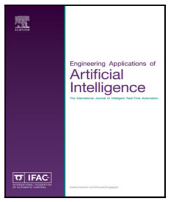
Engineering Applications of Artificial Intelligence 94 (2020) 103770



Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Boosting algorithms for network intrusion detection: A comparative evaluation of Real AdaBoost, Gentle AdaBoost and Modest AdaBoost



Amin Shahraki [a,c,*], Mahmoud Abbasi [b], Øystein Haugen [c]

[a] Department of Informatics, University of Oslo, Oslo, Norway
[b] Faculty of Engineering, Islamic Azad University of Mashhad, Mashhad, Iran
[c] Faculty of Computer Sciences, Østfold University College, Halden, Norway

ARTICLE INFO

Keywords:
Malware detection
Intrusion detection systems
Anomaly detection
Machine learning
Ensemble learning
Boosting techniques
AdaBoost

ABSTRACT

Computer networks have been experienced ever-increasing growth since they play a critical role in different aspects of human life. Regarding the vulnerabilities of computer networks, they should be monitored regularly to detect intrusions and attacks by using high-performance Intrusion Detection Systems (IDSs). IDSs try to differentiate between normal and abnormal behaviors to recognize intrusions. Due to the complex behavior of malicious entities, it is crucially important to adopt machine learning methods for intrusion detection with a fine performance and low time complexity. Boosting approach is considered as a way to deal with this challenge. In this paper, we prepare a clear summary of the latest progress in the context of intrusion detection methods, present a technical background on boosting, and demonstrate the ability of the three well-known boosting algorithms (Real Adaboost, Gentle Adaboost, and Modest Adaboost) as IDSs by using five IDS public benchmark datasets. The results show that the Modest AdaBoost has a higher error rate compared to Gentle and Real AdaBoost in IDSs. Besides, in the case of IDSs, Gentle and Real AdaBoost show the same performance as they have about 70% lower error rates compared to Modest Adaboost, however, Modest AdaBoost is about 7% faster than them. In addition, as IDSs need to retrain the model frequently, the results show that Modest AdaBoost has a much lower performance than Gentle and Real AdaBoost in case of error rate stability.

1. Introduction

In recent years, the Internet and relatively its infrastructure are integral parts of the daily life of human society. The Internet plays an active role in connecting billions of machines in different applications e.g. healthcare, industry, and business. It provides a global network infrastructure to connect a massive number of virtual and physical things like communication devices, Cyber Physical Systems (CPSs), and social networks (Al-Fuqaha et al., 2015). The trend of connecting the world leads to an exponential growth in the amount of human and machine-generated network traffic that passes the network infrastructure to reach destinations. According to the Mobile and Wireless Communications Enablers for the Twenty-Twenty Information Society (METIS) (METIS, 2020), it is expected that the volume of mobile and wireless network traffic will increase by more than a thousand-fold in the next decade. Moreover, there are concrete reasons that we will have 50 billion connected devices by 2025 in the world (Beyene et al., 2017). Cisco reported that around 71% of total IP traffic by 2022 will be related to wireless and mobile devices (Conti et al., 2018) that oblige service providers to establish concrete and impenetrable network infrastructure for protecting users.

Due to the acceleration of Internet of Things (IoT) applications growth and heterogeneous-vulnerable connected devices, the last years have been the most difficult period from security perspectives. Although different techniques are emerging to detect and neutralize attacks, still there are severe concerns regarding the security and privacy of users. IBM X-Force Threat reports show that the client organizations monitored by this company have been experienced a lower percentage of attacks in 2016 than 2015 (Tounsi and Rais, 2018); however, the reduction in cyber-attacks could mean malicious users rely increasingly on proven attacks, hence malicious entities make fewer attempts for an intrusion. Cyber attacks can hurt real-world infrastructures that can disrupt vital services. In 2015, a malware, (called also malicious software), targeted the electrical power network supply in Ukraine and caused a power outage for a long time in the middle of the cold season (Liang et al., 2016). Furthermore, reports indicate a huge number of data breaches throughout the world, e.g. in Panama, Iceland, and the U.S (Graves et al., 2018). Attackers release privileged

* Corresponding author at: Department of Informatics, University of Oslo, Oslo, Norway.
E-mail address: am.shahraki@ieee.org (A. Shahraki).

https://doi.org/10.1016/j.engappai.2020.103770
Received 25 February 2020; Received in revised form 1 June 2020; Accepted 16 June 2020
Available online 1 July 2020
0952-1976/© 2020 Published by Elsevier Ltd.

or sensitive data on the Internet or any other unsafe environments which lead to serious economic loss.

Intensive studies have been carried out by academics and private companies to detect, prevent, and inactivate cyber-attacks. Based on fundamental techniques like Artificial intelligence (AI) and statistical methods, different techniques have been proposed to improve the efficiency of IDS (Kolias et al., 2011). The literature introduces IDS as an essential element of every network security system. Machine learning algorithms as a type of AI technique have attracted lots of attention in the literature as an early detection or prevention network attacks methods (Sreeram and Vuppala, 2019). This scheme considerably eliminates the need for using humans as experts to detect the abnormal patterns manually. In addition, motivated by the recent advances and impressive performance in different fields such as mobile and wireless networks (Zhang et al., 2019), big data (Mohammadi et al., 2018), and computer vision (Akhtar and Mian, 2018), the feasibility of Deep Learning (DL) as a branch of machine learning methods for IDS purposes have been investigated by researchers (Wang et al., 2018). Here, one may categorize the main applications of machine learning in cybersecurity into five main classes:

1. Misuse/Signature detection
2. Anomaly detection
3. Hybrid detection
4. Scan detection
5. Profiling modules

The detailed investigation of these classes is out of the scope of this paper that need further discussions. The interested readers are referred to Dua and Du (2016) and Buczak and Guven (2015) for the detailed descriptions.

As a simple definition, intrusion refers to any type of unauthorized activities that try to harm or access to a digital system (Nisioti et al., 2018). In other words, any cyberattack that could cause possible damage to the data confidentiality, integrity, or availability will be subsumed under the umbrella of intrusion definition. On the other hand, IDS can be defined as a piece of software or hardware that has the duty of monitoring information system to identify malicious activities, users, and/or programs (Liao et al., 2013). According to Hindy et al. (2018), IDS systems can be classified into two large groups, including Signature-based Intrusion Detection System (SIDS) and Anomaly-based Intrusion Detection System (AIDS). SIDSs, also called Knowledge-based Detection systems, use the prior knowledge on detected threats to create accurate signatures to recognize a known attack. The key advantage of SIDSs is that they can detect known attacks with reasonable accuracy (Kreibich and Crowcroft, 2004). Nevertheless, these methods give poor performance in confronting polymorphic and zero-day attacks because there is no matching pattern (i.e. signature) in their databases (Khraisat et al., 2019). To encounter the problem, arising from the poor performance of SIDSs in zero-days and polymorphic attacks detection, AIDSs have been proposed.

Since AIDSs can overcome the limitations of SIDSs, they have attracted lots of attention from researchers (Agrawal and Agrawal, 2015). Anomaly-based IDSs leverage machine learning, statistical-based, or knowledge-based techniques to model the normal patterns of an information system. In this system, if a user or program deviates considerably from the normal pattern, the AIDS will consider it as a potential anomaly, and consequently as an intrusion based on further evaluations. Implementation of an AIDS consists of two important phases, including the training and the testing phase (Patcha and Park, 2007). During the training phase, the AIDS will try to extract the knowledge and learn normal behaviors. In the test phase, the system exposed to unseen data to evaluate the performance. The literature study on anomaly-based IDS reveals that different kinds of machine learning methods were used for developing AIDS. Based on the training phase, one can categorize these methods into five groups:

1. Supervised Learning
2. Unsupervised Learning
3. Semi-supervised Learning
4. Weakly Supervised Learning
5. Reinforcement Learning (RL)

Regardless of natural differences among these methods in their training phase and also matching input data to the labels, each method has its own advantages and disadvantages. For example, some machine learning techniques are expected to perform properly on specific kinds of attacks, but give a lacklustre performance on other kinds. Moreover, due to the broad types of cyber-attacks and network traffic volume and attributes, machine learning has risen to solve serious challenges, e.g. extreme computational and time complexity (Aljawarneh et al., 2018).

As authors in Buczak and Guven (2015) strongly mentioned, although there are a lot of efforts to use ML techniques in IDSs, there is a big difference between using ML in IDS and other applications of ML. In most ML applications, a model needs to be trained at the beginning and then it can be used for a long period. In most cases, there is no specific need to retrain the model in a short period. On the contrary, IDSs need to retrain ML models based on different events listed below:

- Daily training (Bilge et al., 2011).
- Based on analyst requirements (Jemili et al., 2007).
- After identifying a new intrusion (Hansen et al., 2007).
- Upon recognizing any important changes in network behavior (D'Alconzo et al., 2019).

As the first three cases happen unusually or on a regular basis, the rate of the last case is high in dynamic networks e.g., IoT. Regarding the network dynamicity, Network Traffic Monitoring and Analysis (NTMA) techniques recognize networks changes that trigger the ML models to be retrained based on new network behavior.

In this paper, we leverage the well-known ensemble learning approach, named boosting, for intrusion detection applications and present a comparative evaluation of its main variants, i.e. AdaBoost algorithms. Towards this end, we investigate the feasibility of intrusion detection by means of the most famous version of the AdaBoost algorithms, including Real AdaBoost (Schapire and Singer, 1999), Gentle AdaBoost (Friedman et al., 2000) and Modest AdaBoost (Vezhnevets and Vezhnevets, 2005). In Friedman et al. (2000), the authors compare the performance of Gentle AdaBoost with Real AdaBoost that shows the Gentle AdaBoost generally has a better performance. In addition, in Vezhnevets and Vezhnevets (2005) the authors compare the Modest AdaBoost with Gentle and Real AdaBoost. Their results show that Modest AdaBoost has a better performance.

The main idea behind ensemble learning is a combination of the multiple models instead of using a single model to increase performance in machine learning. For example, constructing a set of decision trees as a committee or ensemble of classifiers instead of a single decision tree. One may refer to bagging, boosting, and stacking as the outstanding machine learning techniques which benefit from the ensemble learning idea. Among them, boosting is the most powerful method. The main contributions of our work are summarized as follows:

- Showing the advantages of using booting approaches in IDSs and evaluate the well-known boosting approaches comparatively.
- Evaluating three of the most powerful boosting binary classifiers for accurate network intrusion detection based on the comparative results.
- Increasing the classification accuracy on the benchmark datasets by using the major preprocessing steps before using boosting algorithms.
- Leveraging five famous intrusion detection datasets to evaluate the models, and also provide example works from the literature to validate the results.

