Expert Systems With Applications 162 (2020) 113864

Contents lists available at ScienceDirect

Expert Systems With Applications

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

M-AdaBoost-A based ensemble system for network intrusion detection

Ying Zhou *, Thomas A. Mazzuchi, Shahram Sarkani

Department of Engineering Management and Systems Engineering at the George Washington University, Washington, DC, USA

ARTICLE INFO

Keywords:
Network intrusion detection
Machine learning
Class imbalance
Boosting
Particle swarm optimization
Ensemble learning

ABSTRACT

Network intrusion detection remains a challenging research area as it involves learning from large-scale imbalanced multiclass datasets. While machine learning algorithms have been widely used for network intrusion detection, most standard techniques cannot achieve consistent good performance across multiple classes. In this paper we proposed a novel ensemble system based on the modified adaptive boosting with area under the curve (M-AdaBoost-A) algorithm to detect network intrusions more effectively. We combined multiple M-AdaBoost-A-based classifiers into an ensemble by employing various strategies, including particle swarm optimization. To the best of our knowledge, this study is the first to utilize the M-AdaBoost-A algorithm, which incorporates the area under the curve into the boosting process for addressing class imbalance in network intrusion detection. Compared with existing standard techniques, our proposed ensemble system achieved superior performance across multiple classes in both 802.11 wireless intrusion detection and traditional enterprise intrusion detection.

1. Introduction

Network intrusion detection remains a challenging research area as it involves learning from large-scale imbalanced multiclass datasets, where instances of intrusion are significantly fewer than those of normal traffic. Consequently, most machine learning approaches are only good at detecting some but not all traffic types and hence cannot achieve consistent good performance across multiple classes. To improve network intrusion detection, we herein propose a novel ensemble system based on the modified adaptive boosting with area under the curve (M-AdaBoost-A) algorithm. In our experiments, superior performance was achieved across multiple classes for two distinct intrusion detection domains: 802.11 wireless networks and traditional enterprise networks. Our proposed system is capable of recognizing minority classes, maximizing true positive rates (TPRs), and minimizing false positive rates (FPRs) across multiple classes.

While traditional enterprise networks have been studied extensively for network intrusion detection, wireless network intrusion detection has garnered increasing interest in recent years. The Aegean Wi-Fi Intrusion Dataset (AWID) is a family of datasets focused on intrusion detection in 802.11 wireless networks (AWID Website, 2019; Kolias, Kambourakis, Stavrou, & Gritzalis, 2016). 802.11 wireless networks are typically modeled as a group of users accessing a single wireless access point in a localized area. AWID datasets are relatively new and contain both a normal class and three attack classes: flooding, impersonation, and injection, as described in Table 1.

For traditional enterprise networks, the NSL-KDD (NSL-KDD Website, 2019; Tavallaee, Bagheri, Lu, & Ghorbani, 2009) dataset has served as a benchmark dataset for intrusion detection research. It is an improved version of the original KDD99 dataset collected by MIT Lincoln Labs. The NSL-KDD dataset tackles some of the inherent problems of the original KDD99 dataset by primarily removing duplicated records (Rodda & Erothi, 2016). As shown in Table 2, in addition to normal traffic, the NSL-KDD dataset contains four different classes of attack traffic: denial-of-service (DoS), probe, remote-to-local (R2L), and user-to-root (U2R).

The remainder of this paper will be organized as follows. Section 2 reviews some of the earlier work related to our study. Section 3 describes our proposed methodology and system framework in detail. Section 4 explains our experimental setup and presents the classification results, and Section 5 concludes this paper with discussions.

2. Related studies

Machine learning algorithms for network intrusion detection have been extensively studied, including bagging (Pfahringer, 2000), boosting (Hu, Hu, & Maybank, 2008; Mazini, Shirazi, & Mahdavi, 2019), hybrid ensemble (Alotaibi & Elleithy, 2016; Farid, Harbi, & Rahman,

* Corresponding author.
E-mail addresses: ycindyzhou@gwu.edu (Y. Zhou), mazzu@gwu.edu (T.A. Mazzuchi), sarkani@gwu.edu (S. Sarkani).

https://doi.org/10.1016/j.eswa.2020.113864
Received 7 April 2020; Received in revised form 7 August 2020; Accepted 7 August 2020
Available online 13 August 2020
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

Table 1
Attack types in AWID dataset.

