EAI Endorsed Transactions
on Context-aware Systems and Applications Research Article EAI.EU

A hybrid classification model in improving the classification quality of network intrusion detection systems

Hoang Ngoc Thanh

The Saigon International University, Vietnam

Abstract

Stream-based anomaly detection is an issue that continues to be researched in the cybersecurity environment. Much previous research has applied machine learning as a method to improve anomaly detection in network intrusion detection systems. Recent research shows that network intrusion detection systems still face challenges in improving accuracy, reducing false alarm rates, and detecting new attacks.

The article proposes a hybrid classification model that combines improved data preprocessing techniques with ensemble techniques. Experimental results on the UNSW-NB15 dataset show that the proposed solutions have helped improve the classification quality of network intrusion detection systems compared to some other research.

Received on 23 July 2024; accepted on 25 April 2025; published on 17 June 2025

Keywords: Machine Learning, NIDS, Ensemble, Feature Selection, Resampling, UNSW-NB15

doi:10.4108/eetcasa.6735

1. Introduction

To deal with cyberattacks, the key issue for network administrators is to detect and prevent intrusions quickly. One of the systems used by network administrators today is the intrusion detection system (Intrusion Detection System: IDS).

IDS is a system that monitors network or server traffic to detect unusual phenomena and unauthorized activities entering the network or server system. For IDS, there are three methods to detect attacks: (1) signature-based detection; (2) anomaly-based detection; and (3) detection based on a combination of the above two methods.

Signature-based detection is designed to detect known attacks using the signatures of those attacks. This is an effective method to detect known attacks stored in the IDS database.

However, with new or variant attacks, IDS cannot detect them because the signature of the attack is not stored. To fix this problem, anomaly-based detection compares the user's current activities with predefined profiles to detect intrusions. Anomaly-based detection is effective against unknown attacks or zero-day attacks without any updates to the system. However, this method often has a high false positive rate (FPR) [1].

With anomaly-based intrusion detection, much research has proposed using machine learning (ML) to reduce the FPR and increase detection accuracy. However, to deal with big data, traditional ML techniques require a lot of time for training. Using big data and ML techniques for IDS can solve many challenges, such as computing speed and time, as well as developing accurate IDS [1, 2]. Currently, IDS based on machine learning and deep learning are being deployed as a potential solution to effectively detect cyberattacks. Khraisat et al. [3] detailed a survey of recent research on the methodology, types, and technologies of IDS based on ML, with advantages and limitations. Such approaches still have challenges in generating and updating information about new attacks, as well as reducing false alarm rates and increasing detection accuracy. The content of the article proposes a hybrid classification model, including the

*Corresponding author. Email: hoangngocthanh@siu.edu.vu

use of feature selection, resampling, and ensemble techniques, to improve the classification quality of IDS (QoC) with limited computational resources.

2. Related works

2.1. Feature Selection

Feature selection is a method to eliminate unnecessary or noisy features and select the most relevant subset of features to better classify samples of various attack types. This needs to be done because:

(1) A single selection strategy is not enough to achieve consistency across multiple datasets because network traffic activity is constantly changing [4].

(2) A subset of features for each attack type must be determined, since a common subset is not sufficient to properly represent all the different attacks [4].

(3) feature selection can significantly improve not only detection accuracy but also computational efficiency, where: irrelevant or redundant features can lead to poor detection rates or overfitting [5–7]; more features will cause higher computational cost and complexity [4, 8].

(4) Finally, there are some known types of attacks that have become challenging to identify because they are so complex and can be mislabeled as normal data. Research and experiments have shown that feature selection can solve this problem by defining a subset of features adapted to the behavior of each type of attack [6–8].

