report reliable results.

The rest of the paper is organized as follows. The related work is described in section II. Section III provides insight into the benchmark dataset and evaluation criteria. Section IV introduces the proposed hybrid method. In Section V, the experimental settings and performance analysis of the proposed method are presented. Finally, Section VI concludes this work.

II. RELATED WORK

A. HYBRID METHODS

Aburomman and Ibne Reaz [10] developed an ensemble construction method based on SVM, *k*-NN and PSO for intrusion detection. Six SVM experts and six *k*-NN experts were trained in their method, and two ensemble classifiers were generated by combining the opinions of 12 experts with weighted majority voting. Weights of experts were generated by PSO. In the first ensemble, the parameters of PSO were manually selected, and in the second, those parameters were optimized using local unimodal sampling. Wang *et al.* [22] presented an ensemble classifier that was applied to anomaly intrusion detection based on fuzzy clustering (FC) and ANN. In that work, the FC technique was used to generate different training sets, and the ANN method was adopted to train different prediction models based on the generated training sets. Finally, they employed a fuzzy aggregation module to aggregate the results of all models. Eesa *et al.* [23] proposed a hybrid intrusion detection model. They used the Cuttlefish algorithm (CFA) as a search strategy to produce the optimal subset of features, and a decision tree algorithm as a detection technique on the optimal feature subset. Kuang *et al.* [9] presented an intrusion detection model based on SVM and kernel Principal Component Analysis (KPCA) with GA.

They adopted KPCA to reduce the dimensions of feature vectors and SVM to identify attack activities. To improve the detection performance, they developed an improved radial basis kernel function for SVM. The parameters of SVM were optimized by GA. De la Hoz *et al.* [24] proposed a hybrid model to solve the network intrusion detection problem. In that paper, a multi-objective optimization approach, i.e., the NSGA-II algorithm [25], was applied to feature selection, and Growing Hierarchical Self-Organizing Maps (GHSOMs) [26] were used for both anomaly detection and attack classification. De la Hoz *et al.* [27] presented another anomaly detection approach by hybridizing Principal Component Analysis (PCA), Fisher Discriminant Ratio (FDR), and Probabilistic Self-Organizing Maps (PSOMs). In their study, PCA and FDR were considered for feature selection and noise removal, and a PSOM was used to distinguish normal and abnormal connections. Erfani *et al.* [28] designed a hybrid intrusion detection model in which an unsupervised deep belief network (DBN) [29] was used to learn robust features, and a one-class SVM (1SVM) [30] was adopted to train the detection model. Singh *et al.* [31] presented a technique based on the Online-Sequential Extreme Learning Machine (OS-ELM) to handle intrusion detection. In the proposed technique, alpha profiling and beta profiling were used to reduce the time complexity and size of the training dataset, respectively. An ensemble feature selection technique based on Filtered, Correlation and Consistency was adopted to discard irrelevant features. Bostani and Sheikhan [32] proposed an intrusion detection approach based on a modified Optimum-path forest (OPF) model [33]. This approach employed k-means to partition the original training set into k different homogeneous training subsets, which would be used as the training sets of OPFs. To speed up the OPF, the concepts of centrality and prestige in social network analysis were used to prune training sets by identifying the most informative samples. Karami and Gueerero-Zapata [34] presented a fuzzy anomaly detection system for Content-Centric Networks [35]. The training phase hybridized PSO and k-means to determine the optimal number of clusters, and the detection phase employed a fuzzy approach to detect anomalies. In Table 1, we summarize some recent related studies.

B. C4.5 ALGORITHM

The decision tree technique is a non-parametric supervised learning method used in various disciplines such as statistics, pattern recognition and machine learning; it is independent of domain knowledge and can cope with high-dimensional data. A decision tree is a classifier depicted by a flowchart-like tree structure, in which each internal node partitions the instance space according to the value of a feature, each branch represents an outcome of the partition, and each leaf node holds a class label for a group of instances. The topmost node in a decision tree is the root node. A path from the root to a leaf denotes a classification rule. Using a decision tree classifier, new samples are classified by walking down this tree from the root to a leaf based on their feature values.

A decision tree classifier is learned by constructing a decision tree from a class-labeled training set. A decision tree is constructed by partitioning the training set into subsets according to a *feature selection metric*. The process is repeated on each outcome of the previous partition in a recursive manner, unless the instances in this outcome have the same label or other stopping criteria are reached. Therefore, a key problem is which feature should be selected to best split the set of instances in each recursion. Different implementations of decision tree use different feature selection metrics to measure which is "best." For example, ID3 makes use of information gain [47], C4.5 employs the gain ratio [20], and CART uses the Gini index [48].

C4.5 is a decision tree algorithm developed by Quinlan in 1993 [20], and it was voted one of the top-10 data mining algorithms [49]. C4.5 improved Quinlan's earlier ID3 algorithm [47]. Some of the improvements are as follows:

(1) Use of the gain ratio instead of information gain as the feature selection metric to avoid bias toward features with a large number of values;
(2) Can address both continuous and discrete features;
(3) Can handle features with missing values;
(4) Remove branches that do not help to avoid over-fitting when building the tree.

As mentioned above, C4.5 uses the gain ratio to select the "best" splitting feature when building a decision tree. We now provide definitions of the gain ratio and other related conceptions.

Let $\mathcal{D}$ be a class-labeled training set whose samples fall into h classes: $C_1, \cdots, C_h$. The excepted *information* (i.e., entropy) needed to classify a sample in $\mathcal{D}$ is defined in

$$Info(\mathcal{D}) = -\sum_{i=1}^{h} p_i \log_2(p_i), \quad (1)$$

where p_i is the probability that an arbitrary sample in $\mathcal{D}$ belongs to class C_i. $Info(\mathcal{D})$ is the average amount of information needed to search for the class label of a sample in $\mathcal{D}$.

Suppose we can partition $\mathcal{D}$ into k disjoint subsets: $\mathcal{D}_1, \cdots, \mathcal{D}_k$ based on the values of feature A. After this partitioning, the amount of information still required to arrive at an exact classification is defined in

$$Info_A(\mathcal{D}) = \sum_{j=1}^{k} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times Info(\mathcal{D}_j). \quad (2)$$

The amount of information reduced after splitting $\mathcal{D}$ on feature A is measured by *information gain*. That is,

$$Gain(A) = Info(\mathcal{D}) - Info_A(\mathcal{D}). \quad (3)$$

The ID3 algorithm uses information gain as the feature selection metric [47]. However, this metric biases toward partitions with many outcomes [49]. To overcome this weakness,

TABLE 1. Summary table of some recent related work.