- Comparing the performance of the Real, Gentle, and Modest AdaBoost in different aspects e.g., running time and error rate in IDSs.
- Identifying the best models among three well-known AdaBoost algorithms to be used in IDS based on their performance on frequent retraining.

The rest of the paper is organized as follows. Section 2 provides a survey of the state-of-the-art literature tries to address intrusion detection issues. In Section 4, we present the necessary background on boosting, and the proposed methodology is described in Section 4, while the evaluation results and a comparative analysis is provided in Section 5. Finally, the conclusion is given in Section 6.

2. Previous work and background

In the context of IDS, scholars have deployed different techniques, such as machine learning algorithms, data mining, and ensemble methods to improve the accuracy of intrusion detection. This section surveys state-of-the-art researches on intrusion detection systems to clear the use of machine learning techniques in IDS. We review the related literature briefly in different ML techniques as follows. There are also some survey papers that focus on using ML techniques in IDS systems and also general IDS techniques. The interested readers are referred to read (Tsai et al., 2009; Buczak and Guven, 2015; Sultana et al., 2019). After dramatic advances in computer processing units and the availability of public datasets, DL models become increasingly popular for a wide range of tasks, like machine vision, pattern recognition, and natural language processing (MLP). Many DL-based papers are published in terms of using DL as a part of IDS (Papamartzivanos et al., 2019; Vinayakumar et al., 2019; Khan et al., 2019; Zeng et al., 2019; Tian et al., 2019; Faker and Dogdu, 2019). Scholars in the field of cyber security follow this trend towards using DL models for intrusion detection based on the efficiency of the methods. In Shone et al. (2018), the authors use stacked non-symmetric deep auto-encoder (NDAE) as an unsupervised classifier to decline the human intervention in IDS and also increase the detection accuracy. Al-Hawawreh et al. propose an IDS for industrial IoT based on two well-known DL algorithms including auto-encoder and deep feed-forward neural network. They claim that the proposed method is able to overcome the weakness of Network Intrusion Detection Systems (NIDS) in identifying industrial IoT attacks.

Classical machine learning algorithms are another family of techniques for IDS. Artificial neural network (ANN)-based intrusion detection has been introduced in Ali et al. (2018). The authors use Fast Learning Network (FLN) in combination with Particle swarm optimization (PSO) algorithm to decline the number of false negative/false positives detection. Tao et al. (2018) propose a method for intrusion detection based on a modified version of Support Vector Machine (SVM). More specifically, for feature selection and optimization of the SVM's parameters, they deploy the technique based on genetic algorithm (GA). Teng et al. design a self-adaptive ID system by means of SVM and Environment Classes, agents, roles, groups, and objects (E-CARGO) model. The experimental results reveal that the introduced method is more predominant compared to a single SVM model (Teng et al., 2018). In Alhakami et al. (2019) an AID system for monitoring IoT networks has been proposed based on non-parametric Bayesian model. In this paper, the automatic estimation of model's parameters and weights are adopted to increase the performance of clustering.

Ensemble methods as a subset of machine learning algorithms that have gained growing interest from the scientific community. This interest has been truly deserved because ensemble learning methods have proven themselves to be very efficient and surprisingly powerful in a wide range of use cases. Hence, many techniques have been implemented based on ensemble learning for intrusion detection. For example, Moustafa et al. designed an ensemble intrusion detection method for IoT networks (Moustafa et al., 2019b). This method is especially designed for attacks against well-known IoT protocols, including Domain Name System (DNS), Hyper Text Transfer Protocol (HTTP), and Message Queue Telemetry Transport (MQTT). In Gao et al. (2019) an adaptive ensemble learning technique has been deployed to find network attacks, including Denial of Service (DoS), Probe, R2L and U2R attacks. In order to improve the accuracy of classification, a combination of several classifiers, such as decision tree, random forest, kNN, and deep neural network (DNN) has been chosen. Liu et al. (2019) introduce a possible world-based AdaBoost algorithm, namely PwAdaBoost, to deal with the problem of uncertain data categorization. The authors claim that the introduced method capable of handling different types of uncertain data; hence, it can be used to deal with uncertainty in intrusion analysis. The work in Tang et al. (2020) by Tang et al. proposes to use AdaBoost algorithm, for low-rate denial of service (LDoS) attack detection in communication networks. Towards this end, the authors design and implement an improved version of AdaBoost algorithm, called MF-Adaboost, as well as identify a set of network traffic features as the most informative classification features. The simulation results show the effectiveness of the proposed method in LDoS attacks detection. As two basic methods for classification, Bayesian network and random tree have been used in Wang et al. (2016) to overcome the challenge of unbalanced data sample. In this work, the authors introduce a new intrusion dataset, called KDDcup99 dataset, which is provided to evaluate the introduced ensemble model. In Yuan et al. (2016), a semi-supervised version of Adaboost, one of the most practical ensemble-based algorithms, has been deployed against network anomaly detection. The simulations results show that the proposed technique has better performance than other learning algorithms.

It is noteworthy to mention that boosting algorithms also have been used for other classification tasks. For example, the authors in He et al. (2020) use AdaBoost algorithm for fault diagnosis to achieve high classification accuracy. According to simulation results, the proposed AdaBoost-based classification method obtains high fault diagnosis accuracy. Intrusion detection in Wireless Sensor Networks (WSNs) also become a target in Otoum et al. (2019) regarding the vulnerability of nodes. To that end, Restricted Boltzmann Machine-based Clustered IDS (RBC-IDS) is introduced by authors to monitor WSNs and detect intrusions. The experimental results show the reasonable accuracy rate of the proposed method, more than 97% accuracy under different settings, and also improved detection time than to a famous adaptive machine learning-based IDS. Table 1 provides a summary of the literature review.

3. Ensemble learning

Significant advantages of ensemble learning over using a single classifier motivates us to investigate the feasibility of using boosting algorithms for network intrusion detection. Before diving into the details of the evaluated methods, the necessary background on ensemble learning and boosting approach is provided in this section.

A broad definition of ensemble systems refers to use multiple learning models instead of a single model to obtain better performance. Sheela's paper is one of the earliest work on ensemble learning (Dasarathy and Sheela, 1979). After Sheela, Hansen and Salamon proposed an ensemble of neural networks to boost classification performance (Hansen and Salamon, 1990). Nevertheless, the turning point in ensemble systems came when Schapire published a paper in which he introduced Boosting procedure as a strong classifier. He showed a binary classification problem can be solved by constructing a group of moderately inaccurate classifiers (weak classifiers) (Schapire, 1990). The main idea behind boosting is to learn a combination of weak and simple classifiers, with an error rate rigorously less than random guessing, instead of trying to construct a single complex and strong classifier. The idea of an ensemble of classifiers is shown in Fig. 1. In the following, we consider that the mth weak classifier (binary

Table 1
Comparing literature of IDS Techniques.

Ref.	Method	Category	Technical contribution
Papamartzivanos et al. (2019)	MAPE-K+autoencoder (AE)+Sparse AE	Misuse detection in modern networks	Introduces a scalable, self-adaptive and autonomous method for misuse detection for modern large-scale modern networks by leveraging DL.
Vinayakumar et al. (2019)	Different classical and deep machine learning methods	Intrusion detection	Provides the comprehensive analysis of the performance of different machine learning techniques on different publicly available datasets.
Khan et al. (2019)	Stacked autoencoder	Intrusion detection	Introduces a new two-stage deep learning (TSDL) model through a stacked auto-encoder to efficiently implement a intrusion detection system.
Zeng et al. (2019)	Convolutional Neural Network (CNN)+ Long Short-Term Memory (LSTM)+ SAE	Encrypted traffic classification and intrusion detection	Proposes and establishes a light-weight framework based on DL algorithms for encrypted traffic classification and intrusion detection purposes.
Tian et al. (2019)	Distributed DL with CNN	Web attack detection	Provides a distributed method based on DL for web attack detection on edge devices.
Faker and Dogdu (2019)	Deep Feed-Forward Neural Network (DNN) + Random Forest + Gradient Boosting Tree (GBT)	Intrusion detection	Integrates big data technologies and DL algorithms to boost the performance of IDSs Moreover, it uses homogeneity metric as a feature selection method.
Shone et al. (2018)	Non-symmetric deep auto-encoder (NDAE)	Intrusion detection	Proposes a NDAE method for unsupervised feature learning and a new classification technique based on stacked NDAEs and random forest algorithm.
Ali et al. (2018)	Fast learning network + particle swarm optimization	Supervised intrusion detection system	Proposes using artificial neural network for intrusion detection; as well as introduces a learning model for FLN based on PSO, namely, PSO-FLN.
Tao et al. (2018)	Support vector machine + genetic algorithm	Intrusion detection	Provides a new SVM/GA-based intrusion detection algorithm, in which GA finds the optimal values for the weights and parameters of SVM.
Teng et al. (2018)	E-CARGO model+ decision trees + support vector machine	Intrusion detection	Develops a collaborative and adaptive intrusion detection technique through SVM and decision trees. Moreover, for modeling purposes, E-CARGO model has been used.
Alhakami et al. (2019)	Non-parametric Bayesian	Anomaly-based intrusion detection system	Proposes a Bayesian-based classifier for anomaly based intrusion detection. A feature selection mechanism has been integrated into the proposed method in order to prevent less informative features from compromising the modeling procedure.
Moustafa et al. (2019b)	AdaBoost ensemble learning with decision tree, Naive Bayes and artificial neural network	Intrusion detection	Identifies new statistical flow features to improve attacks detection process in IoT networks. Then, proposes an ensemble learning method through three machine learning algorithms for attacks categorization.
Gao et al. (2019)	Ensemble learning with decision tree, random forest, kNN, and deep neural network (DNN)	Intrusion detection	Provides an adaptive ensemble learning method through using various algorithms to improve the classification accuracy.
Tang et al. (2020)	An improved version of AdaBoost algorithm, called (MF-Adaboost)	low-rate denial of service (LDoS) attack detection	Based on AdaBoost algorithm, a new boosting algorithm has been introduced for identification of LDoS attack.
Wang et al. (2016)	Bayesian network and random tree	Intrusion detection	Provides an intrusion detection method based on Bayesian network and random tree using ensemble learning techniques to handle imbalanced sample classes in IDS datasets.
Yuan et al. (2016)	Semi-supervised Adaboost	Anomaly detection	Integrates ensemble and semi-supervised learning technique to create a new AdaBoost algorithm for network anomaly detection purposes.
Otoum et al. (2019)	Restricted Boltzmann Machine	Sensor network intrusion detection	Examines the possibility of using machine and DL techniques for intrusion detection in wireless sensor networks. Moreover, proposes a Restricted Boltzmann Machine-based Clustered IDS to identify attacks in wireless sensor networks.