Type	Description
Flooding	Flooding attack causes the service interruption of a specific host/server by flooding it with large quantities of useless traffic or external communication requests
Impersonation	Impersonation attack assumes the identity of a legitimate network device
Injection	Injection attack uses existing vulnerabilities in applications to inject a code/string for execution that exceeds the allowable input to the system

Table 2
Attack types in NSL-KDD dataset.

Type	Description
DoS	Denial-of-service attack consumes system resources such that legitimate users and requests are denied of or remain waiting for resources
Probe	Probe attack gathers system information to exploit discovered vulnerabilities
R2L	Remote-to-local attack gains local access to a machine from a remote location by guessing the password
U2R	User-to-root attack obtains local access to a system and exploits the system for weak points to obtain root privileges

2010), genetic fuzzy systems (Abadeh, Mohamadi, & Habibi, 2011), hidden naive Bayes (Koc, Mazzuchi, & Sarkani, 2012), decision tree (Jaiganesh, Mangayarkarasi, & Sumathi, 2014), extreme learning machine (Fossaceca, Mazzuchi, & Sarkani, 2015), random forest (Farnaaz & Jabbar, 2016), artificial neural network (AkashdeepI & Kumar, 2017), stacked autoencoders (Aminanto & Kim, 2017), and support vector machine (Thaseen & Kumar, 2017). One of the challenges associated with network intrusion detection is the class imbalance between normal traffic and attack traffic. Consequently, more robust methods are required for network intrusion detection to achieve consistent good performance across multiple classes. To address class imbalance for improving network intrusion detection, various data resampling strategies have been utilized, such as undersampling of the majority class (Mikhail, Fossaceca, & Iammartino, 2019), oversampling of the minority class (Yang, Zheng, Wu, & Yang, 2019), and the combination of undersampling and oversampling (AkashdeepI & Kumar, 2017). However, the optimal class distribution is unknown, and the selection of the resampling ratio is often subjective and needs to be adjusted to improve classification performance. Wang and Yao (2012) addressed various class imbalance challenges and indicated that the oversampling strategy caused overfitting to the minority classes with low recall and high precision values, and that the undersampling strategy suffered from performance loss on the majority class.