2.2. Data imbalance between classes

The problem of imbalanced data is one of the important issues and has received the attention of many researchers. A dataset is said to be imbalanced when the number of samples belonging to one class label is much smaller than other class labels. To solve the problem, resampling techniques have been proposed. Two main approaches are used: (1) remove some samples from the majority class, called undersampling, and (2) duplicate some of the samples from the minority class, called oversampling. Both oversampling and undersampling aim to change the ratio between the majority and the minority classes [9]. In this way, resampling allows different classes to have a relatively similar influence on the results of the classification model. Research shows that resampling the training dataset improves the accuracy of NIDS [10, 11]. Leevy et al. [12] emphasized its importance and presented a survey of recent research on the problem of class imbalance.

Oversampling (OS). One of the commonly used oversampling techniques is SMOTE (Synthetic Minority Over-Sampling Technique) [13]. The implementation of SMOTE is described as follows: Take a sample $\vec{a}$ from the minority class of the dataset and randomly select one sample $\vec{b}$ from among the k nearest neighbors of the same class $\vec{a}$ (in the feature space). A new synthetic data sample $\vec{x} = \vec{a} + w(\vec{b} - \vec{a})$ is created and added to the dataset, where w is the random weight in the interval $[0, 1]$.

Based on SMOTE, several various techniques have been built and developed. The first is the Cluster SMOTE technique [12]. In this technique, the training dataset is first divided into k clusters using the K-means algorithm, in each cluster the imbalance ratio is calculated:

$$\text{IR} = \frac{\text{Number of minority class samples in the cluster}}{\text{Number of majority class samples in the cluster}}$$

Then use SMOTE to increase the number of minority samples in clusters with IR>1.

Next is the Adaptive Synthetic Sampling technique (ADASYN), which is built by shifting the importance of classification boundaries to difficult minority classes. ADASYN uses a weighted distribution for minority class samples that vary according to training difficulty, where more synthetic data is generated for more difficult minority class samples to learn [14].

Another SMOTE-based innovation is Borderline-SMOTE, Borderline-SMOTE there are two variants, Borderline-SMOTE1 and Borderline-SMOTE2. This method oversamples of minority samples only near the boundary and nearest neighbors of the same type. The difference between the two variants is that Borderline-SMOTE2 uses both positive and negative nearest neighbor. Compared to conventional SMOTE, Borderline-SMOTE does not create synthetic samples for noise, but focuses its efforts near the boundary, thereby helping the decision function to create better boundaries between classes. In terms of performance, Borderline-SMOTE has also been reported to perform better than SMOTE [15].

Undersampling (US). The first known undersampling technique is Tomek Link which is defined as follows: The Tomek association is defined as follows: provides a pair of samples (x_i, x_j), where x_i belongs to the minority class, x_j belongs to the majority class, and $d(x_i, x_j)$ is the distance between x_i and x_j. The pair (x_i, x_j) is called a Tomek Link if there is no x_k that satisfies $d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$. In this way, if two samples form a Tomek Link, then either one of these samples is noisy or both are near the boundary. So one can use Tomek Links to clean up the overlap between classes. By removing overlapping samples, one can establish well-defined clusters in the training dataset, which leads to improved classification quality [16].

Another approach is the Neighborhood Cleaning Rule (NCR) can be described as follows: for each sample

E_i in the training dataset, its three nearest neighbors are found. If E_i belongs to the majority class and the class given by the three nearest neighbors is different from the E_i class, then E_i will be deleted. If E_i belongs to the minority class and the class given by the three nearest neighbors is different from the E_i class, then the nearest neighbors belonging to the majority class are eliminated [9].

The motivation behind the Edited Nearest Neighbors (ENN) algorithm is similar to the Tomek Link. ENN tends to remove more samples than Tomek Link, so it will provide more in-depth data cleaning. Unlike NCR, which is an undersampling method, ENN is used to remove samples from both classes. Therefore, any sample misclassified by its three nearest neighbors will be removed from the training dataset [17].