classifier) is denoted by $H_m: X \rightarrow \{-1, +1\}$, for $m = 1, \ldots, M$. Furthermore, $(x \in X)$ represents input of the classification. According to Kuncheva (2014), one can combine the outputs of the M classifiers (i.e. $(X_1(x), \ldots, H_m(x))$) into a single ensemble classifier (i.e. $(H(x))$) in different kinds of ways, also called classifier fusion strategies. For example, one may refer to majority voting, majority weighted voting, minimum, maximum, average, median, and so on (Kuncheva, 2002). Given a weighted combination of the weak binary classifiers, we have a single strong classifier based on Eq. (1) where $(\omega_1, \ldots, \omega_M)$ denotes a

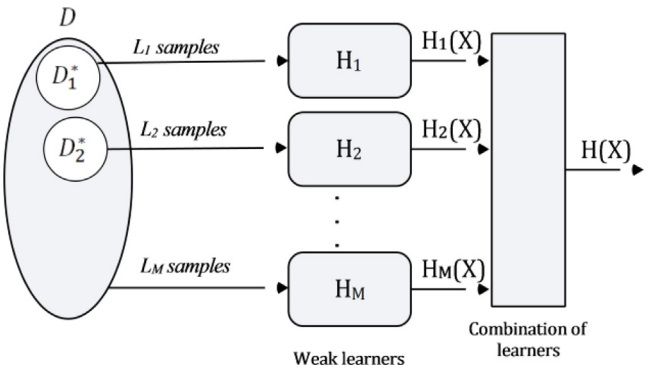


Fig. 1. The idea of ensemble of classifiers.

set of weights that are assigned to classifiers.

$$\left[H(x) = sign\left(\sum_{m=1}^{M} \omega_m H_m(x)\right)\right] \tag{1}$$

Depending upon the degree of accuracy of the previous weak classifiers, at each epoch of the algorithm, each instance from training dataset gains a weight and hence the algorithm gives special attention to the misclassified instances. The initial version of boosting algorithm is presented in Algorithm 1. From Algorithm 1, it is clear that the training dataset is separated into three partitions in a random way, and without replacement (D_1^*, D_2^*, and D_3^*). Considering a given data sample, if D_1^* and D_2^* be mutually agreed about the label of class, this will be the final decision for that data sample. The bunch of data samples on which the two first weak classifiers disagree produces the partition D_3^*.

After Schapire, considerable efforts have been taken to improve the idea of boosting. Also, several variants of boosting approach have been proposed in Freund and Schapire (1996, 1997) and Freund et al. (1996). AdaBoost (short for adaptive boosting) is the most famous boosting algorithm. Several variants of AdaBoost algorithm also have been proposed, such as Real AdaBoost, Gentle AdaBoost and Modest AdaBoost. AdaBoost came up with an idea to make use of weighted versions of the same training samples rather than to select a subset of the training samples in a random way. The algorithm repeatedly uses a different weighted version of the same training samples, and hence AdaBoost does not need a large volume of data for training. Fig. 2 depicts a graphical illustration of AdaBoost algorithm presented in Friedman et al. (2001). Furthermore, the detailed description of this algorithm is presented in Algorithm 2. In line 5 of the Algorithm 2, we have the Heaviside function ($h: \rightarrow \{0,1\}$). The Heaviside function is defined based on Eq. (2).

$$\left[h(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}\right] \tag{2}$$

Because of this fact that both $H_m(x_i)$ and y_i hold values in $\{-1,+1\}$, we have $h\left(H_m\left(x_i\right) - y_i\right) = 1$, if $H_m(x_i) \neq y_i$, and we have $h\left(H_m\left(x_i\right) - y_i\right) = 0$, if $H_m\left(x_i\right) = y_i$. Thus, err_m is the weighted error for the mth weak classifier. More specifically, the effect of this weighted error is to go up the weight of data samples misclassified by H_j. The interested reader is referred to Meir and Rätsch (2003) for more details on the algorithm. Boosting approaches have been quite successfully deployed in different real-world use cases like Huang et al. (2019), Afifi and Abdelhamed (2019), Gosztolya and Busa-Fekete (2019) and Zhao et al. (2019). Motivated by the powerful learning ideas behind the boosting approaches and these remarkable successes, we demonstrate the performance of AdaBoost algorithms for binary classification of network intrusion data. We apply Real AdaBoost, Gentle AdaBoost and Modest AdaBoost instead of the AdaBoost algorithm, because these variants have performed better performance on different types of real-world machine learning problems, e.g. text categorization (TC) and face detection. Moreover, AdaBoost is subject to overfitting in the presence of noisy data. On five IDS publicly available datasets and by using GML AdaBoost MATLAB Toolbox we show that the AdaBoost algorithms lead to the competitive classification results.

Algorithm 1 Boosting Procedure for Classification

Input: *Dataset* $D = \{d_1, d_2, ..., d_N\}$, with $d_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in X$ and $y_i \in \{-1,+1\}$.
Output: A classifier $H: X \rightarrow \{-1,+1\}$.

1: Randomly select, without replacement, $L_1 <$ N samples from D to obtain D_1^*.
2: Run the weak learner on D_1^*, yielding classifier H_1.
3: Select $L_2 < N$ samples from D, with half of the samples misclassified by H_1, to obtain D_2^*.
4: Run the weak learner on D_2^*, yielding classifier H_2.
5: Select all samples from D on which H_1 and H_2. disagree, constructing D_3^*.
6: Run the weak learner on D_3^*, yielding classifier H_3.
7: Construct the strong classifier as a majority vote: $H(\mathbf{x}) = sign(\sum_{m=1}^{3} H_m(\mathbf{x}))$.

Algorithm 2 Boosting Procedure for Classification

Input: *Dataset* $D = \{d_1, d_2, ..., d_N\}$, with $d_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in X$ and $y_i \in \{-1,+1\}$.
Output: A classifier $H: X \rightarrow \{-1,+1\}$.

1: Randomly select, without replacement, $L_1 <$ N samples from D to obtain D_1^*.
2: Run the weak learner on D_1^*, yielding classifier H_1.
3: Select $L_2 < N$ samples from D, with half of the samples misclassified by H_1, to obtain D_2^*.
4: Run the weak learner on D_2^*, yielding classifier H_2.
5: Select all samples from D on which H_1 and H_2. disagree, constructing D_3^*.
6: Run the weak learner on D_3^*, yielding classifier H_3.
7: Construct the strong classifier as a majority vote: $H(\mathbf{x}) = sign(\sum_{m=1}^{3} H_m(\mathbf{x}))$.

4. Using AdaBoost algorithms in IDSs

This section provides the details of AdaBoost algorithms in different aspects. Let us assume that we have N training data samples as follows:

$$\left(x_1, y_1\right), \ldots, \left(x_N, y_N\right), \text{ with } x_i \in \mathbb{R}^k \text{ and } y_i \in \{-1,+1\},$$

we can put these data samples in a random vector (X, Y). Here, X is an N-dimensional vector that models the network traffic features, and Y as an output variable shows the binary class label. The aim of a standard ML classification problem is to predict the class label for Y, depends on the features vector X. In response to $Y \in \{-1,+1\}$, we focus on the classification of binary problems.

4.1. Binary Real AdaBoost

Real AdaBoost is introduced by Schapire and Singer (1999). Real AdaBoost algorithm uses a set of weak classifiers to produce the final classifier results. In this algorithm, a weak classifier produces a class probability estimate. To calculate the contribution of a weak classifier to the final classifier, half of the logit transform of the weak classifier probability is estimated. The details of Real AdaBoost algorithm are provided in Algorithm 3. In this paper, we use decision Classification and Regression Trees (CART), proposed by Breiman (2017), as the weak classifiers and examine trees with different depths, see Breiman et al. (1984). CART is a classification technique in which historical data is used to create decision trees. Then, decision trees are leveraged for classifying of new data samples. In other words, CART is a predictive tree which helps to find the class of data sample by asking a set of Yes/No questions based on the value of a feature. Each input data sample navigates the tree based on the question to be directed to the

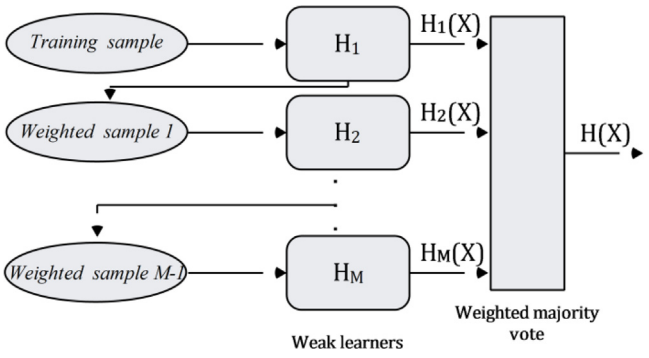


Fig. 2. Graphical description of AdaBoost algorithm.
Source: Inspired by Friedman et al. (2001).

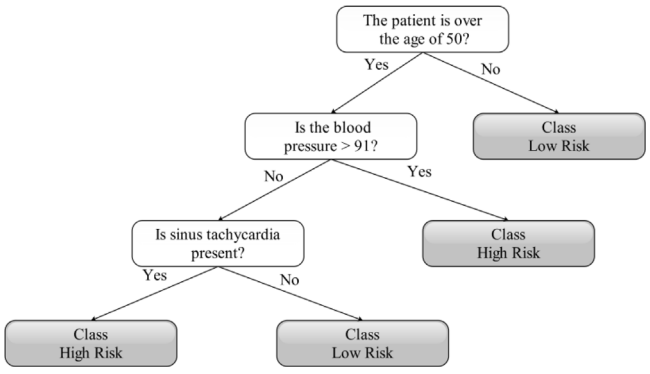


Fig. 3. CART example.

leaf of the tree that is the final class of the sample. An example of CART is presented in Fig. 3.