Resampling methods are often considered as a strategy to mitigate class imbalance at the data level. At the algorithmic level, a number of solutions based on the widely used ensemble method, AdaBoost, have been proposed to address class imbalance. Although AdaBoost has gained popularity in machine learning, it is well known that AdaBoost cannot address class imbalance effectively (Joshi, Agarwal, & Kumar, 2002; Sun, Kamel, Wong, & Wang, 2007; Wang & Yao, 2012). AdaBoost is an algorithm that improves classification performance by combining multiple weak learners into one strong learner. It iteratively updates the weights of samples based on classification errors such that the weights of misclassified samples increase and the weights of correctly classified samples decrease. Consequently, classifiers that focus more attention on misclassified samples are selected rather than the minority class samples. Because AdaBoost focuses on classification accuracy, the algorithm tends to be biased toward the majority class, as the majority class contributes more to the overall classification accuracy (Galar, Fernández, Barrenechea, Bustince, & Herrera, 2011). Mikhail et al. (2019) indicated that AdaBoost alone does not perform well for network intrusion data and presented the semi-boosted nested model. Chawla, Lazarevic, Hall, and Bowyer (2003) proposed SMOTEBoost, which is a variant of AdaBoost based on combining the SMOTE algorithm and the boosting procedure for learning the imbalanced dataset in network intrusion. However, the optimal SMOTE sampling ratio has to be determined. Joshi, Kumar, and Agarwal (2001) proposed RareBoost, which is an enhancement to AdaBoost, in which different treatments were performed to positive and negative predictions to classify rare classes more effectively. However, if the conditions that the true positive rate is greater than the false positive rate and the true negative rate is greater than the false negative rate are not satisfied, the RareBoost algorithm will collapse. In our initial experiments using RareBoost, we discovered that such a constraint is a strong condition not yet satisfied by our data. Cost-sensitive boosting is another direction for class imbalance learning. Various approaches, such as AdaCost (Fan, Stolfo, Zhang, & Chan, 1999), CSB0, CSB1, CSB2 (Ting, 2000), AsymAda (Viola & Jones, 2002), AdaC1, AdaC2, AdaC3 (Sun et al., 2007), CSAda (Masnadi-Shirazi & Vasconcelos, 2007), CGAda (Landesa-Vázquez & Alba-Castro, 2012), AdaDB (Landesa-Vázquez & Alba-Castro, 2013), TCSBoost (Ryu, Jang, & Baik, 2017), and BAdaCost (Fernández-Baldera, Buenaposada, & Baumela, 2018) have been proposed to minimize the total misclassification cost. Cost-sensitive boosting employs cost information that biases the boosting toward the minority class; however, the cost information is not readily available and searching for the optimal cost is time consuming. Li, Zhou, Zhai, Li, and Shao (2019) proposed the AdaBoost-A algorithm, which is an improved AdaBoost algorithm. The AdaBoost-A algorithm incorporates the area under the curve (AUC) into the error calculation during the boosting phase of AdaBoost. The AdaBoost-A algorithm redefines the error performance metric of the AdaBoost algorithm by comprehensively considering the effects of misclassification probability and the AUC. Because the AUC can capture the effects of imbalanced data on the classifier, the AdaBoost-A algorithm can focus more on the minority class samples to avoid a biased classifier favoring the majority class. Our proposed ensemble method was developed utilizing the AdaBoost-A concept with several modifications and extensions. Moreover, we compared the performance of our proposed approach to those of SAMME (Zhu, Zou, Rosset, & Hastie, 2009) and AdaBoost.M1 (Freund & Schapire, 1996), which are two widely used standard boosting techniques in the multiclass setting. Both SAMME and AdaBoost.M1 are accuracy-oriented boosting algorithms, which tend to be biased toward the majority class as the majority class contributes more to the overall classification accuracy. Therefore, our proposed ensemble method aims at outperforming both of these algorithms, especially in recognizing minority classes, maximizing TPRs, and minimizing FPRs across multiple classes.

To further improve network intrusion detection, numerous approaches to combine multiple classifiers into an ensemble have been studied. Mukkamala, Sung, and Abraham (2005) combined neural networks, support vector machines, and multiple adaptive regressive splines into an ensemble by simple majority voting to detect network intrusion. Using a more sophisticated approach, Aburomman and Ibne Reaz (2016) applied the particle swarm optimization (PSO) algorithm to combine multiple k-nearest neighbor and support vector machine learners into an ensemble for network intrusion detection. PSO (Kennedy & Eberhart, 1995) is a heuristic global optimization algorithm based on the swarm intelligence of bird and fish flock movements. The PSO algorithm offers many advantages (Beheshti & Shamsudding, 2013). First, PSO involves few or no assumptions regarding the problem being optimized and does not require the optimization problem to be differentiable as with other classic optimization methods. Next, PSO can handle an extremely large search space for candidate solutions with easy implementation and fast convergence. Most importantly, the improved version of the PSO algorithm avoids premature convergence by offering more control of the parameters. Owing to these advantages, the PSO algorithm is widely used across many applications. In network intrusion detection, while PSO can be used for combining classifiers, it has been

more frequently used for feature selection and parameter tuning. PSO is applied to obtain the optimum feature subset, which is then fed into classifiers (Malik, Shahzad, & Khan, 2015; Tu, Chuang, Chang, & Yang, 2006; Zainal, Maarof, & Shamsuddin, 2007; Zhang, Gao, & Wang, 2008). Furthermore, PSO is employed to optimize the parameters of various algorithms (Chen, Qian, & Chen, 2009; Michailidis, Katsikas, & Georgopoulos, 2008; Srinoy, 2007; Tian & Liu, 2010; Xu, Rui, & Xiao, 2011; Zhou, Li, & Li, 2009). Additionally, PSO is utilized for both feature selection and parameter tuning (Ma, Liu, & Liu, 2008; Wang, Hong, Ren, & Li, 2009). In our study, both the PSO algorithm and simple majority voting (SMV) were studied to fuse multiple M-AdaBoost-A-based classifiers into an ensemble to further improve network intrusion detection. Hence, our proposed system can be considered as a multiple-boosting-fused ensemble.