2.3. Ensemble Techniques

Homogeneous Ensemble. Folino et al. [18] proposed a method for distributed intrusion detection that uses genetic programming to create classifiers based on decision trees. These classifiers are then assembled into an ensemble using AdaBoost.M2, a variant of AdaBoost. The KDDCup99 dataset is used to evaluate the proposed system. Experimental results demonstrate the ability of genetic programming to successfully handle the problem of intrusion detection on distributed data. Gudadhe et al. [19] used Boosting to assemble a family of decision trees into an ensemble. They presented experimental research in which the method they developed was compared with Naïve Bayes, KNN, eClass0, and eClass1. The report shows that their method outperforms other methods on the KDDCup99 dataset. An advantage over previous research is that this implementation is capable of detecting all types of attacks. Syarif et al. [20] implemented Bagging, Boosting, and Stacking methods to solve the intrusion detection problem. The main goal of their research was to improve the classification accuracy and reduce the FPR for classifying the NSL-KDD dataset. Bagging and Boosting techniques are built with four traditional classification algorithms: Naïve Bayes, decision trees, JRip, and KNN. Additionally, heterogeneous ensembles are constructed using a stacking strategy, where four algorithms are used one after another to perform meta-level classification. Their approach achieves over 99% accuracy in detecting known intrusions. However, for new types of intrusions, the accuracy is only 60%. Using a homogeneous ensemble generated with Bagging and Boosting showed no significant increase in accuracy. On the other hand, heterogeneous ensembles established with stacking lead to a significant reduction (46.84%) in the FPR.

Heterogeneous Ensemble. An original contribution was presented by Govindarajan and Chandrasekaran [21], who proposed a hybrid ensemble based on the decisions of different classes. They implemented a generalized version of the Bagging and Boosting algorithm. Adaptive Resampling and Combining, also known as Arcing, to generate different training datasets for two classifiers uses a base function (RBF) neural network and a support vector machine (SVM). Additionally, the authors implemented the best first search (BFS) for feature selection. The final decision was reached by a majority vote. Experiments performed on the NSL-KDD dataset demonstrated that a hybrid method is more effective than a single classifier. The reported classification accuracy of the RBF-SVM is 85.17%. Haq et al. [22] developed the IDS in three phases: (1) a hybrid approach to feature selection; (2) classification with base classifiers; and (3) implementing a voting strategy of majority to form the final decision. The feature selection process is based on three methods: BFS, genetic search (GS), and ranking search (RS). The final set of features is obtained by combining the results from all three feature selection algorithms, where the most commonly selected features by all three algorithms are selected for the final set. The classifications in the second stage are performed using three classification algorithms: Naïve Bayes, Bayesian networks, and decision trees. The experimental procedure is performed on the NSL-KDD dataset. Although the proposed method showed improved computational efficiency, it classified the data with less accuracy when compared to a majority voting scheme based only on the features selected by RS. Tama and Rhee [23] performed binary (normal vs. attack) classification of the NSL-KDD dataset, with results based on majority voting and averaging of posterior probabilities. Additionally, they developed two hybrid feature selection methods, one based on the swarm optimization method and one based on correlation, to preprocess the training and testing datasets. The base classifiers in the combination include the C4.5 decision tree, random forest (RF), and CART. Experimental results indicate that the best performance is achieved by a combination based on averaging the posterior probabilities. It is worth noting that they also obtained similar results when using Boosting with the C4.5 base classifier.

3. Proposed Hybrid Classification Model

3.1. Proposed Feature Selection Algorithm (pFSA)

To select features, the article uses two algorithms, pBFE and pFFC [24], proposed by the author. The pBFE and pFFC algorithms were proposed on the basis of solving the limitations of the Backward Feature Elimination algorithm (BFE) [25] and the Forward Feature Construction algorithm (FFC) [25].

3.2. Proposed Dataset Resampling

Proposed Oversampling Algorithm (pOSA). The oversampling techniques all rely on k nearest neighbors to create synthetic data samples with the participation of all features. The problem is that there are unimportant or noisy features when calculating distances to determine the k nearest neighbors, which can affect the quality of algorithms. To eliminate these unimportant or noisy features, the author [24] proposes to use two solutions presented in Algorithm 1 and Algorithm 2. The time complexity of the algorithm is $O(N \times (N-1)/2)$, where N is the number of features in the dataset.