In Algorithm 3, it is clear that the main differences between Real AdaBoost and AdaBoost lie in lines 3 and 4. To be precise, for Real AdaBoost, these lines mean calculating and assessing the degree of confidence of the predicted class label for each data sample (i.e. $|H(x)|$). In other words, if $H(x)$ approaches zero or if there is a roughly equal number of positive/negative data points in the given subset of input data, it means a low confidence prediction and vice versa. There are two major differences between *Freund and Schapire's* AdaBoost algorithm and *Real AdaBoost*: (1) in Freund and Schapire's version, weak hypotheses are restricted to range $[-1, +1]$, however, in Real AdaBoost weak hypotheses are able to have range over all of $\mathbb{R}$, (2) Freund and Schapire's version proposes a specific value of α_m, where Real AdaBoost prescribes unspecified choice of α_m to control the impact of each of the weak classifiers (Schapire and Singer, 1999). Based on the results of Schapire and Singer (1999), the overall performance of Real AdaBoost on the training dataset is better than AdaBoost (discrete). The results also show clear advantages on the test dataset, in particular for a comparatively small number of epochs. Furthermore, the most interesting fact about Real AdaBoost seems to be related to larger problems, in which the superiority of this algorithm over discrete AdaBoost is greatest. In Schapire and Singer (1999), rational argumentations have not provided by the authors; however, this improvement might be due to the availability of large amounts of clean training data since for small and noisy datasets one can see a weaker performance as a result of overfitting.

Algorithm 3 Real AdaBoost algorithm for binary classification

Input: *Dataset* $D = \{d_1, d_2, ..., d_N\}$, with $d_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in X$ and $y_i \in \{-1, +1\}$. M, the maximum number of classifiers
Output: A classifier $H : X \to \{-1, +1\}$.

1: Initialize the weights $w_i = 1/N, i \in \{1....N\}$
2: **for** $m = 1$ **to** M **do**
3: Fit the class probability estimate $p_m(\mathbf{x}) = \hat{P}_w(y = 1|\mathbf{x})$, *using* w_i.
4: Set $H_m = \frac{1}{2}\log((1 - p_m(\mathbf{x}))p_m(\mathbf{x})) \in R$
5: Update the weights: $w_i \leftarrow w_i \exp(-y_i H_m(x_i))$
6: Renormalize to weights.
7: **end for**
8: Final Classifier: $H(\mathbf{x})$=sign$(\sum_{j=1}^{M} \alpha_j H_j(\mathbf{x}))$

4.2. Binary Gentle AdaBoost

Gentle AdaBoost algorithm is proposed by Friedman et al. in Friedman et al. (2000). The authors leverage adaptive Newton steps in the optimization of each step to find stationary point rather than exact optimization. The Gentle AdaBoost uses weighted least-squares regression for minimization of exponential error function $E\left[\exp(-yH(x))\right]$ at each iteration. The Gentle AdaBoost is more stable than the Real AdaBoost because the Gentle algorithm does not need to calculate log ratios. Numerically, log ratios can be unstable, causing very large updates. Empirical evidence in Friedman et al. (2000) reveals that the performance of Gentle AdaBoost and Real AdaBoost has been extensively compared against each other, or even the Gentle AdaBoost outperforms its rival, particularly when stability is a main concern.

4.3. Binary Modest AdaBoost

The primary goal for designing the Modest AdaBoost was to introduce a boosting algorithm with less generalization error, even if it means at the cost of higher training error (Vezhnevets and Vezhnevets, 2005). The Modest AdaBoost uses a natural stopping criterion, unlike the previous variants. The experimental results indicate that the Modest AdaBoost provides better performance than Gentle AdaBoost with regard to generalization test error; however, the algorithm declines training error at a much slower speed and the error rate sometimes does not reach zero. This algorithm improves the Gentle AdaBoost algorithm by means of an *inverted distribution* as a weighting scheme (see Algorithm 5). In the standard distribution (ω_i) the misclassified data samples by earlier weak learners gain higher weights, whereas in the inverted distribution ($\overline{\omega}_i$), the correctly classified data samples by earlier weak learners gain higher weights. In Algorithm 5, lines 6 and 7 compute the direct and inverted distributions. The expression in line 8 used as an update for the current weak classifier, so that it declines weak classifiers contribution, when it performs strongly well on data samples that have been already correctly classified with high margin. The key reason behind this idea is that the algorithm tries to make the weak classifiers work only in their domain.

Algorithm 4 Gentle AdaBoost algorithm for binary classification

Input: *Dataset* $D = \{d_1, d_2, ..., d_N\}$, with $d_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in X$ and $y_i \in \{-1, +1\}$. M, the maximum number of classifiers
Output: A classifier $H : X \to \{-1, +1\}$.

1: Initialize the weights $w_i = 1/N, i \in \{1....N\}$
2: **for** $m = 1$ **to** M **do**
3: Train $H_m(\mathbf{x})$ by weighted least-squares of $y - i$ to x_i, with weights w_i
4: Update $H(\mathbf{x}) \leftarrow H(\mathbf{x}) + H_m(\mathbf{x})$
5: Update $w_i \leftarrow w_i \exp(-y_i H_m(x_i))$ and renormalize to $\sum_i w_i = 1$
6: **end for**
7: Final Classifier: $H(\mathbf{x})$=sign$(\sum_{j=1}^{M} \alpha_j H_j(\mathbf{x}))$

Algorithm 5 Modest AdaBoost algorithm for binary classification

Input: *Dataset* $D = \{d_1, d_2, ..., d_N\}$, with $d_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in X$ and $y_i \in \{-1, +1\}$. M, the maximum number of classifiers
Output: A classifier $H : X \rightarrow \{-1, +1\}$.

Initialize the weights $w_i = 1/N, i \in \{1....N\}$
for $m = 1$ **to** M **do**
  while $H_m \neq 0$ **do**
    Train $H_m(\mathbf{x})$ by weighted least-squares of y_i to x_i, with weights w_i.
    Compute "inverted" distribution $\bar{w}_i$=(1-w_i) and renormalize to $\sum_i \bar{w}_i = 1$
    Compute $P_m^{+1} = P_w(y = +1, H_m(\mathbf{x}))$,$\bar{P}_m^{+1} = P_{\bar{w}}(y = +1, H_m(\mathbf{x}))$
    Compute $P_m^{-1} = P_w(y = -1, H_m(\mathbf{x}))$,$\bar{P}_m^{-1} = P_{\bar{w}}(y = -1, H_m(\mathbf{x}))$
    Set $H_m(\mathbf{x})$=(P_m^{+1}(1-P_m^{+1})-P_m^{-1}(1-P_m^{-1}))
    Update $w_i \leftarrow w_i \exp(-y_i\ H_m(x_i))$ and renormalize to $\sum_i w_i = 1$
  end while
end for
Final Classifier: $H(\mathbf{x})$=sign($\sum_{j=1}^{M} \alpha_j H_j(\mathbf{x})$)

5. Evaluation and results

In this section, the performance of the AdaBoost techniques are evaluated to show how the techniques confront with network intrusions in computer networks.

5.1. Description of the benchmark datasets

As the classification methods should be evaluated to reveal their performance, we use five publicly available intrusion datasets to compare them listed below. To show the efficiency of AdaBoost algorithms, we consider three types of datasets in terms of their volume. KDD-CUP99 is a high-volume dataset, UNSW-NB15, TRAbID and NSL-KDD are mediumsize datasets and CICIDS2017 is considered as a low-volume dataset.

5.1.1. KDD Cup '99 dataset

KDD Cup '99 is one of the widely used datasets in the context of intrusion detection purposes. Originally, the dataset has 41 features (k = 41) and more than five million records (N = 5,000,000). The KDD Cup '99 dataset is introduced for the Fifth International Conference on Knowledge Discovery and Data Mining through computer network simulations (Protić, 2018). This dataset is derived from 1998 DARPA datasets and four main group of attacks are included in it, covering denial of service, user to root, remote to local, and probe. The redundancy of records in the training set and the duplication of records in the test set are the main issues about the KDD Cup '99 dataset. In our simulations, we only use 39 features and 50,000 records of the dataset to evaluate the AdaBoost algorithms.

5.1.2. UNSW-NB15

This dataset was created in 2015 and contains the real modern normal traffics and the contemporary synthesized attacks. Nine different types of abnormal traffic are simulated by IXIA simulation tool, such as DoS, worms, and exploits. A key characteristic of this dataset is that it contains a combination of the real modern normal traffic patterns and the synthetical abnormal behaviors. The simulation period is reported to be 16 h. Originally, the dataset has 49 features (k = 49) and the total number of records is two million and 540,044 records (N = 2,540,044). For this study, we select a subset of the dataset as the training set and testing set from the different kinds of attacks and normal behavior. To evaluate the AdaBoost algorithms, we select 36,628 records of the dataset and 39 features. More information can be found in Moustafa and Slay (2015).

5.1.3. TRAbID

The TRAbID dataset is introduced in 2017 (Viegas et al., 2017). In this dataset, the real and valid data are generated through client–server interaction. Different types of services are considered for the user with normal behavior, including HTTP, SMTP, SSH, and SNMP. Moreover, two main groups of generated attacks are included which are port scans (FIN-Scan, NULL-Scan, ACKScan, OS Fingerprinting, Service Fingerprinting, UDP-Scan, XMAS-Scan) and DoS (ICMP flood, HTTP flood, SYN flood, SMTP flood, TCP keep alive) attacks. The simulation period for this dataset is 8 h, with more than 615 million traffic packets. To evaluate the AdaBoost algorithms we select 36,628 records of the dataset and 36 features.

5.1.4. NSL-KDD

NSL-KDD dataset is introduced by Canadian Institute for Cybersecurity in 2009 to deal with the issues related to the KDD CUP 99 dataset (Tavallaee et al., 2009), including a large number of redundant samples and producing surprising results on this dataset, even with applying very simple machine learning techniques. For that end, the dataset goes through some process, such as removing all the redundant samples in both training and testing sets and constructing a more challenging subset of the dataset. In our study, we use two subsets of the NSL-KDD dataset, namely KDDTrain+ and KDDTest+. The former is a 20% subset of the NSL-KDD train set and the latter is the full NSL-KDD test set. To evaluate the AdaBoost algorithms, we select 40,000 records of the dataset and 23 features.