3. Methodology

Our proposed M-AdaBoost-A-based ensemble system has two variants: M-AdaBoost-A-SMV and M-AdaBoost-A-PSO. Although the two variants involve different methods of fusing the ensemble, they were both developed based on the AdaBoost-A concept (Li et al., 2019) with several modifications and extensions. The distinctions between our proposed M-AdaBoost-A based ensemble and AdaBoost-A (Li et al., 2019) are highlighted as follows:

First, the AdaBoost-A algorithm was only applied to classify binary classes for small datasets in the study of Li et al. (2019), while our proposed M-AdaBoost-A based ensemble was extended to multiclass classification with the one-versus-rest strategy for the large datasets investigated in this study. Multiclass classification can be transformed into multiple binary classification problems using one-versus-rest and one-versus-one strategies. In our study, the one-versus-rest binarization approach was selected for computational efficiency. The binarization enabled us to utilize the AUC to mitigate class imbalance and improve the classification performance of intrusive minority classes.

Next, unlike Li et al. (2019) AdaBoost-A algorithm, which uses decision stumps (one-level decision trees) as weak base learners, our M-AdaBoost-A ensemble utilizes decision trees with a maximum depth of five as weak base learners. Decision stumps are typically used as weak base learners for AdaBoost; however, our M-AdaBoost-A ensemble with decision stumps as weak learners yielded poor model performance in the initial experiments. Decision stumps are too weak to satisfy the boostability condition in our application. Cortes, Mohri, and Syed (2014) indicated that for some challenging data, boosting decision stumps was not adequate to achieve a high performance level. Mukherjee and Schapire (2013) studied the optimal requirements of weak classifiers for boosting. To obtain the suitable weak learners that are neither too weak nor too strong, we investigated a range of values from 1 to 10 for the maximum depth of decision trees. Based on our experiments, decision trees with a maximum depth of five were selected as weak learners in our M-AdaBoost-A ensemble as they demonstrated good model performance during the training phase.

Subsequently, while boosting by reweighting with full-sized training data is typically performed, our M-AdaBoost-A ensemble used boosting by resampling with a predefined small sampling ratio to efficiently handle large datasets. Two implementations of boosting exist: boosting by reweighting and boosting by resampling. Boosting by reweighting can only be applied to base learners that can handle observation weights. Alternatively, boosting by resampling can be applied to any base learner. Seiffert, Khoshgoftaar, Van Hulse, and Napolitano (2008) empirically evaluated the differences between these two boosting implementations and recommended boosting by resampling on imbalanced data. Hence, boosting by resampling was selected in our M-AdaBoost-A based ensemble. More specifically, at each boosting iteration, a small proportion of the data from the entire training data was drawn with replacement based on the observation weights and the predetermined sampling ratio. Furthermore, we ensured that at least one instance from each of the binary classes (one versus rest) was included in the selected subsample. This selected subsample was used instead of the full sample to train the base weak learner at the current iteration. The accuracy of the boosting is expected to be enhanced by increasing randomness through using small subsamples at each iteration. Using subsamples in place of full-sized training data can overcome the overcapacity of the base learners and reduce the computational cost, especially when handling large-scale data. The introduction of subsamples into boosting iterations can be regarded as a hybrid of bagging and boosting or stochastic boosting. Similar subsampling ideas have been used in boosting trials in multiple studies (Fossaceca et al., 2015; Friedman, 2002; Webb, 2000; Xia & Hoi, 2013).