Algorithm 1 Oversampling combined with pFFC

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using pOSA

begin
  Initialize:
  Calculate IoF of all features on the dataset D
    ▷ IoF: Important of the Features
  $S_{max} \leftarrow$ all features sorted in descending of IoF
  $S_{min} \leftarrow$ the features obtained from pFSA
  $S_{opt} \leftarrow S_{min}$
  $S_{add} = S_{max} \setminus S_{min}$
  OS on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{add}) **do**
    $s \leftarrow S_{add}\ [i\text{ -}1]$
    $S_1 = S_{opt} \cup \{s\}$
    OS on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after OS **then**
      $Best \leftarrow$ the QoC of S_1 after OS
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates with s **then**
          $S_2 = S_{opt} \cup \{s\} \setminus \{s_{ca}\}$
          if QoC of S_2 after OS > $Best$ **then**
            $S_1 \leftarrow S_2$
            $Best \leftarrow$ the QoC of S_2 after OS
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

Proposed Undersampling Algorithm (pUSA). The undersampling techniques also rely on k nearest neighbors to remove unwanted overlap between classes with the participation of all features. The problem is that there are unimportant features, or even noise when calculating distances to determine the k nearest neighbors, which can affect the quality of undersampling algorithms.

Algorithm 2 Oversampling combined with pBFE

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using pOSA

begin
  Initialize:
  Calculate IoF of all features on the dataset D
    ▷ IoF: Important of the Features
  $S_{max} \leftarrow$ all features sorted in ascending of IoF
  $S_{min} \leftarrow$ the features obtained from pFSA
  $S_{opt} \leftarrow S_{max}$
  $S_{del} = S_{max} \setminus S_{min}$
  OS on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{del}) **do**
    $s \leftarrow S_{del}\ [i\text{ -}1]$
    $S_1 = S_{opt} \setminus \{s\}$
    OS on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after OS **then**
      $Best \leftarrow$ the QoC of S_1 after OS
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates and has IoF < s **then**
          $S_2 = S_{opt} \setminus \{s_{ca}\}$
          if QoC of S_2 after OS > $Best$ **then**
            $S_1 \leftarrow S_2$
            $Best \leftarrow$ the QoC of S_2 after OS
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

To eliminate these unimportant or noisy features, the author [24] proposes to use two solutions presented in Algorithm 3 and Algorithm 4. The time complexity of the algorithm is $O(N \times (N-1)/2)$, where N is the number of features in the dataset.

3.3. Proposed Ensemble Techniques

Algorithm 5 and Algorithm 6 in detail the selection of ensemble classifiers using homogeneous and heterogeneous techniques. Accordingly, the training dataset D is divided into k subsets of the same size (k-fold). In the first loop, the first subset is used as the testing dataset, and $(k-1)$ the remaining subsets are used as the training dataset. These training and testing datasets are used to train and test ensemble classifiers using homogeneous and heterogeneous matching techniques. In the next loop, the second subset is used as the testing dataset, and $(k-1)$ the remaining subsets are used as the training dataset. These training and testing datasets

Algorithm 3 Undersampling combined with pFFC

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using pUSA

begin
  Initialize:
  Calculate IoF of all features on the dataset D
    ▷ IoF: Important of the Features
  $S_{max} \leftarrow$ all features sorted in descending of IoF
  $S_{min} \leftarrow$ the features obtained from pFSA
  $S_{opt} \leftarrow S_{min}$
  $S_{add} = S_{max} \setminus S_{min}$
  US on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{add}) **do**
    $s \leftarrow S_{add}[i\text{-}1]$
    $S_1 = S_{opt} \cup \{s\}$
    US on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after US **then**
      $Best \leftarrow$ the QoC of S_1 after US
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates with s **then**
          $S_2 = S_{opt} \cup \{s\} \setminus \{s_{ca}\}$
          if QoC of S_2 after US > $Best$ **then**
            $S_1 \leftarrow S_2$
            $Best \leftarrow$ the QoC of S_2 after US
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