Work	Year	Learning Technique	Feature Selection Method	Optimization Method	Ensemble Method
Wang *et al.* [22]	2010	FC, ANN	-	-	fuzzy aggregation
Koc *et al.* [36]	2012	HNB	CFS, CONS, INTERACT	-	-
Lin *et al.* [18]	2012	SVM, DT	SA	SA	-
Meng and Kwok [37]	2013	SVM, DT, k-NN	Manual selection of eight features	-	Majority voting
De la Hoz *et al.* [24]	2014	GHSOM	NSGA-II	-	-
Kuang *et al.* [9]	2014	SVM	KPCA	GA	-
De la Hoz *et al.* [27]	2015	PSOM	PCA, FDR	-	-
Eesa *et al.* [23]	2015	DT	CFA	-	-
Lin *et al.* [8]	2015	k-means, k-NN	6 features in [38] , 19 features in [39]	-	-
Singh *et al.* [31]	2015	OS-ELM	Ensemble feature selection technique	-	-
Karami and Guerrero-Zapata [34]	2015	k-means	-	PSO	-
Karami and Guerrero-Zapata [40]	2015	RBF-NN	-	NSGA-II, PSO	-
Aburomman and Ibne Reaz [10]	2016	SVM, k-NN	-	LUS	WMA, PSO
Ji *et al.* [41]	2016	DT, SVM, NB, NN	ANOVA	-	-
Erfani *et al.* [28]	2016	1SVM	DBNs	-	-
Bamakan *et al.* [42]	2016	MCLP, SVM	TVCPSO	TVCPSO	-
Canbay and Sagiroglu [43]	2016	k-NN	GA	-	-
Dash [44]	2017	ANN	-	GS, GSPSO	-
Al-Yaseen *et al.* [45]	2017	k-means, SVM, ELM	-	-	-
Bostani and Sheikhan [32]	2017	OPF, k-means	-	-	-
Ashfaq *et al.* [46]	2017	SSL, NNR_w	-	-	-

FC: fuzzy clustering
ANN: artificial neural networks
HNB: hidden naïve bayes
CFS: correlation-based feature selection
CONS: consistency-based feature selection
SVM: support vector machines
DT: decision tree
SA: simulated annealing
GHSOM: growing hierarchical self-organizing maps
NSGA-II: non-dominated sorting genetic algorithm II
KPCA: kernel principal component analysis
GA: genetic algorithm
PSOM: probabilistic self-organizing maps
PCA: principal component analysis
FDR: fisher discriminant ratio
CFA: cuttlefish algorithm
OS-ELM: online sequential ELM
ELM: extreme learning machines
PSO: particle swarm optimization
RBF-NN: radial basis function neural networks
LUS: local unimodal sampling
WMA: weighted majority algorithm
NB: naïve bayes
NN: neural networks
ANOVA: analysis of variance
1SVM: one-class SVM
DBNs: deep belief networks
MCLP: multiple criteria linear programming
TVCPSO: time-varying chaos PSO
GS: gravitational search
GSPSO: hybrid GS and PSO
OPF: optimum-path forest
SSL: semi-supervised learning
NNR_w: neural network with random weights

C4.5 adopts the measure of *gain ratio* [20]. That is,

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}, \tag{4}$$

where $SplitInfo(A)$ is the "split information" that describes the potential information generated by splitting $\mathcal{D}$ into v outcomes according to feature A. It is defined in

$$SplitInfo_A(\mathcal{D}) = -\sum_{j=1}^{k} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \log_2(\frac{|\mathcal{D}_j|}{|\mathcal{D}|}). \tag{5}$$

In C4.5, the feature with the maximum value of gain ratio is selected as the partitioning feature.

C. k-NEAREST NEIGHBORS ALGORITHM

The k-nearest neighbors (k-NN) algorithm is a simple and effective supervised learning technique [19] and was also elected as one of the top-10 data mining algorithms [49]. This algorithm assigns a class label to an unlabeled object based on the class labels of its k nearest neighbors. Consider a class-labeled dataset $\mathcal{D}$ and an unlabeled object o. To predict

TABLE 2. Number of instances in NSL-KDD.

	Normal	DoS	Probe	U2R	R2L	Total
KDDTrain$^+$	67,343	45,927	11,656	52	995	125,973
KDDTest$^+$	9,711	7,458	2,421	200	2,754	22,544
KDDTest^{-21}	2,152	4,342	2,402	200	2,754	11,850

the label of o, k-NN computes the distance (or similarity) between o and all samples in $\mathcal{D}$ to determine the k nearest neighbors of o, denoted as $kNN(o)$. Then, o is labeled according to the majority class of its k nearest neighbors. That is,

$$l(o) = \arg\max_{c} \sum_{s \in kNN(o)} I(c = l(s)), \quad (6)$$

where $l(o)$ is the predicted label of o, c is a class label, and $l(s)$ is the class label of o's neighbor s. In (6), $I(\cdot)$ is an indicator function that returns 1 if c equals to $l(s)$, and 0 otherwise.

One of the key elements in k-NN is the distance measure. We use the Spearman rank correlation coefficient (*Spearman coefficient* for short) to measure the distance between two samples in this paper. The Spearman coefficient is a non-parametric and distribution-free statistical method for measuring the rank correlation between two independent variables, which is appropriate for continuous, discrete and ordinal variables [50].

Let $X = \{x_i\}_{i=1}^n$, $Y = \{y_i\}_{i=1}^n$ be two variables and L_X, L_Y be the corresponding lists stored x_i, y_i in descending order, respectively. The ranks of x_i, y_i in L_X, L_Y are respectively marked as x_i', y_i'. The distance between X, Y measured by the Spearman coefficient is defined as

$$\rho(X, Y) = 1 - \frac{\sum_{i=1}^n (x_i' - \bar{x'})(y_i' - \bar{y'})}{\sqrt{\sum_{i=1}^n (x_i' - \bar{x'})^2 \sum_{i=1}^n (y_i' - \bar{y'})^2}}, \quad (7)$$

where $\bar{x'} = \frac{1}{n}\sum_{i=1}^n x_i'$ and $\bar{y'} = \frac{1}{n}\sum_{i=1}^n y_i'$.

III. DATASET AND EVALUATION CRITERIA

A. DATASET

In the field of intrusion detection, only a few public datasets are available to evaluate the performance of IDSs. The NSL-KDD dataset [21] is an effective benchmark, which improved the famous KDDCup99 dataset by solving some inherent problems existing in it. The NSL-KDD dataset provides one training set, KDDTrain$^+$, and two testing sets, KDDTest$^+$ and KDDTest^{-21}. KDDTrain^{-21}, a subset of the KDDTest$^+$, does not include records that are correctly classified by all 21 classifiers. Table 2 lists the numbers of instances in the training set and testing sets. As seen in Table 2, the number of instances in the NSL-KDD dataset is in the reasonable range, which makes it affordable to conduct experiments on the entire dataset. For the KDDCup99 dataset, researchers have usually run experiments on randomly selected small portion, which may cause inconsistent evaluation results.

Each instance in the NSL-KDD dataset consists of 41 *input features* and a *class* label. The class label specifies whether the status of an instance is either normal or attack. Attacks in NSL-KDD are grouped into four types: denial of service (DoS), Probe, user to root (U2R), and remote to local (R2L). Detailed information regarding those features and attack types can be found in Ref. [51]. Table 2 also shows the numbers of instances of normal events and different attack types. Obviously, R2L and U2R are low-frequency attacks in KDDTrain$^+$.

TABLE 3. Confusion matrix.