5.1.5. CICIDS2017

Similar to NSL-KDD, CICIDS2017 dataset is introduced by Canadian Institute for Cybersecurity in 2017 in order to provide a dataset that contains the most up-to-date attacks for researchers in the area of cybersecurity (Sharafaldin et al., 2018). The dataset is captured for a 5-day period, from 9 a.m., Monday, July 3, 2017 to 5 p.m. on Friday July 7, 2017. Eight different types of attacks, including Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS have been considered in the CICIDS2017 dataset. This dataset contains more than 80 network traffic features. In this work, we only consider 65 features for our simulations and 3100 records.

5.2. Dataset preprocessing

Data preprocessing is a key step towards developing machine learning models since the quality of data can greatly affect the abilities of the model. Hence, it is absolutely crucial that we preprocess the selected datasets before feeding them into the AdaBoost algorithms.

The first step we take towards the preprocessing is dealing with NULL values. In the majority of intrusion detection datasets, there are always few NULL values. Regardless of the machine learning problem one confronts (e.g. regression and classification), it is a serious challenge for the model to deal with these NULL values, independently. Therefore, we intervened directly against the NULL values and removed them from the benchmark datasets.

Dealing with nominal categorical type variables is another integral preprocessing step in the context of machine learning. There are some features in the benchmark datasets which keep character type values. For example, in the UNSW-NB15, the features such as protocol, state and service hold nominal values (e.g. UDP, ARP, and CON). These features cannot be feed directly into a machine learning model. Thus, we deploy the available functions in order to transform the nominal categorical features to the appropriate form. For example, we can represent "CON" with 0 and "UDP" with 1.

The last but not least, another crucial stage towards data preprocessing is feature scaling. Considering the benchmark datasets, the values of some features are not on the same scale, e.g. in the UNSW-NB15 dataset, "Source bits per second" feature has big numbers in bit, whereas "Source to destination packet count" feature hold roughly

small integer numbers. Consequently, it is expected that our boosting models give more special attention on "Source bits per second" than "Source to destination packet count", which is a behavioral bias towards the feature with greater values. To address this issue, we carry out standardization.

5.3. Results and discussion

In this paper, we carried out K-Fold cross validation in order to explore the potential of the proposed classification techniques and the average exponential loss function for determining the test error rate based on Eq. (3):

$$\left[L\left(y, f\left(x\right)\right) = \frac{1}{N}\sum_{i=1}^{N} \exp(-y_i f\left(x_i\right)) \right] \tag{3}$$

In addition, to show the performance of the boosting algorithms in terms of running time due to retraining purposes, we consider the maximum number of Iteration = 100. Based on the number of Iterations, the running time of the algorithm is limited and the results are reliable to show that boosting algorithm can reach a stable error rate after a deterministic number of Iterations. To examine the performance of the boosting algorithms in case of running time, we use the GML AdaBoost MATLAB Toolbox. In addition we limit the amount of accessible memory and CPU by MATLAB. Then we executed the algorithms sequentially and release all resources before starting a new execution. To compare the performance of the boosting algorithms, we consider two aspects including accuracy and running time of them in different depths. After that, we conclude the evaluation by comparing the results in the two aspects to show the efficiency of the algorithms to be used by IDSs.

5.3.1. Comparing the accuracy of the AdaBoost algorithms

Fig. 4 compares the performance of Real, Gentle, and Modest AdaBoost algorithm on the five different benchmark datasets in case of error rate. We plot the test error of each algorithm for the datasets as the function of the number of iterations. This experiment is conducted for a maximum depth of 2. The lowest test error rate is associated with Gentle AdaBoost, nevertheless, the error curve shows that the performance of Real AdaBoost is at, or close to Gentle AdaBoost for many rounds. As seen on Fig. 4, Modest AdaBoost yield is worse than the other two ones, especially in the initial iterations and smaller datasets. One of the interesting facts about the evaluated classifiers is that they did not over-fit which is clear from the figure. Moreover, all the evaluated algorithms show better performance on the large datasets than the datasets with a lower number of training samples. In the case of retraining the models in IDS, the figure shows that the Modest AdaBoost is much weaker than Gentle and Real Adaboost. As an example, error rates of Real and Gentle AdaBoost are stable after about Iteration = 20 in Fig. 4.b, but Modest AdaBoost error rate is always higher than the Gentle and Real AdaBoost and it is not stable before Iteration = 70. Despite this fact that the running time of Modest AdaBoost is lower than its counterparts in our experiments, one needs more iteration to achieve the same performance as the Real and Gentle AdaBoost.

5.3.2. The impacts of depths on the performance of the AdaBoost algorithms

We also ran our classifiers with three more depths, i.e. depth = 3, 4, and 5, to analyze the behaviors of the AdaBoost variants, as depicted in Figs. 5, 6, and 7, respectively. When we increase the maximum allowed depth for the algorithms, the performance of the algorithms is improved in terms of the test error. For example, for UNSW-NB15 dataset, the average of test error after 100 rounds is Real = 0.1189, Gentle = 0.1185, and Modest = 0.3058 for the depth of 2, while these figures are Real = 0.0611, Gentle = 0.0647, and Modest = 0.1887 for the depth of 4. A similar pattern is also honest about the other datasets. From the figures, it is clear that the performance of the classifications is better than the case we used CART with a depth of 2 on the same datasets, especially for the first iterations. However, as we construct the deeper trees, it is more difficult to have a deep sense of the decision rules of the trees. In contrast to the deep trees, the trees with small depth generate decisions faster, and also they produced rules that are easy for the explanation. By comparing the test error curves in Figs. 5, 6 and, 7, it is clear that the performance of Modest AdaBoost is disappointing than its counterparts, in spite of the fact that we increase the maximum allowed depth. Vezhnevets et al. referred to the one known disadvantage of Modest AdaBoost algorithm in their paper (Vezhnevets and Vezhnevets, 2005) as they found out that Modest AdaBoost often cannot reach zero error on training data and it can be a possible reason for the poor performance of Modest AdaBoost than its rivals in our case. Also, in the case of retraining, the results of Figs. 5 to 7 show that Real AdaBoost and Gentle AdaBoost have the same performance. As the same example, in TRAbID dataset, error rates of Real and Gentle AdaBoost are stable after Iteration = 20. In contrast, the stability of error rate of Modest AdaBoost is improved by increasing the depth as it is stable after around Iteration = 60, Iteration = 50 and Iteration = 50 in depths 3, 4, and 5 respectively.

5.3.3. Overall comparison of accuracy and running time among the AdaBoost algorithms

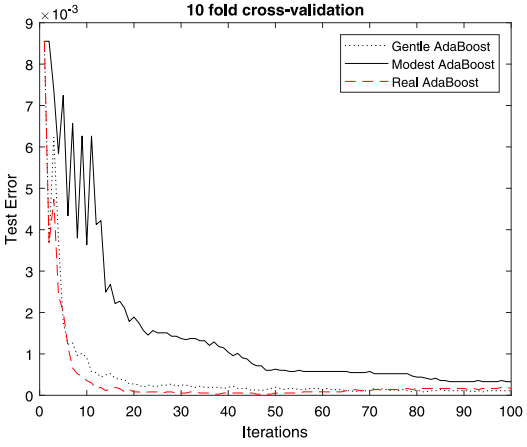
Table 2 compares the error rates of the AdaBoost algorithms on different datasets and depths. To confirm the results shown in the figures above, the table shows that Real and Gentle AdaBoost have the same error rates in different depths. There is no considerable difference between these algorithms, but Modest AdaBoost is worse than both of them in all depths. The results show that both Real and Gentle AdaBoost is relatively 70% better than Modest AdaBoost in case of error rate. Also, it shows that Gentle AdaBoost is about 2% better than Real Adaboost.

Table 3 compares the running times of the AdaBoost algorithms in different depths. The table shows that in most cases of depth = 2, Real AdaBoost has weaker performance compared to Gentle Adaboost. In other depths, Gentle AdaBoost is worse than Real AdaBoost in case of running time. Moreover, the table shows that Modest AdaBoost is the fastest algorithm among evaluated AdaBoost algorithms. The results show that the speed of both Real and Gentle AdaBoost are about 7% worse than Modest Adaboost. However, in the case of retraining, it is obvious that Modest AdaBoost is much weaker than Real and Gentle Adaboost. To reduce the number of Iterations to reach a stable error rate, we need to execute the model in more depths e.g., 4 or 5. Executing Modest AdaBoost in more depths increases the running time considerably that makes the model inappropriate for IDSs.

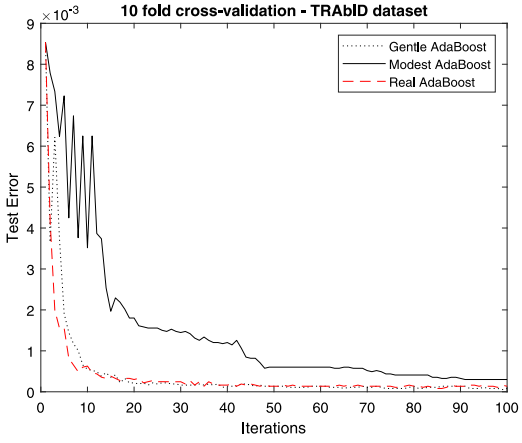
5.3.4. Concluding the evaluation

Comparing the error rates and running times of the evaluated AdaBoost algorithms shows the Gentle and Real AdaBoost have almost the same performance, but also they have better performance than Modest AdaBoost as they improve the error rate about 70% but they are slower than Modest only about 7% in intrusion detection systems. As the original paper of Modest AdaBoost (Vezhnevets and Vezhnevets, 2005) compared the algorithm with Gentle AdaBoost and briefly Real Adaboost, it is important to compare their results with ours. The results of the aforementioned work confirm that Modest AdaBoost has a higher error rate in small numbers of Iterations. They also mentioned that Modest AdaBoost has the same performance compared to Gentle and Real AdaBoost in case of running time, but there is no experiment to show the performance.

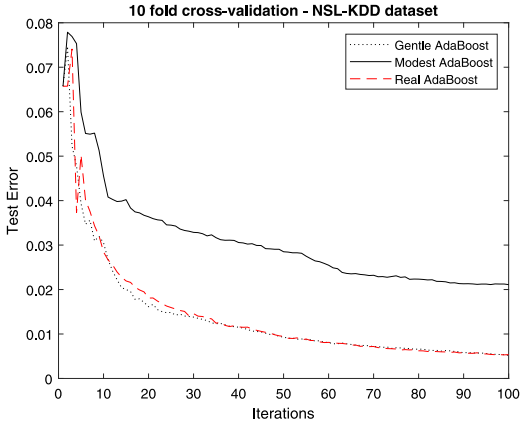
The comparison shows that our results confirm theirs, but also our results dive into more details e.g., running time and error rate, especially in the field of Intrusion detection.