Algorithm 4 Undersampling combined with pBFE

Input
D - Training Dataset
Output
S_{opt} - The optimal subset of features using pUSA

begin
  Initialize:
  Calculate IoF of all features on the dataset D
    ▷ IoF: Important of the Features
  $S_{max} \leftarrow$ all features sorted in ascending of IoF
  $S_{min} \leftarrow$ the features obtained from pFSA
  $S_{opt} \leftarrow S_{max}$
  $S_{del} = S_{max} \setminus S_{min}$
  US on training dataset D using features $\in S_{opt}$
  for $i \leftarrow 1$ to len(S_{del}) **do**
    $s \leftarrow S_{del}[i\text{-}1]$
    $S_1 = S_{opt} \setminus \{s\}$
    US on the dataset D using the features $\in S_1$
    if QoC of S_1 is better than S_{opt} after US **then**
      $Best \leftarrow$ the QoC of S_1 after US
      for each s_{ca} in S_{opt} **do**
        if s_{ca} correlates and has IoF < s **then**
          $S_2 = S_{opt} \setminus \{s_{ca}\}$
          if QoC of S_2 after US > $Best$ **then**
            $S_1 \leftarrow S_2$
            $Best \leftarrow$ the QoC of S_2 after US
          end if
        end if
      end for
      $S_{opt} \leftarrow S_1$
    end if
  end for
  return S_{opt}
end

are used to train and test the ensemble classifiers, as done in the first iteration. This process is repeated k times. The classification results of the ensemble classifiers are presented as the average value of the evaluation indexes after k iterations, used to compare and select the best ensemble classifier when classifying attacks on the UNSW-NB15 dataset.

4. Experimental results

The dataset used in the experiments is UNSW-NB15, which is a dataset created by the Australian Cyber Security Center, including nine types of attacks: Worms, Shellcode, Backdoor, Analysis, Reconnaissance, DoS, Fuzzers, Exploits, and Generic [26]. The UNSW-NB15 dataset has several advantages when compared with other datasets: (1) it contains synthetic attack activities; (2) the probability distributions of the training and testing datasets are similar; (3) it includes a set of features from the payload and header of packets to effectively reflect network packets; and (4) the dataset has many complex patterns.

Feature selection techniques used: $pBFE$ and $pFFC$. Resampling techniques used: mOS and mUS. Homogenous techniques used: Bagging, AdaBoost, Stacking, Decorate, Voting, and RF. Heterogeneous techniques were used: Voting and Stacking.

The proposed hybrid classification model is presented in Figure 1. Table 1 shows the proposed feature selection, resampling, and ensemble techniques used with each type of attack.

The achieved evaluation indexes include True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN), Accuracy (Accu), Sensitivity (Sens), Specificity (Spec), FPR, and False Negative Rate (FNR) shown in the Table 2 with nine attack types in the UNSW-NB15 dataset.

The above classification results are also used to compare with some recent research on IDS using the

Algorithm 5 Building classifiers using homogenous ensemble techniques

Input
D - Dataset
k - k-fold cross-validation
n - Number of base classifiers in the ensemble
M - Set of ML techniques
E - Set of homogeneous ensemble techniques
Output
Best - Best homogeneous ensemble classifier

begin
  Initialize:
  $bf = 0$
  Divide set D into k subsets of equal size D_k
  for each $e \in E$ **do**
    for each $m \in M$ **do**
      $f = 0$
      for $i \leftarrow 1$ to k **do**
        $T = D \setminus D_i$
        Use T to train e with ML technique m
        Use D_i to test the ensemble e
        f_i = The classification quality of e
        $f = f + f_i$
      end for
      if $f > bf$ **then**
        $bf = f$
        $Best = e$
      end if
    end for
  end for
  return *Best*
end