Finally, in the study by Li et al. (2019), PSO was utilized to optimize the coefficients of AdaBoost-A weak learners within boosting iterations. In our proposed approach, because boosting by resampling introduced diversity and resulted in classification outcome variations, strategies such as SMV and PSO were applied to combine multiple M-AdaBoost-A-based classifiers. In the next two sections, each variant of our proposed M-AdaBoost-A-based ensemble is described in detail.

3.1. M-AdaBoost-A-SMV

The M-AdaBoost-A-SMV ensemble can be considered as a set of multilevel nested models. The terms “ensemble”, “expert learner”, “sub-learner” and “base learner” are defined at different levels from the top to the bottom. The diagram in Fig. 1 depicts the M-AdaBoost-A-SMV system architecture. The utilization of the SMV strategy to fuse the M-AdaBoost-A-SMV ensemble is computationally inexpensive and suitable for efficiently handling large data.

As shown in Fig. 1, at the top level, the entire collection of the models is referred to as the M-AdaBoost-A-SMV ensemble. At the next lower level, it comprises 10 multiclass expert learners denoted as M-AdaBoost-A-EL. Further down, each expert learner is composed of multiple binary sub-learners. The one-versus-rest binary sub-learner utilizes the M-AdaBoost-A algorithm with decision trees as weak base learners. Each M-AdaBoost-A binary sub-learner determines the one-versus-rest outcomes. Then each expert learner combines K binary sub-learners and determines the class predictions belonging to one of the K classes. Finally, the SMV strategy determines the predictions of the M-AdaBoost-A-SMV ensemble by considering the outcomes from 10 expert learners. Veeramachaneni, Yan, Goebel, & Osadciw (2007) stated that ensembles with as few as 10 classifiers are adequate to sufficiently reduce error. Multiple studies (Aburomman & Ibne Reaz, 2016; Kausar, Ishtiaq, Jaffar, & Mirza, 2010; Veeramachaneni et al., 2007; Yang, Zhang, & Wang, 2009) were able to improve classification performance by fusing ensembles with 4 to 12 learners. With a manageable computational cost, the 10 expert learners in our ensemble are expected to make complementary contribution for improving classification performance.

The M-AdaBoost-A binary classifiers are considered as the core part in our proposed M-AdaBoost-A-SMV ensemble. The pseudocode of the binary classifier M-AdaBoost-A is outlined in Fig. 2. As highlighted in the pseudocode, we introduced two additional input parameters including d for the maximum depth of decision tree, and p for the sampling ratio. The introduction of sampling ratio p shows the main difference between our proposed M-AdaBoost-A algorithm and the original AdaBoost-A algorithm (Li et al., 2019). Within each boosting trial, our approach used the small subsample based on the observation weights and the sampling ratio to train the base learner, whereas the original AdaBoost-A used the full-sized sample. The requirement that at least one instance from each class must present in our subsample is crucial to prevent the minority class from being ignored. While the original AdaBoost-A does not have this requirement, we considered that this requirement is necessary to deal with the severe class imbalance in our datasets. Moreover, both subsampling with replacement and subsampling without replacement were experimented initially. We chose subsampling with replacement in our boosting trials as it yielded significantly better performance than