		Predicted	
		Attack	Normal
Actual	Attack	TP	FN
	Normal	FP	TN

The 41 features in the NSL-KDD dataset contain three symbolic, two binary, and 36 continuous features. Symbolic features should be converted into numeric features, as most classifiers only accept numeric values. In this paper, we adopt the simple scheme used in Refs. [10], [52] to handle symbolic features. The scheme maps symbolic values to integer values with a range from 1 to M, where M is the number of distinct symbols for a feature. For class labels, Normal is mapped to 0, DoS to 1, Probe to 2, R2L to 3, and U2R to 4.

B. EVALUATION CRITERIA

In this paper, the performance of intrusion detection models is evaluated by five widely used measures: *accuracy*, *precision*, *detection rate*[1] (DR), *F1-score*, and *false alarm rate* (FAR) [22], [31], [42]. The definitions of these measures are based on a confusion matrix, as shown in Table 3. That is,

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \quad (8)$$

$$Precision = \frac{TP}{TP + FP}, \quad (9)$$

$$DR = \frac{TP}{TP + FN}, \quad (10)$$

$$FAR = \frac{FP}{FP + TN}, \quad (11)$$

$$F1 - score = \frac{2 \times Precision \times DR}{Precision + DR}. \quad (12)$$

In Table 3, TP is the number of attacks correctly classified as abnormal, TN is the number of normal connections correctly classified as normal, FP is the number of normal

[1] also called *recall*

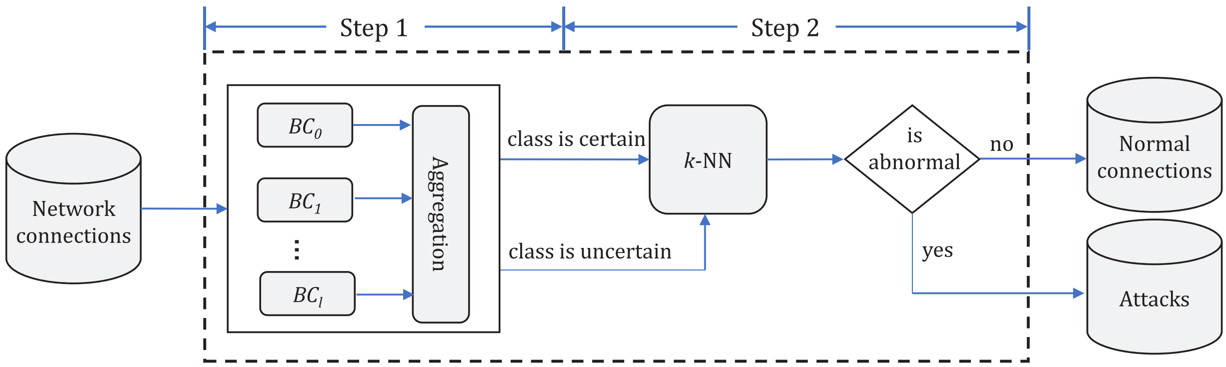


FIGURE 1. **Framework of the detection procedure of the proposed method.**

connections incorrectly classified as abnormal, and FN is the number of attacks incorrectly classified as normal.

IV. METHODOLOGY

In computer systems, vulnerabilities always exist, and new vulnerabilities will be discovered continuously. This results in various network intrusions that try to compromise the confidentiality, integrity, and/or availability of computer systems. An IDS is a security mechanism to minimize those risks. Attracted by the ability to identify known and unknown network intrusions, researchers have paid more attention to anomaly-based detection approaches [24], [41], [53]. Furthermore, to protect network security, identification of the type of an intrusion is more valuable than just determining that an attack occurred. It is crucial to provide the exact intrusion information to network administrators so that they can take relevant actions to secure the computing infrastructure.

To implement effective intrusion detection, we propose a two-step hybrid method in this paper. The overview of its detection procedure is shown in Fig. 1. In step 1, the proposed method employs $(l + 1)$ BCs and one aggregation module to classify network connections. For a connection, each of the $(l + 1)$ BCs may assign a class label to it, and then the aggregation module summarizes those results and makes a final decision. After step 1, those connections whose classes are uncertain will be further classified in step 2 by k-NN.

A detailed description of the proposed method will be presented in the following subsections.

A. STEP 1

In step 1, the proposed method uses several BCs and one aggregation module to determine the class of a network connection. Each BC is responsible for one class. That is, given a connection x, $BC_i(i \in [0, l])$ detects whether or not x belongs to class i. The number of BCs, denoted as $(l+1)$, is equal to the number of classes. In this study, there are five classes in the benchmark dataset, i.e., Normal, DoS, Probe, R2L, and U2R. Among the five classes, Normal denotes normal events, and the other four are attack types. As described in Subsection III-A, the five classes are mapped to 0, 1, 2, 3, and 4, respectively. Accordingly, BC_0 is used to distinguish normal and abnormal connections, and the other BCs are used to predict attack types. Specifying the exact type of an attack is quite necessary for an IDS. System administrators can take relevant actions to secure the computing infrastructure according to attack types.

In the proposed hybrid approach, $(l+1)$ BCs determine the classes of connections separately. One BC takes charge of one class. For a connection x, if BC_i considers that it belongs to class i, we define $BC_i(x) = 1$, and otherwise $BC_i(x) = 0$. The aggregation module collects the outcomes of $(l+1)$ BCs, and then makes a final decision. There are three cases:

(1) $\sum_{i=0}^{l} BC_i(x) = 1$. That is, only one class label is assigned to x.
(2) $\sum_{i=0}^{l} BC_i(x) = 0$. That is, no class label is assigned to x.
(3) $\sum_{i=0}^{l} BC_i(x) \geq 2$. That is, x is assigned two or more class labels.

In case 1, *x's class is certain*. In case 2, x's class is unknown, and in case 3, x's class is ambiguous. For cases 2 and 3, we define that *x's class is uncertain*.

In general, the network intrusion dataset is imbalanced. For instance, in KDDTrain$^+$, U2R has very low frequency. However, the severity of the impact of U2R is higher than that of DoS or Probe. In our hybrid approach, one BC focuses only on the focal attack type. Hence, the bias that towards frequent types will be reduced. Additionally, only when $\sum_{i=0}^{l} BC_i(x) = 1$, the aggregation module considers that x's class is certain. If two or more BCs label x with the corresponding classes, x's class is uncertain. Based on this strategy, the precision of the proposed method can be improved.

Algorithm 1 shows the procedure for detecting the class label of a network connection. By using Algorithm 1, connections can be partitioned into two groups: the certain class label group and uncertain class label group. These two groups will be sent to step 2 separately to further analyze the classes of connections in the uncertain class label group. This work will be described in Subsection IV-B.