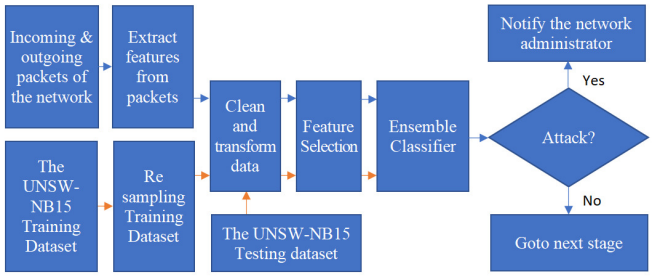

Figure 1. The proposed hybrid model to detect network attack types

UNSW-NB15 dataset. Table 3 compares the accuracy index with the results achieved by articles [27], [28], and [29]. Table 4 compares the sensitivity index with the results achieved by articles [27] and [30]. The cells highlighted are the ones that give the best results. Accordingly, the proposed hybrid classifier achieves better accuracy than other articles in Reconnaissance (Recce), Exploits, and Generic. Regarding sensitivity,

Algorithm 6 Building classifiers using heterogeneous ensemble techniques

Input
D - Dataset
k - k-fold cross-validation
n - Number of base classifiers in the ensemble
M - Set of ML techniques
E - Set of heterogeneous ensemble techniques
Output
Best - Best heterogeneous ensemble classifier

begin
  Initialize:
  $bf = 0$
  Divide set D into k subsets of equal size D_k
  for each $e \in E$ **do**
    $f = 0$
    for $i \leftarrow 1$ to k **do**
      $T = D \setminus D_i$
      Use T to train e with base classifiers
      Use D_i to test the ensemble e
      f_i = The classification quality of e
      $f = f + f_i$
    end for
    if $f > bf$ **then**
      $bf = f$
      $Best = e$
    end if
  end for
  return *Best*
end

Table 1. Proposed techniques for each type of attack

Attacks	pFSA	pOSA	pUSA	Ensemble
Worms	pBFE	BL-SMOTE1+pBFE	ENN+pFFC	AdaBoost
Shellcode	pBFE	BL-SMOTE1+pFFC	NCR+pBFE	Mix Stacking
Backdoor	pBFE	ADASYN+pBFE	NCR+pFFC	AdaBoost
Analysis	pBFE	ADASYN+pFFC	ENT+pBFE	Bagging
Recce	pBFE	CL-SMOTE+pBFE	ENN+pBFE	AdaBoost
DoS	pBFE	CL-SMOTE+pBFE	ENT+pFFC	Mix Stacking
Fuzzers	pBFE	CL-SMOTE+pFFC	NCR+pFFC	AdaBoost
Exploits	pBFE	ADASYN	ENT+pFFC	Bagging
Generic	pBFE	BL-SMOTE2+pFFC	ENT+pFFC	AdaBoost

Table 2. Evaluation indexes of the proposed hybrid model

Attacks	TP	FP	TN	FN	Accu	Sens	Spec	FPR	FNR
Worms	33	208	82080	11	.9973	.7500	.9975	.0025	.2500
Shellcode	365	1367	80587	13	.9832	.9656	.9833	.0167	.0344
Backdoor	413	2366	79383	170	.9692	.7084	.9711	.0289	.2916
Analysis	336	3313	78342	341	.9556	.4963	.9594	.0406	.5037
Recce	2989	299	78537	507	.9902	.8550	.9962	.0038	.1450
DoS	1124	83	78160	2965	.9630	.2749	.9989	.0011	.7251
Fuzzers	4394	966	75304	1668	.9680	.7248	.9873	.0127	.2752
Exploits	8529	683	70517	2603	.9601	.7662	.9904	.0096	.2338
Generic	18143	0	63461	728	.9912	.9614	1,0000	.0000	.0386
Normal	35940	781	44551	1060	.9776	.9714	.9828	.0172	.0286