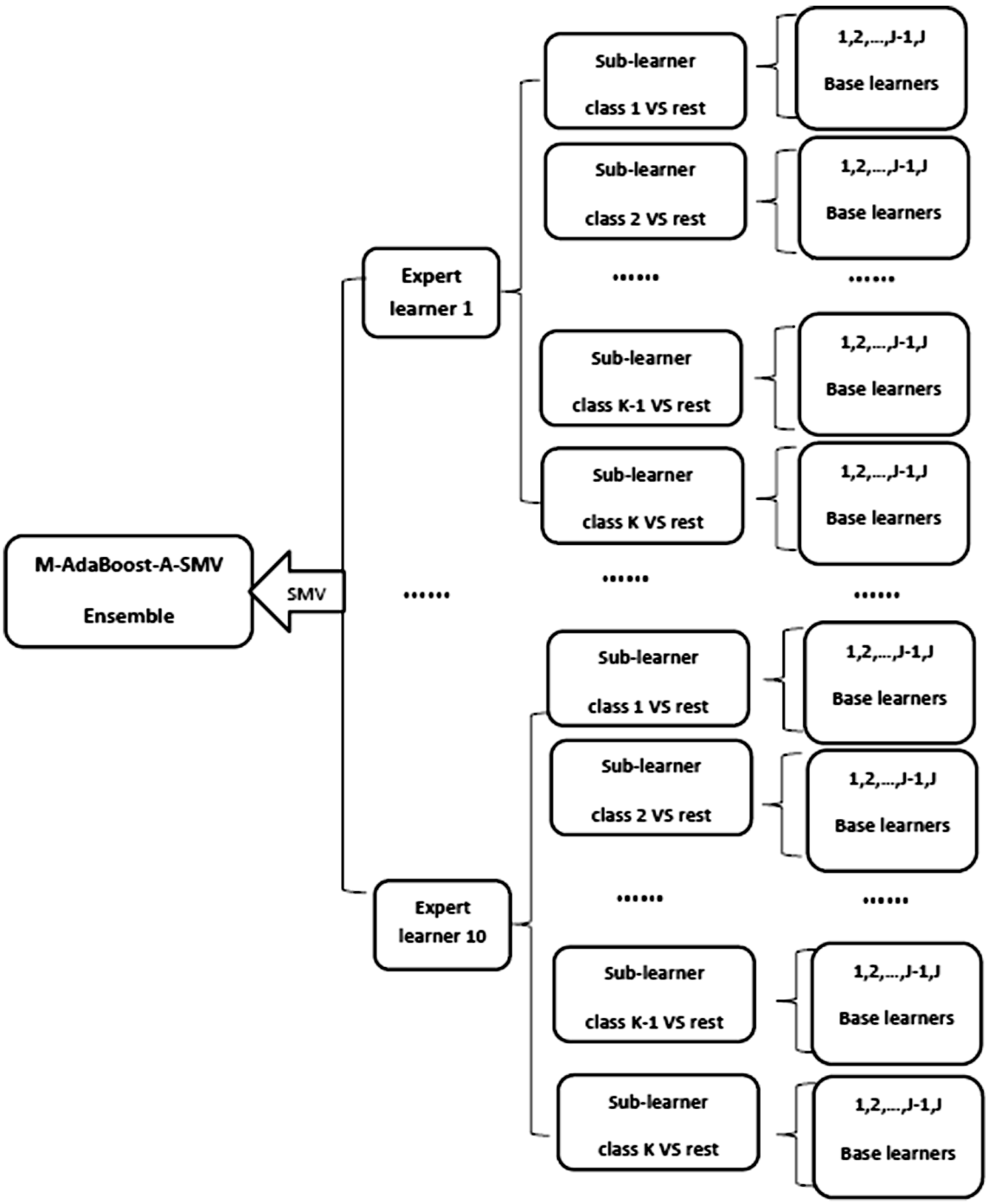


Fig. 1. M-AdaBoost-A-SMV ensemble framework.

subsampling without replacement. While during each boosting iteration, the base learner is trained on the resampled subset of the training data, the misclassification errors are evaluated on the entire training data; subsequently, the observation weights for the entire training data are updated according to the error performance metric.

As shown in Fig. 2, for M-AdaBoost-A algorithm, we adopted the same error performance metric e_j as in Li et al. (2019). The error performance metric e_j is a product of multiplying both the classification error rate and the AUC. The AUC is the area under the receiver operating characteristic curve, which is a plot of FPRs on the x-axis and TPRs on the y-axis. For data with class imbalance, biased classifiers can classify almost all the instances into the majority negative class and yield both small TPRs and small FPRs. The AUC of the biased classifiers favoring the majority class will be small; consequently, the measurement of (1-AUC) will be large even though the misclassification rates are small. Therefore, because the AUC captures the effects of imbalanced data on the classifiers, the M-AdaBoost-A algorithm does not produce biased classifiers, favoring the majority class. In other words, unlike the standard AdaBoost algorithm that only focuses on misclassified samples, our M-AdaBoost-A algorithm can build a strong unbiased classifier by combining weak base learners that focus more on the minority class samples at the end of the iterations.