In this paper, we employ the C4.5 algorithm [20] to train the BCs. C4.5 is a multiclass classification technique, and it is competent for binary classification of course. The description

Algorithm 1 Class Detection

Input: A network connection x
Output: Class label of x
// Predict x's class by each BC separately
for $i = 0 \rightarrow l$ **do**
  Send x to BC_i ;
end for
// Make a final decision in aggregation module
Aggregate the outcomes of all BCs;
if x's class is certain **then**
  return x's class;
else
  return uncertain;
end if

Algorithm 2 Train $BC_1, \cdots, BC_l$

Input: Training set $\mathcal{D}_A$
Output: A group of BCs
for $i = 1 \rightarrow l$ **do**
  Represent $\mathcal{D}_A$ as $\mathcal{D}_A^i$;
  for each $x \in \mathcal{D}$ **do**
    if x belongs to class i **then**
      $l'(x) = 1$
    else
      $l'(x) = 0$
    end if
  end for
  $BC_i = C4.5(\mathcal{D}_A^i)$;
end for
return $\{BC_i\}_{i=1}^{l}$;

of C4.5 was given in Subsection II-B. In addition to the advantages mentioned in Subsection II-B, one significant reason that we use C4.5 is that it incorporates a feature selection technique. In many research studies, feature selection is a critical step before training an intrusion detection model [9], [18], [24], [27]. However, when using C4.5 as in this paper, we no longer need to perform feature selection in advance.

In this study, we train BC_0 and other BCs, separately. To learn BC_0 using C4.5, we need to represent the intrusion detection dataset as a binary classification dataset, and then apply C4.5 on the new dataset. The formal description of the learning process is as follows.

Given an intrusion detection training set $\mathcal{D}$, $x \in \mathcal{D}$ is an instance. Let $l(x)$ denote the class label of x and $l'(x)$ be its new class label. The value of $l'(x)$ is defined as

$$l'(x) = \begin{cases} 1, & \text{if } l(x) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Note that the class label of Normal events is mapped to 0. For each instance in $\mathcal{D}$, we replace its class label in light of (13). Then, we mark the new dataset as $\mathcal{D}'$. After that, we run C4.5 on $\mathcal{D}'$ to learn BC_0. That is,

$$BC_0 = C4.5(\mathcal{D}'). \tag{14}$$

To train $BC_1, \cdots, BC_l$, we generate a new training set (marked as $\mathcal{D}_A$), that contains all abnormal instances of the original intrusion detection training set. Because $BC_1, \cdots, BC_l$ are adopted to predict attack types of possible abnormal connections, we use the new training set, $\mathcal{D}_A$, rather than the original training set to train them. The learning of $BC_1, \cdots, BC_l$ is similar to that of BC_0; the procedure is shown in Algorithm 2. In Algorithm 2, $l'(x)$ denotes x's new label in $\mathcal{D}_A^i$.

B. STEP 2

After step 1, there may exist a group of connections whose classes are uncertain. For those connections, their classes are further identified in step 2. In this step, we employ the k-NN algorithm (see Subsection II-C) to undertake this task. k-NN is a type of lazy learning technique in which all computations are deferred until a query is given. In this step, for a connection whose class is uncertain, we search its k nearest neighbors from the connections whose classes are certain, and then determine its class by majority vote of its k nearest neighbors. As mentioned in Subsection II-C, the distance between two connections is measured by the Spearman coefficient [50].

When finishing this operation, all connections' classes are determined. For abnormal behaviors, the connections will be prevented, and their attack types will be reported to administrators.

V. EXPERIMENT RESULTS

To evaluate the detection performance of the proposed hybrid method, a series of experiments were conducted on the NSL-KDD dataset. The training set and testing sets were described in Subsection III-A. All experiments were implemented in the MATLAB 2012b environment.

A. SETTING OF PARAMETER k

In our hybrid method, a primary parameter is the k in the k-NN algorithm used in step 2, which affects the detection performance of our method. For each connection in the uncertain class label group, step 2 searches its k nearest neighbors from the certain class label group, then identifies its class label according to the k nearest neighbors. In this paper, the optimal value of k is decided based on the detection accuracy on two testing sets. Fig. 2 heuristically shows the accuracy with different values of k. We can see from Fig. 2 that the accuracy on KDDTest^{+} and KDDTest^{-21} have roughly the same trend. The best possible solution for the value of k ranges from 10 to 16. In this study, we set $k = 15$.

B. PERFORMANCE ANALYSIS

The proposed hybrid method is composed of two steps. In this subsection, we experimentally analyze the performance of these two steps as well as the combination of the two steps (i.e., the proposed method).

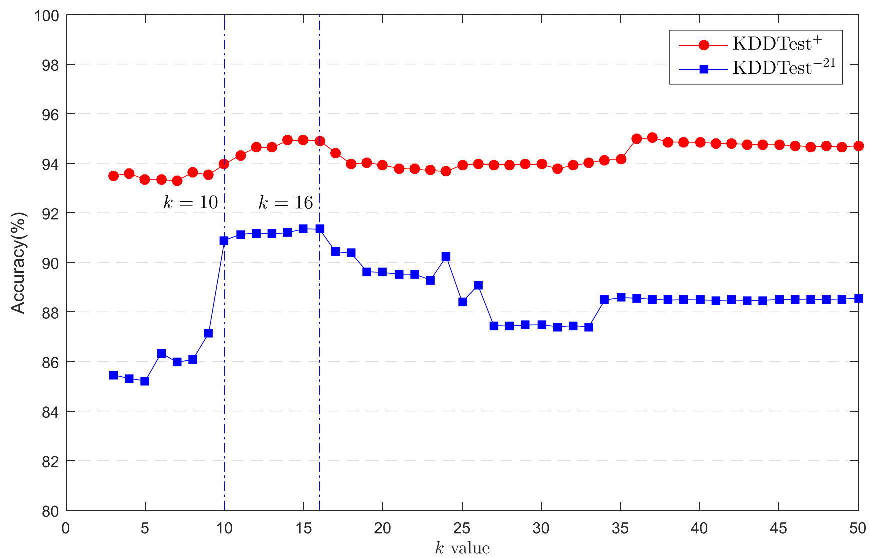


FIGURE 2. Values of k vs. accuracy for identifying the optimal value of k. The best possible k ranges from 10 to 16. This study sets $k = 15$.

TABLE 4. Confusion matrix over KDDTest$^+$.

	Predicted	Step 1		Step 2		Step 1 & 2	
		Attack	Normal	Attack	Normal	Attack	Normal
Actual	Attack	8,634	201	3,208	790	11,842	991
	Normal	41	2,138	113	7,419	154	9,557

TABLE 5. Confusion matrix over KDDTest^{-21}.

	Predicted	Step 1		Step 2		Step 1 & 2	
		Attack	Normal	Attack	Normal	Attack	Normal
Actual	Attack	5,499	201	3,284	714	8,783	915
	Normal	35	540	75	1,502	110	2,042

TABLE 6. Performance of the proposed method for detecting abnormal connections.