Table 3. Compare the accuracy index with some recent research

Attacks	Proposed hybrid model	[27]	[28]	[29]
Worms	0.9973	0.9978	0.9992	0.9728
Shellcode	0.9832	0.9833	0.9940	0.9992
Backdoor	0.9692	0.9793	0.9911	0.9906
Analysis	0.9556	0.9930	0.9926	0.9944
Recce	0.9902	0.9618	0.9533	0.9874
DoS	0.9630	0.9571	0.9490	0.9814
Fuzzers	0.9680	0.9504	0.9147	0.9892
Exploits	0.9601	0.9358	0.9012	0.9391
Generic	0.9912	0.9870	0.9823	0.9834
Normal	0.9776	0.9459	0.9354	0.9816

Table 4. Compare the sensitivity index with some recent research

Attacks	Proposed hybrid model	[27]	[30]
Worms	0.7500	0.1837	-
Shellcode	0.9656	0.3639	-
Backdoor	0.7084	0.6732	-
Analysis	0.4963	0.2045	-
Recce	0.8550	0.4604	0.7170
DoS	0.2749	0.1429	0.0500
Fuzzers	0.7248	0.6442	-
Exploits	0.7662	0.7622	0.5464
Generic	0.9614	0.8137	0.9672
Normal	0.9714	0.9739	0.9800

the proposed hybrid classifier is better than other articles in most of them except Generic and Normal.

5. CONCLUSION

Experimental results show that a hybrid classification model, including the use of feature selection, resampling, and ensemble techniques, can improve the classification quality of IDS under limited computational resources. However, the achieved results also have limitations and directions for further research:

(1) The training time of the proposed hybrid classification model is still large, and the model has not been tested with new attack types in real-time.

(2) Improving processing efficiency through the parallel processing approach, as well as optimizing the parameters of machine learning techniques, is an open issue.

(3) It is necessary to research solutions based on modern machine learning models such as fast learning, and deep learning, ... for the problem of detecting network intrusions.

References

[1] SM Othman, FM Ba-Alwi, NT Alsohybe and AY Al-Hashida, "Intrusion detection model using machine learning algorithm on Big Data environment," J Big Data, vol. 5, no. 34 https://doi.org/10.1186/s40537-018-0145-4, 2018.

[2] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," Artificial Intelligence Review, vol. 55, p. 453–563, 2022.

[3] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," Cybersecurity, vol. 2, no. 1, pp. 1-22, 2019.

[4] Z. Liu, R. Wang, M. Tao and X. Cai, "A class-oriented feature selection approach for multi-class imbalanced network traffic datasets based on local and global metrics fusion," Neurocomputing, vol. 168, pp. 365-381, 2015.

[5] HI Alsaadi, RM Almuttairi, O. Bayat and a. ON Ucani, "Computational intelligence algorithms to handle dimensionality reduction for enhancing intrusion detection system," J. Inf. Sci. Eng., vol. 36, no. 2, pp. 293-308, 2020.

[6] O. Almomani, "A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms," Symmetry (Basel), vol. 12, no. 6, pp. 1-20, 2020.

[7] MS Bonab, A. Ghaffari, FS Gharehchopogh and P. Alemi, "A wrapper-based feature selection for improving performance of intrusion detection systems," Int. J. Commun. Syst., vol. 33, no. 12, pp. 1-25, 2020.

[8] Y. Zhu, J. Liang, J. Chen and Z. Ming, "An improved NSGA-III algorithm for feature selection used in intrusion detection," Knowledge-Based Systems, vol. 116, pp. 74-85, 2017.

[9] N. Junsomboon, "Combining Over-Sampling and Under-Sampling Techniques for Imbalance Dataset," in Proceedings of the 9th International Conference on Machine Learning and Computing, 2017.

[10] S. Bagui and K. Li, "Resampling imbalanced data for network intrusion detection datasets," Journal of Big Data, vol. 8, no. 6, 2021.

[11] H. Ahmed, A. Hameed and N. Bawany, "Network intrusion detection technique using oversampling and machine learning algorithms," PeerJ Computer Science 8:e820 DOI 10.7717/peerj-cs.820, 2022.