3.2. M-AdaBoost-A-PSO

The variant M-AdaBoost-A-PSO is similar to M-AdaBoost-A-SMV

INPUT: training data $T = (x_i, y_i),\ x_i \in R^n,\quad y_i \in (-1,1)\ ,\ i = 1,2 \ldots \ldots n$

sampling ratio p, decision tree's maximum depth d

initial weight for sample i, $w_1^i = \frac{1}{n}$

for $j = 1,2, \ldots \ldots, J$, where J denotes the total number of weak learners

sample data S_j from training data T with replacement using sampling ratio p based on weight distribution w_j and ensure at least one instance from each class present

train weak learner h_j as a decision tree with maximum depth d on the sampled data S_j

calculate the scaled classification error rate e_j on the training data T

$$e_j = 2 \times \left(1 - AUC_j\right) \times \sum_{i=1}^{n} w_j^i \times I(h_j(x_i) \neq y_i)$$

calculate the coefficient of weak learner as α_j

$$\alpha_j = 0.5 \times log \frac{1 - e_j}{e_j}$$

update the weight of each sample in the training data T

$$w_{j+1}^i = \frac{w_j^i}{Z_j} \times \exp\left(-\alpha_j \times h_j(x_i) \times y_i\right)$$

$$Z_j = \sum_{i=1}^{n} w_j^i \exp\left(-\alpha_j \times h_j(x_i) \times y_i\right)$$

end for

OUTPUT: combine the weak learners to build the binary classifier

$$H(x) = sign\left(\sum_{j=1}^{J} \alpha_j \times h_j(x)\right)$$

Fig. 2. Pseudocode for M-AdaBoost-A algorithm .
adapted from Li et al. (2019)

except that M-AdaBoost-A-PSO utilizes the PSO algorithm to fuse the ensemble. Fig. 3 depicts the M-AdaBoost-A-PSO system architecture. After the multiclass predictions from each expert learner are obtained, the PSO algorithm is utilized to weigh the prediction outcomes of each expert learner and combine them into an ensemble. The weights are optimized based on the expert learners' performance for classifying samples in the training dataset; subsequently, the final weight solutions are applied to the test dataset. The multiclass predictions y_{il} for sample i from expert learner l are fed into the PSO optimization engine, and weight u_l is assigned to expert learner l by the PSO algorithm. Assume that K classes exist; therefore, the final prediction y_i from the ensemble for the *ith* sample can be defined as shown in Eq. (1):

$$y_i = argmax_k \sum_{l=1}^{10} u_l || y_{il} = k || \tag{1}$$

$$where || y_{il} = k || = \begin{cases} 1 & if\ \ y_{il} = k \\ 0 & if\ \ y_{il} \neq k \end{cases}$$

To optimize the weights, the fitness or objective function f for the PSO algorithm is as defined in Eq. (2):

$$f(u) = Ratio_{TPR/FPR} = \frac{\sum_{k=1}^{K} TPR_k / K}{\sum_{k=1}^{K} FPR_k / K} \tag{2}$$