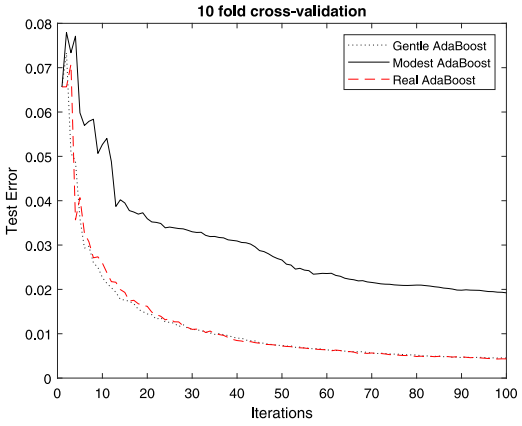
(a) UNSW-NB15



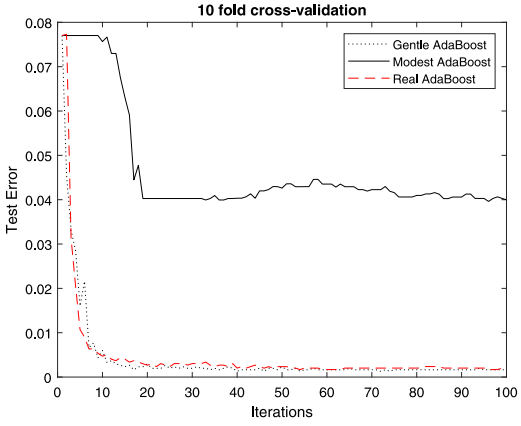
(b) TRAbID



(c) NSL-KDD



(d) KDD Cup '99



(e) CICIDS2017

Fig. 4. Comparison of Real, Gentle, and Modes AdaBoost on the five datasets (the maximum depth = 2).

Table 2
Comparing error rate of Adaboost algorithms on different datasets and depths.

	Depth = 2			Depth = 3			Depth = 4			Depth = 5		
	Real	Gentle	Modest	Real	Gentle	Modest	Real	Gentle	Modest	Real	Gentle	Modest
KDDCUP99	0.1189	0.1185	0.3058	0.0803	0.0825	0.2359	0.0611	0.0647	0.1887	0.0538	0.0573	0.1566
UNSW-NB15	0.00401	0.0050	0.0155	0.0032	0.0030	0.0111	0.0023	0.0027	0.0075	0.0016	0.0021	0.0070
TRAbID	0.0033	0.0041	0.0150	0.0034	0.0029	0.0107	0.0034	0.0033	0.0075	0.0022	0.0022	0.0064
NSL-KDD	0.1431	0.1399	0.3203	0.0927	0.1008	0.2432	0.0753	0.0778	0.2027	0.0753	0.0778	0.2027
CICIDS2017	0.0460	0.0529	0.4744	0.0439	0.0436	0.4075	0.0283	0.0350	0.3666	0.0276	0.0331	0.3418

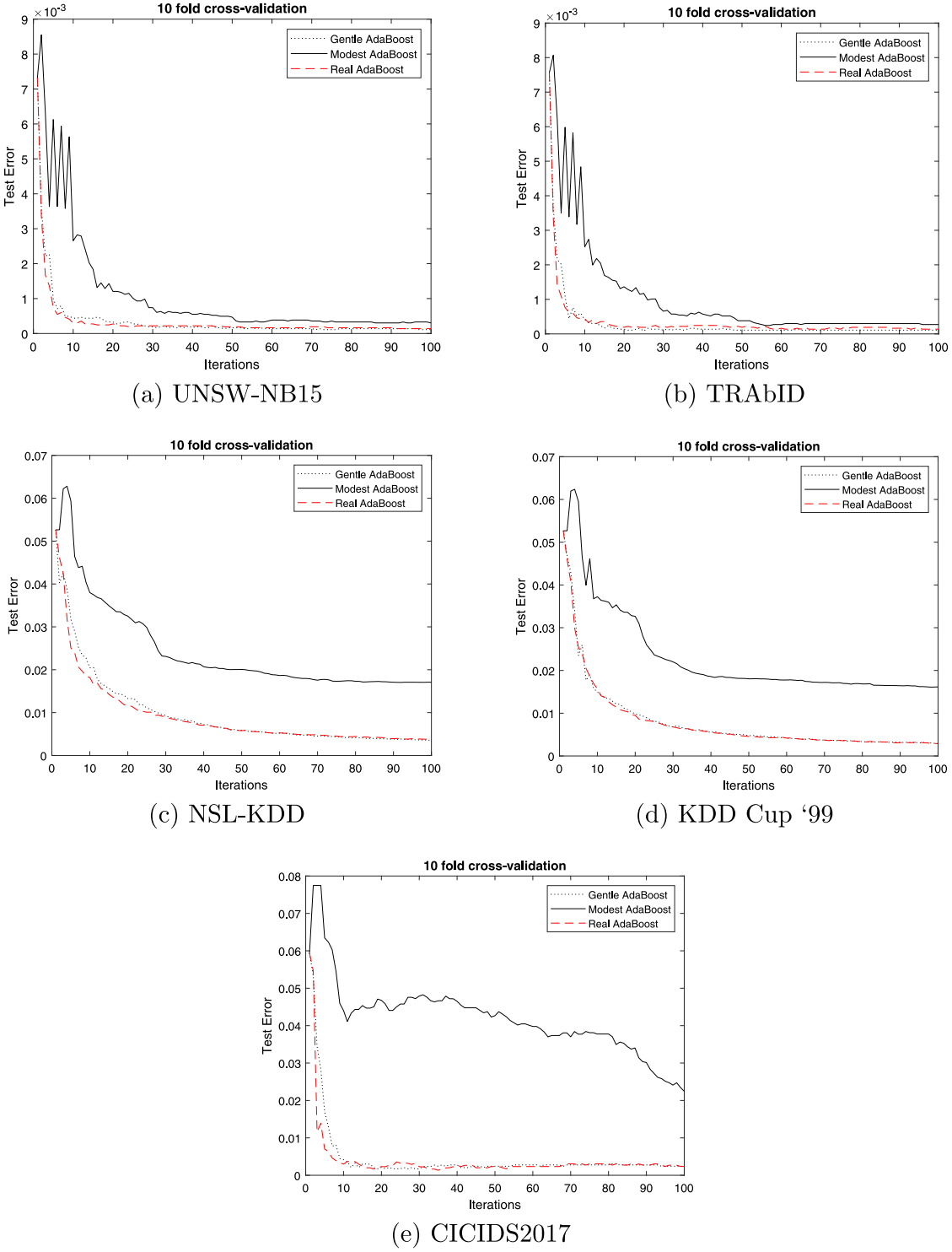


Fig. 5. Comparison of Real, Gentle, and Modes AdaBoost on the five datasets (the maximum depth = 3).

Table 3
Comparing running time of Adaboost algorithms on different datasets and depths in second.

	Depth = 2			Depth = 3			Depth = 4			Depth = 5		
	Real	Gentle	Modest	Real	Gentle	Modest	Real	Gentle	Modest	Real	Gentle	Modest
KDDCUP99	711,7	766,6	709,5	1170,4	1163,6	1097,1	1400,1	1580,9	1280,9	1602,8	1626,6	1587,3
UNSW-NB15	600,6	589,0	532,0	792,6	783,7	777,3	1149,2	1168,1	991,8	1143,1	1154,3	1142,0
TRAbID	543,8	457,4	448,9	680,2	737,151	663,1	940,8	1032,6	908,5	1177,4	1172,9	1152,0
NSL-KDD	376,9	384,3	353,9	604,9	606,7	561,9	812,6	835,6	789,3	1075,1	1068,5	989,2
CICIDS2017	55,8	55,4	52,8	85,2	88,7	82,3	106,0	108,0	105,9	133,4	132,3	131,9