[12] J. Leevy, T. Khoshgoftaar, R. Bauder and N. Seliya, "A survey on addressing high-class imbalance in big data," Journal of Big Data, vol. 5, no. 1, 2018.

[13] NV Chawla, KW Bowyer, LO Hall and WP Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, p. 321–357, 2002.

[14] Y. Pristyanto, AF Nugraha, A. Dahlan, LA Wirasakti, AA Zein and I. Pratama, "Multiclass Imbalanced Handling using ADASYN Oversampling and Stacking Algorithm," 2022," in 2022 16th International Conference on Ubiquitous Information Management and Communication, DOI: 10.1109/IMCOM53663.2022.9721632, 2022.

[15] A. Pathak, "Analysis of Different SMOTE Based Algorithms on Imbalanced Datasets," International Research Journal of Engineering and Technology (IRJET), vol. 8, no. 8, pp. 4111-4114, 2021.

[16] T. Elhassan, M. Aljurf, F. Al-Mohanna and M. Shoukri, "Classification of Imbalance Data using Tomek Link (T-Link) Combined with Random Under-Sampling (RUS) as a Data Reduction Method," Journal of Informatics and Data Mining, vol. 1, 2016.
[17] D. Guan, W. Yuan, Y.-K. Lee and S. Lee, "Nearest neighbor editing aided by unlabeled data," Information Sciences, vol. 179, pp. 2273-2282, 2009.
[18] G. Folino, C. Pizzuti and G. Spezzano, "An ensemble-based evolutionary framework for coping with distributed intrusion detection," Genetic Programming and Evolvable Machines, vol. 11, pp. 131-146, 2010.
[19] M. Gudadhe, P. Prasad and K. Wankhade, "A new data mining based network intrusion detection model," in Computer and Communication Technology (ICCCT), International Conference on, IEEE, 2010.
[20] I. Syarif, E. Zaluska, A. Prugel-Bennett and G. Wills, "Application of Bagging, Boosting and Stacking to Intrusion Detection," in International Workshop on Machine Learning and Data Mining in Pattern Recognition, 2012.
[21] M. Govindarajan and R. Chandrasekaran, "Intrusion detection using an ensemble of classification methods," in World Congress on Engineering and Computer Science, 2012.
[22] NF Haq, AR Onik and FM Shah, "An ensemble framework of anomaly detection using hybridized feature selection approach (HFSA)," in SAI Intelligent Systems Conference (IntelliSys), 2015.
[23] BA Tama and KH Rhee, "A combination of PSO-based feature selection and tree-based classifiers ensemble for intrusion detection systems," Advances in Computer Science and Ubiquitous Computing, Springer, pp. 489-495, 2015.
[24] HN Thanh, "The data preprocessing in improving the classification quality of network intrusion detection systems," EAI Endorsed Transactions on Context-aware Systems and Applications, vol. 9 (2023), pp. 1-14, 2023.
[25] S. Rosaria, I. Adae, H. Aaron and B. Michael, Seven Techniques for Dimensionality Reduction, Zurich Switzerland: KNIME, 2014.
[26] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive dataset for network intrusion detection systems," in Conference on Military Communications and Information Systems, 2015.
[27] D. Papamartzivanos, FG Mármol and G. Kambourakis, "Dendron: Genetic trees driven rule induction for network intrusion detection systems," Future Generation Computer Systems, vol. 79, pp. 558-574, 2018.
[28] J. Sharma, C. Giri, O.-C. Granmo and M. Goodwin, "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation," EURASIP Journal on Information Security, vol. 2019, pp. 1-15, 2019.
[29] S. Moualla, K. Khorzom and A. Jafar, "Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset," Computational Intelligence and Neuroscience, vol. 2021, pp. 1-13, 2021.
[30] V. Kumar, D. Sinha, AK Das, SC Pandey and RT Goswami, "An integrated rule based intrusion detection system: analysis on UNSW-NB15 dataset and the real time online dataset," Cluster Computing, vol. 23, p. 1397–1418, 2019.