Testing Set	Step	Accuracy	Precision	DR	F1-score	FAR
KDDTest$^+$	1	97.80%	99.53%	97.72%	98.62%	1.88%
	2	92.17%	96.60%	80.24%	87.66%	1.50%
	1 & 2	94.92%	98.72%	92.28%	95.39%	1.59%
KDDTest^{-21}	1	96.24%	99.37%	96.47%	97.90%	6.09%
	2	85.85%	97.77%	82.14%	89.28%	4.76%
	1 & 2	91.35%	98.76%	90.57%	94.49%	5.11%

Tables 4 and 5 list the confusion matrices obtained by the proposed method over KDDTest$^+$ and KDDTest^{-21}, respectively. In both tables, we only consider normal events and attack activities, that is, we do not care about the types of abnormal connections. In Table 4, the number of attack connections correctly identified in step 1 is 8,634, while the number of incorrectly detected is merely 41. For step 2, the corresponding numbers are 3,208 and 113, respectively. By combining the results of steps 1 and 2, the proposed method (i.e., Step 1 & 2 in Table 4) successfully recognizes 11,842 attack connections on KDDTest$^+$; only 154 normal connections are incorrectly predicted as attacks. Thus, the precision of the proposed method is up to 98.72% (see Table 6). For normal events, the proposed method pinpoints 9,557 out of 9,711 instances such that the false alarm rate is as low as 1.59% (see Table 6). In Table 5, the proposed

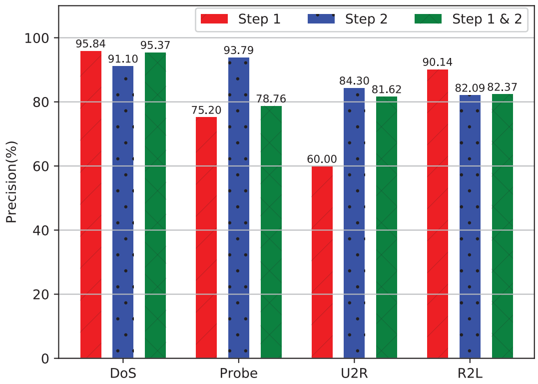
TABLE 7. Numbers of instances of four attack types detected in each step.

			DoS	Probe	U2R	R2L	Total
KDDTest^{+}	Step 1	TP	6,149	1,634	9	64	7,856
		FP	267	539	6	7	819
	Step 2	TP	645	483	102	1,623	2,853
		FP	63	32	19	354	468
KDDTest^{-21}	Step 1	TP	3,033	1,615	9	64	4,721
		FP	290	510	6	7	813
	Step 2	TP	645	485	102	1,684	2,916
		FP	15	24	18	386	443

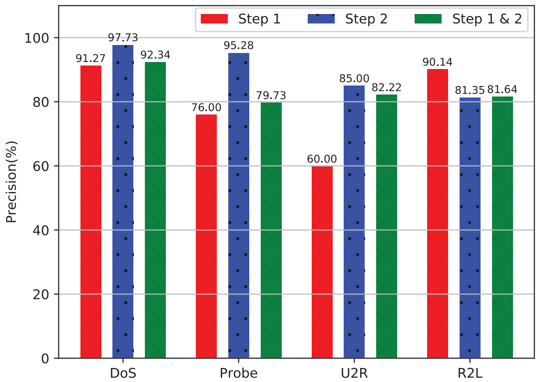
method discovers 8,893 attacks. Among them, 8,783 are real abnormal behaviors. Thus, its precision is up to 98.76% (see Table 6). Based on the results in Tables 4 and 5, we obtain the overall performance of the proposed method in detection of abnormal activities in terms of accuracy, precision, detection rate, F1-score, and false alarm rate. The results are shown in Table 6. On both testing sets, step 1 achieves very high detection performance with respect to accuracy, precision, detection rate, and F1-score. However, the false alarm rates of step 2 are lower than those of step 1 on two testing sets. On KDDTest^{+}, the precision and detection rate of step 1 are respectively up to 99.53% and 97.72%, and its false alarm rate is as low as 1.88%. These results indicate that the strategy of using binary classifiers and the aggregation module in step 1 is successful. Although the overall performance of step 2 is lower than that of step 1, it is decent. The reason is that the instances classified in step 2 are those that cannot be identified in step 1. Benefitting from the contributions of steps 1 and 2, the proposed method can achieve reliable results. In addition, the performance on KDDTest^{-21} is lower than that on KDDTest^{+}. That is a normal phenomenon, because KDDTest^{-21} removes 10,694 instances that are easily classified from KDDTest^{+} (see Table 2).

Furthermore, we list the numbers of instances of four attack types detected in each step in Table 7. In Table 7, TP indicates the numbers of instances whose types are correctly assigned, and FP shows the numbers of instances whose types are wrongly assigned. We can observe from Table 7 that step 1 identifies more instances than step 2 for DoS and Probe attacks but recognizes less instances than step 2 for R2L and U2R attacks on both testing sets. This scenario results from the imbalance of the NSL-KDD dataset. For DoS and Probe attacks, there are enough instances to train the corresponding BCs used in step 1. Therefore, step 1 can correctly identify most of instances whose types are DoS and Probe. Alternatively, R2L and U2R are two low-frequency attacks in KDDTrain^{+}. However, these two types have more instances in the testing sets than in the training set. In particular, R2L attacks are not low frequency. Thus, step 1 only detects a small portion of instances for R2L and U2R types. However, depending on the results of step 1, step 2 correctly identifies a great deal of instances that belong to these two attack types. In addition, the numbers of instances detected in step 2 for each attack type on both testing sets are the same or very close. This phenomenon is reasonable because the instances that are not easy to detect are the same in both testing sets.

Table 7 depicts the detection performance of two steps for four attack types in terms of quantity. Next, we describe the effectiveness in terms of precision. The precision of the proposed method for each attack type is calculated according to Table 7. The corresponding results are shown in Fig. 3. On the whole, the results are fairly good. As shown in Fig. 3, step 1 yields a precision of 95.84% and 91.27% for DoS on KDDTest^{+} and KDDTest^{-21}, respectively. Considering the results in Table 7, we can conclude that step 1 of the proposed method is very effective for detection of DoS attacks. For this type, the instances that are difficult to detect are further classified in step 2. For those instances, the precision of step 2 is 91.10% and 97.73% on KDDTest^{+} and KDDTest^{-21}, respectively. By integrating these two steps, the proposed method (i.e., Step 1 & 2 in Fig. 3) achieves the precision of 95.37% and 92.34% on KDDTest^{+} and KDDTest^{-21}, respectively. Consequently, the proposed method is highly competent in detection of DoS attacks. For Probe, the precision of step 1 is a little low, however, step 2 obtains high precision. The practice effectiveness of step 2 yields a precision of the proposed method of approximately to 80% in the detection of Probe attacks. For U2R attacks, step 1 only detects a small portion of instances (see Table 7), and its precision is just 60%. Fortunately, step 2 makes up for the weakness. Its precision is up to 84.3% and 85% on KDDTest^{+} and KDDTest^{-21}, respectively. Depending on the contribution of step 2, the proposed method presents approximately 82% of precision for U2R attacks on both testing sets. This is a quite satisfactory result compared with other methods (see Subsection V-C). In the detection of R2L attacks, step 1 achieves higher precision than step 2; conversely, step 2 detects far more instances than step 1 (see Table 7). Taking the low frequency of R2L over the training set into account, we deem that the performance of both steps is sufficient. According to the results of the two steps, the proposed method manifests a precision of approximately 82% for R2L attacks on both testing sets. In comparison with other methods (see Subsection V-C), this result is extremely decent. In summary, the combination



(a)



(b)

FIGURE 3. Precision obtained by the proposed method for four attack types. (a) KDDTest^{+}. (b) KDDTest^{-21}.