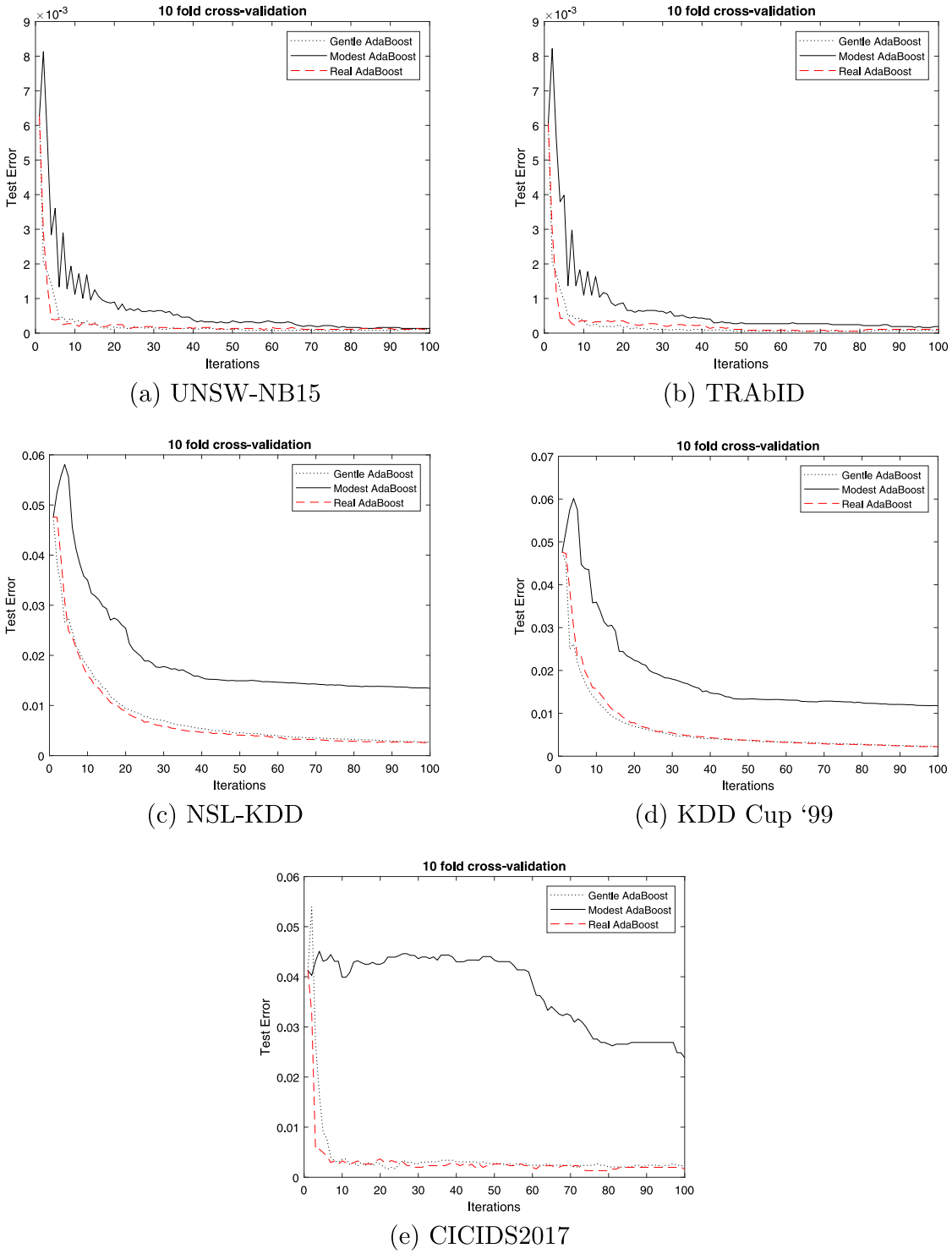


Fig. 6. Comparison of Real, Gentle, and Modes AdaBoost on the five datasets (the maximum depth = 4).

5.4. Validation of the results

The NSL-KDD dataset has been considered by a large number of researches. As mentioned earlier, on the two subsets of the NSL-KDD dataset, i.e. KDD Train+ and KDD Test+, an error rate of 0.0011, 0.0033, and 0.0011 is achieved by Real, Modest and Gentle AdaBoost algorithms respectively. These results can be directly compared to the work in Kim et al. (2014), in which a hybrid method combines C4.5 decision tree and one-class SVM models have been adopted on the same subset as we used. The experimental results show high intrusion detection accuracy, a detection rate around one for different settings. Sindhu et al. propose a lightweight intrusion detection system based on decision tree (Sindhu et al., 2012). Their model gives an impressive detection rate of 98.4%. Further papers can be found in the literature (Javaid et al., 2016; Tavallaee et al., 2009).

The UNSW-NB15 is one of the biggest intrusion detection datasets and the details of this dataset well investigated in the literature. In Bamakan et al. (2017), the authors developed a Ramp Loss K-Support

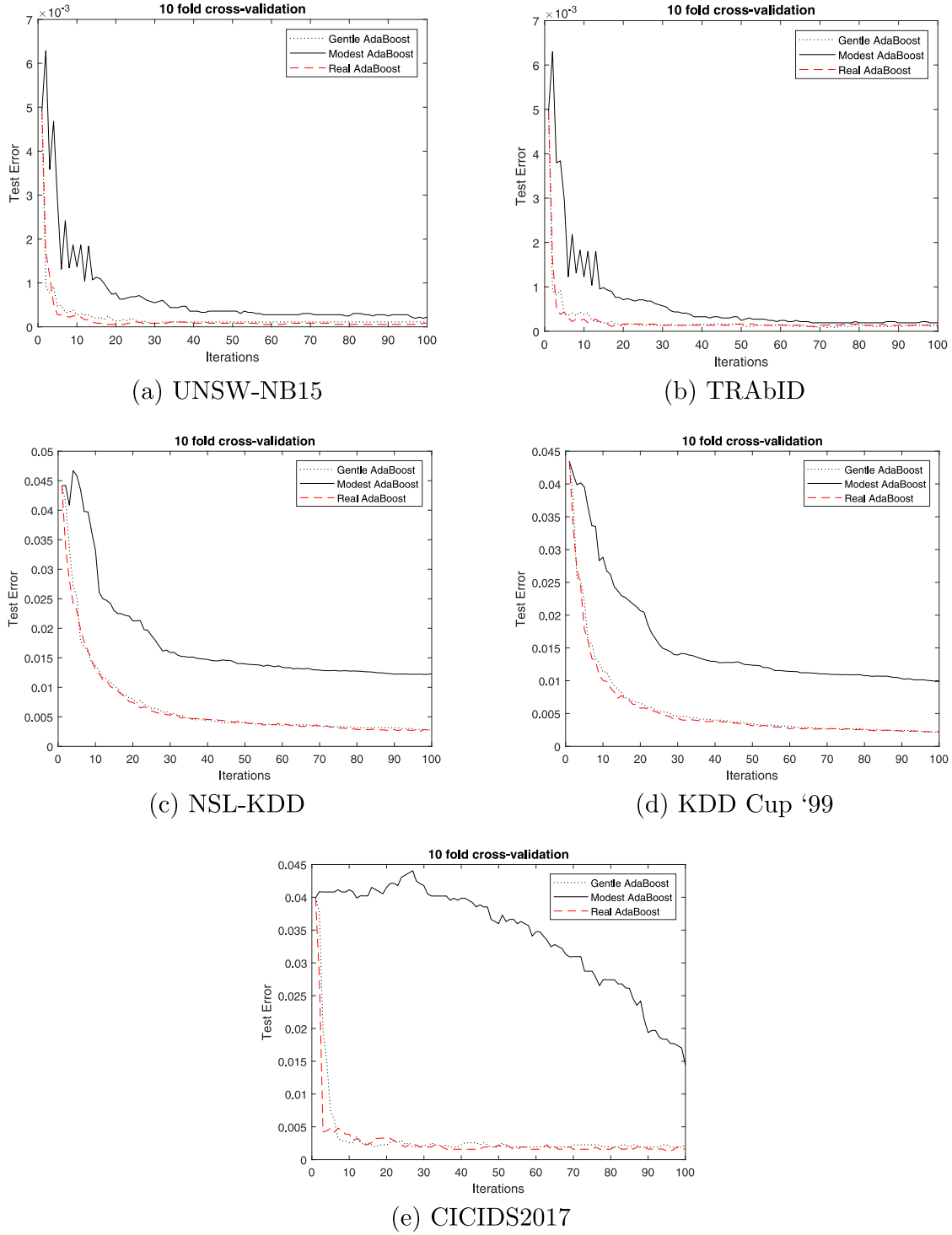


Fig. 7. Comparison of Real, Gentle, and Modes AdaBoost on the five datasets (the maximum depth = 5).

Vector Classification-Regression model for multi-class intrusion detection and used UNSW-NB15 as a benchmark dataset. They reported a total detection rate of 98.86%, which is remarkable. We can provide notable examples that achieve great performance on the UNSW dataset. The interested reader is referred to Moustafa et al. (2019a,b, 2018) for more details.

TRAbID is a recently released dataset in the context of IDS. In the related dataset paper, the authors employed two machine learning techniques, decision tree and Naive Bayes, for the identification of two main groups of attacks, i.e. Probing and DoS attacks. They reported high accuracy under different settings. For example, 99.41% for decision tree in DoS detection and 98.42% for Naive Bayes in Probing detection.

6. Conclusion

In this paper, three well-known AdaBoost binary classifiers are used as an intrusion detection system to process five public IDS datasets. Necessary preprocessing steps are applied and Real AdaBoost, Gentle AdaBoost, and Modest AdaBoost algorithms are used to deal with the binary classification problem. The simulation results on the datasets

reveal that the proposed methods are efficient and make boosting approach a strong player for intrusion data classification. By comparing, we observe that the performance of Real and Gentle AdaBoost is better than Modest AdaBoost on all datasets, and we hence recommend the use of these two AdaBoost variants for binary classification in the context of intrusion detection. The results show that Real AdaBoost and Gentle AdaBoost have the same performance as they are better than Modest AdaBoost in two aspects. First, they have about 70% better error rate compared to Modest Adaboost. Second, their error rate curves are stable after a few numbers of iteration as it is an important factor in IDSs, because the ML models in IDSs needs to be retrained frequently. In contrast, Modest AdaBoost is faster than Real and Gentle AdaBoost for about 7%, but is negligible as the Modest AdaBoost needs to be executed in more depths to reach the stability of error rate. Running the Modest AdaBoost in more depths increases the running time considerably that is a big challenge in IDSs. Validation of the results indicates that our boosting classifiers provide good results, in comparison with other works, and also they do not need sophisticated implementation and tuning thanks to the GML AdaBoost MATLAB Toolbox.

CRediT authorship contribution statement

Amin Shahraki: Conceptualization, Methodology, Validation, Investigation, Formal Analysis, Writing - original draft, Writing - review & editing, Visualization . **Mahmoud Abbasi:** Software, Resources, Data curation, Writing - original draft. **Øystein Haugen:** Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Afifi, M., Abdelhamed, A., 2019. Afif4: deep gender classification based on adaboost-based fusion of isolated facial features and foggy faces. J. Vis. Commun. Image Represent. 62, 77–86.

Agrawal, S., Agrawal, J., 2015. Survey on anomaly detection using data mining techniques. Procedia Comput. Sci. 60, 708–713.

Akhtar, N., Mian, A., 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access 6, 14410–14430.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M., 2015. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutor. 17, 2347–2376.

Alhakami, W., A.L.harbi, A., Bourouis, S., Alroobaea, R., Bouguila, N., 2019. Network anomaly intrusion detection using a nonparametric bayesian approach and feature selection. IEEE Access 7, 52181–52190.

Ali, M.H., Al Mohammed, B.A.D., Ismail, A., Zolkipli, M.F., 2018. A new intrusion detection system based on fast learning network and particle swarm optimization. IEEE Access 6, 20255–20261.

Aljawarneh, S., Aldwairi, M., Yassein, M.B., 2018. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J. Comput. Sci. 25, 152–160.

Bamakan, S.M.H., Wang, H., Shi, Y., 2017. Ramp loss k-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem. Knowl.-Based Syst. 126, 113–126.

Beyene, Y.D., Jäntti, R., Ruttik, K., 2017. Random access scheme for sporadic users in 5g. IEEE Trans. Wireless Commun. 16, 1823–1833.

Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M., 2011. Exposure: Finding malicious domains using passive dns analysis. In: Ndss, pp. 1–17.

Breiman, L., 2017. Classification and Regression Trees. Routledge.

Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A., 1984. Classification and Regression Trees. CRC Press.