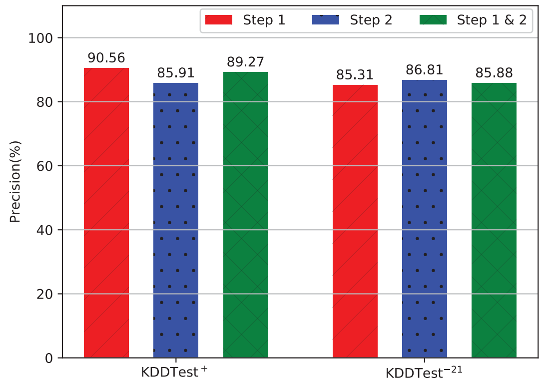


FIGURE 4. Total precision obtained by the proposed method for four attack types.

of steps 1 and 2 makes the proposed method an effective intrusion detection model.

In Fig. 3, step 2 seems more effective than step 1. Actually, this is not the case. In Fig. 4, we show the total precision of the proposed method for detection of four attack types. The precision is also computed based on Table 7. In Fig. 4, the precision of step 2 is slightly higher than that of step 1 on KDDTest^{-21} but lower than that of step 1 on KDDTest^{+}. Note that step 1 detects far more instances than step 2 (see Table 7) and sends the remaining parts to step 2. Step 2 cannot work independently; it is based on the outcome of step 1. Thus, the two steps of the proposed method are an organic whole. By combining the two steps, the proposed method demonstrates pleasing performance.

Finally, we list the detection rates of the proposed method for four attack types in Table 8 to comprehensively exhibit its performance. As we can see from Table 8, for the two low-frequency attack types, i.e., R2L and U2R, the corresponding detection rates are approximately 62% and 55.5%, respectively. Considering the precision for these types presented in Fig. 3, we can conclude that the proposed method is fairly effective in detection of R2L and U2R attacks. However, we hope to further improve the detection rate for U2R attacks in our future work. For the other two types, the proposed

TABLE 8. Detection rates of the proposed method for four attack types.

Testing set	DoS	Probe	U2R	R2L
KDDTest^{+}	91.10%	87.44%	55.50%	61.26%
KDDTest^{-21}	84.71%	87.43%	55.50%	63.47%

method provides better detection rates than for R2L and U2R. On the basis of the results in Fig. 3 and Table 8, we can see that the proposed method is adept at detecting DoS attacks. For Probe attacks, the performance of the proposed method has some room for improvement.

C. PERFORMANCE COMPARISON

In this subsection, we evaluate the detection performance of the proposed method by means of experimental comparison. The techniques for comparison include C4.5, Random Forests (RF), k-NN, Backward Propagation Neural Network (BPNN), and Naïve Bayes (NB), which are often used for intrusion detection. Parameter settings for the competing methods are as follows:

- The default settings in MATLAB 2012b are used for C4.5 and RF.
- k is set to 15 for k-NN.
- The number of hidden units is 18 for BPNN.
- A multinomial distribution is used for NB.

For the sake of convenience, we name our method BC+k-NN. Table 9 lists the experimental results on KDDTest^{+} and KDDTest^{-21} in terms of accuracy, precision, detection rate (DR), F1-score, and false alarm rate (FAR). The best results on each testing set are highlighted in boldface. What is noteworthy is that the values in Table 9 were calculated based on the results obtained by each method for distinguishing normal and abnormal events. It is evident from Table 9 that our method (i.e., BC+k-NN) achieves the highest accuracy, precision, detection rate, and F1-score, as well as the lowest false alarm rate. On KDDTest^{+}, the accuracy and detection rate of our method are 94.92% and 92.28%, respectively, whereas the highest accuracy and detection rate among the results of other methods are only 81.01% and 69.02%

TABLE 9. Performance comparsion of six methods in detection of abnormal connections.

Testing set	Method	Accuracy	Precision	DR	F1-score	FAR
KDDTest$^+$	C4.5	81.01%	96.67%	69.02%	80.54%	3.14%
	RF	77.06%	96.49%	61.96%	75.46%	2.98%
	k-NN	76.70%	97.21%	60.81%	74.82%	2.31%
	BPNN	75.34%	91.74%	62.29%	74.20%	7.41%
	NB	77.75%	93.36%	65.57%	77.04%	6.16%
	BC+k-NN	**94.92%**	**98.72%**	**92.28%**	**95.39%**	**1.59%**
KDDTest^{-21}	C4.5	63.95%	95.08%	59.00%	72.82%	13.75%
	RF	55.34%	94.27%	48.37%	63.93%	13.24%
	k-NN	55.73%	95.54%	48.15%	64.03%	10.13%
	BPNN	57.46%	93.30%	51.73%	66.56%	16.73%
	NB	58.12%	90.20%	54.77%	68.16%	26.81%
	BC+k-NN	**91.35%**	**98.76%**	**90.57%**	**94.49%**	**5.11%**

TABLE 10. Numbers of instances correctly detected by six methods for four attack types.

Testing set	Method	DoS	Probe	U2R	R2L	Total
KDDTest$^+$	C4.5	5,641	1,622	14	55	7,332
	RF	5,910	1,674	6	4	7,594
	k-NN	5,756	1,459	5	83	7,303
	BPNN	5,702	1,896	12	5	7,615
	NB	4,785	**2,173**	10	547	7,515
	BC+k-NN	**6,794**	2,117	**111**	**1,687**	**10,709**
KDDTest^{-21}	C4.5	2,526	1,603	14	55	4,198
	RF	2,692	1,674	6	4	4,376
	k-NN	2,668	1,441	5	83	4,197
	BPNN	2,657	1,863	0	11	4,531
	NB	1,882	**2,157**	10	547	4,596
	BC+k-NN	**3,678**	2,100	**111**	**1,748**	**7,637**

TABLE 11. Detection rates obtained by six methods for four attack types.

Testing set	Method	DoS	Probe	U2R	R2L
KDDTest$^+$	C4.5	75.64%	67.00%	7.00%	2.00%
	RF	79.24%	69.14%	3.00%	0.15%
	k-NN	77.18%	60.26%	2.50%	3.01%
	BPNN	76.45%	78.31%	6.00%	0.18%
	NB	64.16%	**89.76%**	5.00%	19.86%
	BC+k-NN	**91.10%**	87.44%	**55.50%**	**61.26%**
KDDTest^{-21}	C4.5	58.18%	66.74%	7.00%	2.00%
	RF	62.00%	69.69%	3.00%	0.15%
	k-NN	61.45%	59.99%	2.50%	3.01%
	BPNN	61.19%	77.56%	0.00%	0.40%
	NB	43.34%	**89.80%**	5.00%	19.80%
	BC+k-NN	**84.71%**	87.43%	**55.50%**	**63.47%**

(both obtained by C4.5), respectively. For precision, our method achieves a very high result that is up to 98.72%. It seems that the competing methods also get high performance in terms of precision, but their detection rates are somewhat lower than that of the proposed method. Therefore, the best value of F1-score, a measure that synthesizes both precision and detection rate, obtained by those methods is just 80.54%, while our method achieves 95.39% for F1-score. In addition, the proposed method reduces the false alarm rate to 1.59% on KDDTest$^+$. This proves that the detection results of our method are very reliable. On the whole, the corresponding detection performance of all methods on

TABLE 12. Precision obtained by six methods for four attack types.

Testing set	Method	DoS	Probe	U2R	R2L
KDDTest^{+}	C4.5	94.24%	61.02%	50.00%	11.22%
	RF	95.54%	**82.10%**	75.00%	57.14%
	k-NN	94.98%	77.85%	62.50%	**96.51%**
	BPNN	94.99%	70.88%	50.00%	41.76%
	NB	**95.80%**	63.84%	71.43%	91.32%
	BC+k-NN	95.37%	78.76%	**81.62%**	82.37%
KDDTest^{-21}	C4.5	87.98%	60.97%	50.00%	11.22%
	RF	90.76%	**83.83%**	**85.71%**	66.67%
	k-NN	89.86%	78.96%	62.50%	**96.51%**
	BPNN	89.37%	78.15%	0.00%	57.89%
	NB	90.09%	67.36%	71.43%	93.66%
	BC+k-NN	**92.34%**	79.73%	82.22%	81.64%

TABLE 13. F1-scores obtained by six methods for four attack types.

Testing set	Method	DoS	Probe	U2R	R2L
KDDTest^{+}	C4.5	83.92%	63.87%	12.28%	3.39%
	RF	86.63%	75.07%	5.77%	0.29%
	k-NN	85.16%	67.94%	4.81%	5.85%
	BPNN	84.72%	74.41%	10.71%	0.36%
	NB	76.85%	74.61%	9.35%	32.63%
	BC+k-NN	**93.18%**	**82.87%**	**66.07%**	**70.26%**
KDDTest^{-21}	C4.5	70.04%	63.72%	12.28%	3.39%
	RF	73.67%	76.11%	5.80%	0.29%
	k-NN	72.99%	68.18%	4.81%	5.85%
	BPNN	72.65%	77.85%	0.00%	0.79%
	NB	58.53%	76.98%	9.35%	32.77%
	BC+k-NN	**88.36%**	**83.40%**	**66.27%**	**71.42%**

KDDTrain^{-21} is weaker than that on KDDTest^{+}. The reason is that KDDTrain^{-21} does not include the instances that are easy to classify. Even so, our method still achieves commendable results. Its accuracy and F1-score are up to 91.35% and 94.49%, and its false alarm rate is as low as 5.11%. Compared with the other methods, the performance of our method is much improved.

Table 9 displays the overall performance of different intrusion detection methods in distinguishing normal and abnormal instances. In the following, we will present the performance for detecting individual attack types of the proposed method in comparison with other techniques.

Table 10 provides the numbers of instances whose attack types are correctly assigned by all methods. As shown in Table 10, our method correctly recognizes more instances than other methods for DoS, R2L, and U2R attacks. For Probe, Naïve Bayes (NB) identifies the most instances, and our method is next after it. Taking the total instances correctly classified into consideration, our method far outnumbers the others. Table 11 demonstrates the detection rates obtained by the six methods. We can evidently see from Table 11 that the detection rates of competing methods are much lower in comparison with the proposed method for low-frequency types, i.e., R2L and U2R.

Next, we show the precision of the six methods for the four attack types in Table 12. It is interesting to note that k-NN acquires the best precision for R2L attacks on both testing sets. However, this result does not indicate that k-NN is effective in the detection of R2L attacks because its detection rate is just 3.01% on both testing sets (see Table 11). Although Naïve Bayes gets the highest detection rates for Probe attacks, as shown in Table 11, its precision is much lower compared with others. To fairly evaluate the capabilities of all methods in detecting network intrusion, we display the F1-scores obtained by all methods in Table 13. As seen in Table 13, our method exhibits its superior performance. Specifically, for R2L and U2R attacks, our method is far better than the others. Therefore, our method is very effective in the detection of network intrusion.

Finally, Table 14 shows a performance comparison of the proposed method with some recent hybrid methods using the NSL-KDD dataset in terms of accuracy and false alarm rate. The results in Table 14 are evaluated on the testing set of KDDTest^{+}. It can be seen from the table that the BC+k-NN method ranks third and second with respect to accuracy and false alarm rate, respectively. Although MOPF [32] ranks first based on false alarm rate, its accuracy is lower than that of BC+k-NN. For the metric of accuracy,

TABLE 14. Performance comparision of BC+k-NN with recent hybrid methods.

Methods	Accuracy	FAR
GHSOM+NSGA-II [24]	99.12%*	2.24%
SVC+KPCA [54]	93.40%	14.00%
OS-ELM+FST [31]	98.66%**	1.74%***
PSOM+PCA+FDR [27]	90.00%	NA
MOPF [32]	91.74%	1.44%*
SSL+NNR$_w$ [46]	84.12%	NA
BC+k-NN	94.92%***	1.59%**

* Ranked first.
** Ranked second.
*** Ranked third.

GHSOM+NSGA-II [24] and OS-ELM+FST [31] perform better than BC+k-NN, but their false alarm rates are higher than that of BC+k-NN. In addition, the proposed method only has one parameter. Conversely, the competing methods usually contain many parameters. For instance, the MOPF method [32] has four parameters. Thus, our method is competitive.

VI. CONCLUSION

This paper has presented an effective two-step hybrid intrusion detection approach based on binary classification and k-NN technique. In step 1, the proposed method uses several binary classifiers and one aggregation module to identify abnormal connections. By means of a binary classification technique, the proposed method reduces the bias that towards frequent attack types. In addition, the strategy used in the aggregation module improves the detection performance of the proposed method. In this study, we employ the C4.5 algorithm to train binary classifiers in consideration of its several advantages. For the connections whose classes are uncertain after step 1, the proposed method further classifies them in step 2 using the k-NN algorithm. Step 2 is a useful supplement to step 1. The combination of two steps makes the proposed method an effective intrusion detection technique.

Two experiments were conducted on the NSL-KDD dataset. The first experiment shows that step 1 not only correctly detects more abnormal connections than step 2 but also achieves better performance than step 2 for detecting abnormal connections in terms of accuracy, precision, detection rate, and F1-score. For individual attack types, step 1 correctly classifies more instances than step 2 for DoS and Probe attacks but correctly detects fewer instances than step 2 for R2L and U2R attacks. The reason lies in the imbalance of the NSL-KDD dataset. The results obtained from the second experiment demonstrate that the proposed method outperforms baselines (i.e., C4.5, Random Forests, k-NN, Backward Propagation Neural Network, and Naïve Bayes) with respect to various performance metrics in the detection of abnormal behaviors. For detection of the four attack types, the proposed method presents superior performance, especially for low-frequency attack types (i.e., R2L and U2R), in terms of F1-score.