Buczak, A.L., Guven, E., 2015. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Commun. Surv. Tutor. 18, 1153–1176.

Conti, M., Li, Q.Q., Maragno, A., Spolaor, R., 2018. The dark side (- channel) of mobile devices: A survey on network traffic analysis. IEEE Commun. Surv. Tutor. 20, 2658–2713.

D'Alconzo, A., Drago, I., Morichetta, A., Mellia, M., Casas, P., 2019. A survey on big data for network traffic monitoring and analysis. IEEE Trans. Netw. Serv. Manag. 16, 800–813.

Dasarathy, B.V., Sheela, B.V., 1979. A composite classifier system design: concepts and methodology. Proc. IEEE 67, 708–713.

Dua, S., Du, X., 2016. Data Mining and Machine Learning in Cybersecurity. Auerbach Publications.

Faker, O., Dogdu, E., 2019. Intrusion detection using big data and deep learning techniques. In: Proceedings of the 2019 ACM Southeast Conference. ACM, pp. 86–93.

Freund, Y., Schapire, R.E., 1996. Game theory, on-line prediction and boosting. In: Proceedings of the ninth annual conference on Computational learning theory, pp. 325–332.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of online learning and an application to boosting. J. Comput. System Sci. 55, 119–139.

Freund, Y., Schapire, R.E., et al., 1996. Experiments with a new boosting algorithm. In: Icml, Vol. 96. Citeseer, pp. 148–156.

Friedman, J., Hastie, T., Tibshirani, R., 2001. The Elements of Statistical Learning. In: Springer Series in Statistics, vol. 1.

Friedman, J., Hastie, T., Tibshirani, R., et al., 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Statist. 28, 337–407.

Gao, X., Shan, C., Hu, C., Niu, Z., Liu, Z., 2019. An adaptive ensemble machine learning model for intrusion detection. IEEE Access 7, 82512–82521.

Gosztolya, G., Busa-Fekete, R., 2019. Calibrating AdaBoost for phoneme classification. Soft Comput. 23, 115–128.

Graves, J.T., Acquisti, A., Christin, N., 2018. Should credit card issuers reissue cards in response to a data breach?: Uncertainty and transparency in metrics for data security policymaking. ACM Trans. Internet Technol. (TOIT) 18, 54.

Hansen, J.V., Lowry, P.B., Meservy, R.D., McDonald, D.M., 2007. Genetic programming for prevention of cyberterrorism through dynamic and evolving intrusion detection. Decis. Support Syst. 43, 1362–1374.

Hansen, L.K., Salamon, P., 1990. Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. 12, 993–1001.

He, Y.-L., Zhao, Y., Hu, X., Yan, X.-N., Zhu, Q.-X., Xu, Y., 2020. Fault diagnosis using novel adaboost based discriminant locality preserving projection with resamples. Eng. Appl. Artif. Intell. 91, 103631.

Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., Bellekens, X., 2018. A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. arXiv preprint arXiv:1806.03517.

Huang, Q., Chen, Y., Liu, L., Tao, D., Li, X., 2019. On combining biclustering mining and AdaBoost for breast tumor classification. IEEE Trans. Knowl. Data Eng. 32, 728–738.

Javaid, A., Niyaz, Q., Sun, W., Alam, M., 2016. A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-Inspired Information and Communications Technologies. formerly BIONETICS, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 21–26.

Jemili, F., Zaghdoud, M., Ahmed, M.B., 2007. A framework for an adaptive intrusion detection system using bayesian network. In: 2007 IEEE Intelligence and Security Informatics. IEEE, pp. 66–70.

Khan, F.A., Gumaei, A., Derhab, A., Hussain, A., 2019. A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access 7, 30373–30385.

Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques datasets and challenges. Cybersecurity 2, 20.

Kim, G., Lee, S., Kim, S., 2014. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Syst. Appl. 41, 1690–1700.

Kolias, C., Kambourakis, G., Maragoudakis, M., 2011. Swarm intelligence in intrusion detection: A survey. Comput. Secur. 30, 625–642.

Kreibich, C., Crowcroft, J., 2004. Honeycomb: creating intrusion detection signatures using honeypots. ACM SIGCOMM Comput. Commun. Rev. 34, 51–56.

Kuncheva, L.I., 2002. A theoretical study on six classifier fusion strategies. IEEE Trans. Pattern Anal. Mach. Intell. 24, 281–286.

Kuncheva, L.I., 2014. Combining Pattern Classifiers: Methods and Algorithms. John Wiley & Sons.

Liang, G., Weller, S.R., Zhao, J., Luo, F., Dong, Z.Y., 2016. The 2015 ukraine blackout: Implications for false data injection attacks. IEEE Trans. Power Syst. 32, 3317–3318.

Liao, H.-J., Lin, C.-H.R., Lin, Y.-C., Tung, K.-Y., 2013. Intrusion detection system: A comprehensive review. J. Netw. Comput. Appl. 36, 16–24.

Liu, H., Zhang, X., Zhang, X., 2019. Pwadaboost: Possible world based adaboost algorithm for classifying uncertain data. Knowl.-Based Syst. 186, 104930.

Meir, R., Rätsch, G., 2003. An introduction to boosting and leveraging. In: Advanced Lectures on Machine Learning. Springer, pp. 118–183.

METIS, 2020. Mobile and wireless communications enablers for the 2020 information society. URL: http://www.metis2020.com.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., Guizani, M., 2018. Deep learning for iot big data and streaming analytics: A survey. IEEE Commun. Surv. Tutor. 20, 2923–2960.

Moustafa, N., Adi, E., Turnbull, B., Hu, J., 2018. A new threat intelligence scheme for safeguarding industry 4.0 systems. IEEE Access 6, 32910–32924.

Moustafa, N., Slay, J., 2015. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference. MilCIS, IEEE, pp. 1–6.

Moustafa, N., Slay, J., Creech, G., 2019a. Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks. IEEE Trans. Big Data 5, 481–494.

Moustafa, N., Turnbull, B., Choo, K.R., 2019b. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet Things J. 6, 4815–4830.

Nisioti, A., Mylonas, A., Yoo, P.D., Katos, V., 2018. From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. IEEE Commun. Surv. Tutor. 20, 3369–3388.

Otoum, S., Kantarci, B., Mouftah, H.T., 2019. On the feasibility of deep learning in sensor network intrusion detection. IEEE Netw. Lett. 1, 68–71.

Papamartzivanos, D., Gómez Mármol, F., Kambourakis, G., 2019. Introducing deep learning self-adaptive misuse network intrusion detection systems. IEEE Access 7, 13546–13560.

Patcha, A., Park, J.-M., 2007. An overview of anomaly detection techniques: Existing solutions and latest technological trends. Comput. Netw. 51, 3448–3470.

Protić, D.D., 2018. Review of kdd cup'99, nsl-kdd and kyoto 2006+ datasets. Vojnotehnički glasnik 66, 580–596.

Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5, 197–227.

Schapire, R.E., Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. Mach. Learn. 37, 297–336.

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP, pp. 108–116.

Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q., 2018. A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Topics Comput. Intell. 2, 41–50.

Sindhu, S.S.S., Geetha, S., Kannan, A., 2012. Decision tree based light weight intrusion detection using a wrapper approach. Expert Syst. Appl. 39, 129–141.

Sreeram, I., Vuppala, V.P.K., 2019. Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm. Appl. Comput. Inform. 15, 59–66.

Sultana, N., Chilamkurti, N., Peng, W., Alhadad, R., 2019. Survey on sdn based network intrusion detection system using machine learning approaches. Peer-to-Peer Netw. Appl. 12, 493–501.

Tang, D., Tang, L., Dai, R., Chen, J., Li, X., Rodrigues, J.J., 2020. Mf-adaboost: Ldos attack detection based on multi-features and improved adaboost. Future Gener. Comput. Syst. 106, 347–359.

Tao, P., Sun, Z., Sun, Z., 2018. An improved intrusion detection algorithm based on ga and svm. IEEE Access 6, 13624–13631.

Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A., 2009. A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. IEEE, pp. 1–6.

Teng, S., Wu, N., Zhu, H., Teng, L., Zhang, W., 2018. Svm-dt-based adaptive and collaborative intrusion detection. IEEE/CAA J. Automat. Sinica 5, 108–118.

Tian, Z., Luo, C., Qiu, J., Du, X., Guizani, M., 2019. A distributed deep learning system for web attack detection on edge devices. IEEE Trans. Ind. Inf. 16, 1963–1971.

Tounsi, W., Rais, H., 2018. A survey on technical threat intelligence in the age of sophisticated cyber attacks. Comput. Secur. 72, 212–233.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y., Lin, W.-Y., 2009. Intrusion detection by machine learning: A review. Expert Syst. Appl. 36, 11994–12000.

Vezhnevets, A., Vezhnevets, V., 2005. Modest adaboost-teaching adaboost to generalize better. In: Graphicon, Vol. 12. pp. 987–997.

Viegas, E.K., Santin, A.O., Oliveira, L.S., 2017. Toward a reliable anomalybased intrusion detection in real-world environments. Comput. Netw. 127, 200–216.

Vinayakumar, R., Alazab, M., Soman, K.P., Poornachandran, P., AlNemrat, A., Venkatraman, S., 2019. Deep learning approach for intelligent intrusion detection system. IEEE Access 7, 41525–41550.

Wang, Y., Shen, Y., Zhang, G., 2016. Research on intrusion detection model using ensemble learning methods. In: 2016 7th IEEE International Conference on Software Engineering and Service Science. ICSESS, IEEE, pp. 422–425.

Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M., 2018. Hastids: Learning hierarchical spatial–temporal features using deep neural networks to improve intrusion detection. IEEE Access 6, 1792–1806.

Yuan, Y., Kaklamanos, G., Hogrefe, D., 2016. A novel semi-supervised adaboost technique for network anomaly detection. In: Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. ACM, pp. 111–114.

Zeng, Y., Gu, H., Wei, W., Guo, Y., 2019. Deep-full-range : A deep learning based network encrypted traffic classification and intrusion detection framework. IEEE Access 7, 45182–45190.

Zhang, C., Patras, P., Haddadi, H., 2019. Deep learning in mobile and wireless networking: A survey. IEEE Commun. Surv. Tutor. 21, 2224–2287.

Zhao, H., Yu, H., Li, D., Mao, T., Zhu, H., 2019. Vehicle accident risk prediction based on adaboost-so in vanets. IEEE Access 7, 14549–14557.