REFERENCES

[1] Y. Chen, A. Abraham, and B. Yang, "Hybrid flexible neural-tree-based intrusion detection systems," *Int. J. Intell. Syst.*, vol. 22, no. 4, pp. 337–352, 2007.
[2] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.
[3] S.-Y. Wu and E. Yen, "Data mining-based intrusion detectors," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5605–5612, 2009.
[4] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, Mar. 2017.
[5] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Syst. Appl.*, vol. 29, no. 4, pp. 713–722, 2005.
[6] C. Guo, Y. Ping, N. Liu, and S.-S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, pp. 391–400, Nov. 2016.
[7] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1690–1700, 2014.
[8] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl.-Based Syst.*, vol. 78, pp. 13–21, Apr. 2015.
[9] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Appl. Soft Comput.*, vol. 18, pp. 178–184, May 2014.
[10] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016.
[11] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.
[12] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.
[13] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *J. Netw. Comput. Appl.*, vol. 66, pp. 1–16, May 2016.
[14] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur, "Bayesian event classification for intrusion detection," in *Proc. 19th Annu. Comput. Secur. Appl. Conf.*, Washington, DC, USA, Dec. 2003, pp. 14–23.
[15] L. Khan, M. Awad, and B. Thuraisingham, "A new intrusion detection system using support vector machines and hierarchical clustering," *VLDB J.*, vol. 16, no. 4, pp. 507–521, 2007.
[16] C. Xiang, P. C. Yong, and L. S. Meng, "Design of multiple-level hybrid classifier for intrusion detection system using Bayesian clustering and decision trees," *Pattern Recognit. Lett.*, vol. 29, no. 7, pp. 918–924, 2008.
[17] S. Peddabachigari, A. Abraham, C. Grosan, and J. Thomas, "Modeling intrusion detection system using hybrid intelligent systems," *J. Netw. Comput. Appl.*, vol. 30, no. 1, pp. 114–132, 2007.
[18] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Appl. Soft Comput.*, vol. 12, no. 10, pp. 3285–3290, 2012.
[19] D. T. Larose, *k-Nearest Neighbor Algorithm*. Hoboken, NJ, USA: Wiley, 2005, pp. 90–106.
[20] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
[21] 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. Defence Appl.*, Jul. 2009, pp. 1–6.
[22] G. Wang, J. Hao, J. Ma, and L. Huang, "A new approach to intrusion detection using artificial neural networks and fuzzy clustering," *Expert Syst. Appl.*, vol. 37, no. 9, pp. 6225–6232, 2010.
[23] A. S. Eesa, Z. Orman, and A. M. A. Brifcani, "A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems," *Expert Syst. Appl.*, vol. 42, no. 5, pp. 2670–2679, 2015.
[24] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez, "Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps," *Knowl.-Based Syst.*, vol. 71, pp. 322–338, Nov. 2014.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[26] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.

[27] E. De la Hoz, E. De La Hoz, A. Ortiz, J. Ortega, and B. Prieto, "PCA filtering and probabilistic SOM for network intrusion detection," *Neurocomputing*, vol. 164, pp. 71–81, Sep. 2015.

[28] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.

[29] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[30] D. Wang, D. S. Yeung, and E. C. C. Tsang, "Structured one-class classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1283–1295, Dec. 2006.

[31] R. Singh, H. Kumar, and R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Syst. Appl.*, vol. 42, no. 22, pp. 8609–8624, 2015.

[32] H. Bostani and M. Sheikhan, "Modification of supervised OPF-based intrusion detection systems using unsupervised learning and social network concept," *Pattern Recognit.*, vol. 62, pp. 56–72, Feb. 2017.

[33] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *Int. J. Imag. Syst. Technol.*, vol. 19, no. 2, pp. 120–131, 2009.

[34] A. Karami and M. Guerrero-Zapata, "A fuzzy anomaly detection system based on hybrid PSO-Kmeans algorithm in content-centric networks," *Neurocomputing*, vol. 149, pp. 1253–1269, Feb. 2015.

[35] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Experim. Technol.*, 2009, pp. 1–12.

[36] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier," *Expert Syst. Appl.*, vol. 39, no. 18, pp. 13492–13500, 2012.

[37] Y. Meng and L.-F. Kwok, "Enhancing false alarm reduction using voted ensemble selection in intrusion detection," *Int. J. Comput. Intell. Syst.*, vol. 6, no. 4, pp. 626–638, 2013.

[38] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern Recognit.*, vol. 43, no. 1, pp. 222–229, 2010.

[39] X.-Q. Zhang, C.-H. Gu, and J.-J. Lin, "Intrusion detection system based on feature selection and support vector machine," in *Proc. 1st Int. Conf. Commun. Netw. China*, Oct. 2006, pp. 1–5.

[40] A. Karami and M. Guerrero-Zapata, "A hybrid multiobjective RBF-PSO method for mitigating DoS attacks in named data networking," *Neurocomputing*, vol. 151, pp. 1262–1282, Mar. 2015.

[41] S.-Y. Ji, B.-K. Jeong, S. Choi, and D. H. Jeong, "A multi-level intrusion detection method for abnormal network behaviors," *J. Netw. Comput. Appl.*, vol. 62, pp. 9–17, Feb. 2016.

[42] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization," *Neurocomputing*, vol. 199, pp. 90–102, Jul. 2016.

[43] Y. Canbay and S. Sagiroglu, "A hybrid method for intrusion detection," in *Proc. IEEE Int. Conf. Mach. Learn. Appl.*, Dec. 2016, pp. 156–161.

[44] T. Dash, "A study on intrusion detection using neural networks trained with evolutionary algorithms," *Soft Comput.*, vol. 21, no. 10, pp. 2687–2700, May 2017.

[45] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system," *Expert Syst. Appl.*, vol. 67, pp. 296–303, Jan. 2017.

[46] 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.

[47] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[48] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Boca Raton, FL, USA: CRC Press, 1984.

[49] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, 2008.

[50] J. H. Zar, *Spearman Rank Correlation*. Indianapolis, IN, USA: Wiley, 2005.

[51] 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.

[52] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1184–1199, 2011.

[53] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 303–336, 1st Quart., 2014.

[54] E. De la Hoz, A. Ortiz, J. Ortega, and E. De la Hoz, *Network Anomaly Classification by Support Vector Classifiers Ensemble and Non-Linear Projection Techniques*. Berlin, Germany: Springer, 2013, pp. 103–111.

LONGJIE LI received the B.S. and Ph.D. degrees in computer science from Lanzhou University, Lanzhou, China, in 2006 and 2014, respectively. He is currently an Assistant Professor with the School of Information Science and Engineering, Lanzhou University. His research interests include data mining, machine learning, and complex network analysis. He is a member of China Computer Federation.

YANG YU was born in 1991. She received the master's degree in computer science from Lanzhou University, Lanzhou, China, in 2017. She is currently an Engineer with the China Railway Signal and Communication Co., Ltd.

SHENSHEN BAI was born in 1983. She is currently pursuing the Ph.D. degree in computer science, Lanzhou University. Her research interests include data mining and complex network analysis.

YING HOU was born in 1993. She is currently pursuing the master's degree in computer science at Lanzhou University, Lanzhou, China. Her research interests include data mining and intrusion detection.

XIAOYUN CHEN is currently a Professor with the School of Information Science and Engineering, Lanzhou University. Her research interests include data mining, big data analysis, and complex network analysis. She is a senior member of the China Computer Federation.