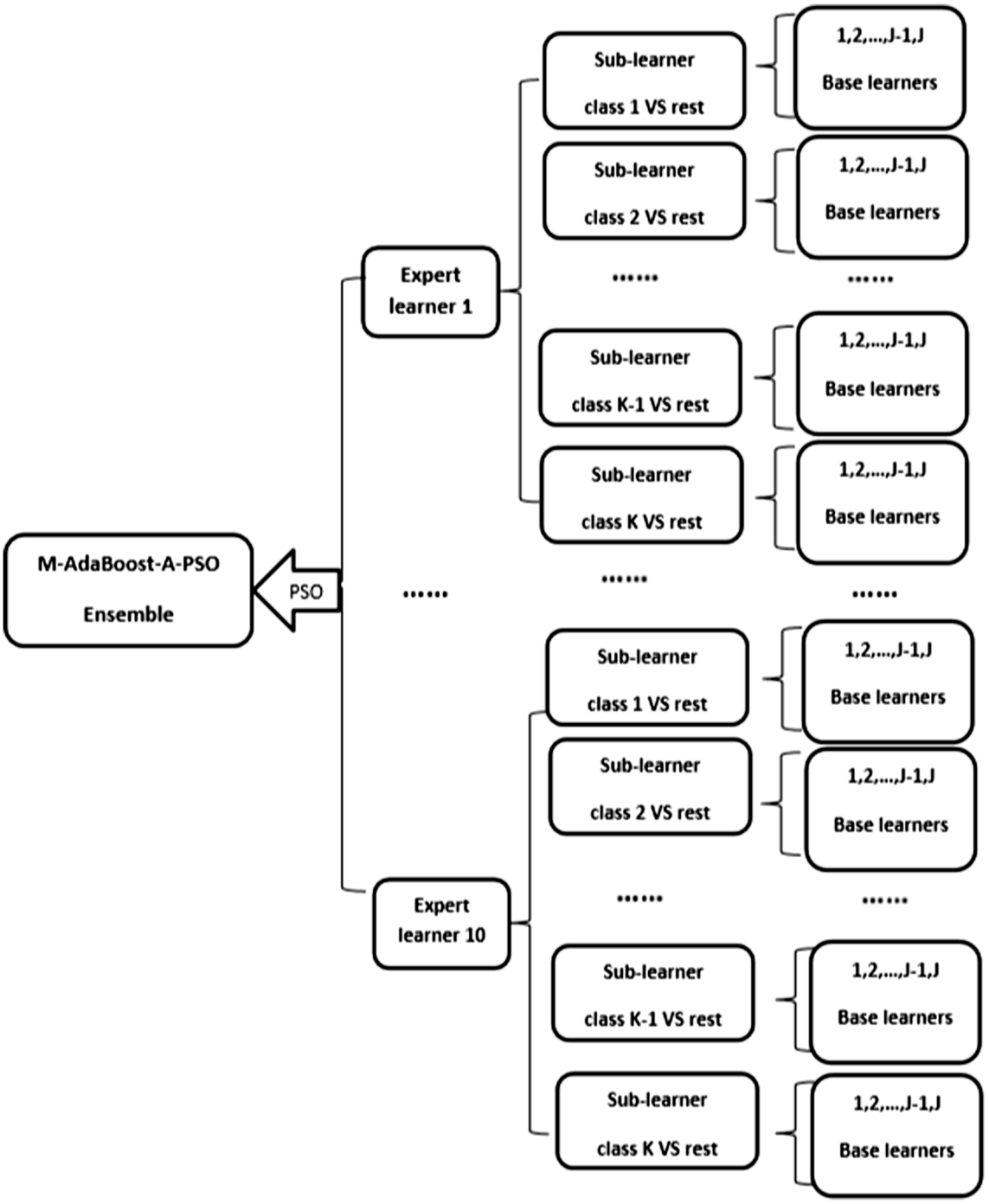


Fig. 3. M-AdaBoost-A-PSO ensemble framework.

where $u = (u_1, u_2, \cdots, u_{10})$ denotes the weight vector for the 10 expert learners, TPR_k denotes the TPR for a given class k, FPR_k denotes the FPR for a given class k, and $Ratio_{TPR/FPR}$ denotes the ratio between the average TPR and the average FPR over all the classes.

Other metrics such as accuracy, precision, and recall are less suitable to be chosen as the objective function. For example, it has been noted that accuracy may not adequately measure the network intrusion detection performance. Fossaceca et al. (2015) pointed out that accuracy rates in the 98% range for very large multi-class imbalanced datasets are not necessarily an indication of good detection performance as the misleading high accuracy may be achieved by detecting nearly all normal traffic but failing to detect most of the minority intrusive traffic. Precision or recall measures only one aspect of the classification performance. The F1-measure combines both precision and recall which can be utilized as the objective function. However, for better network intrusion detection, researchers often aim to increase the TPR and decrease the FPR simultaneously (Mokarian, Faraahi, & Delavar, 2013). We considered that $Ratio_{TPR/FPR}$ is more intuitive than F1 measure in our application. The primary objective of our study is to maximize TPRs and minimize FPRs across all network traffic classes. Hence, we chose to maximize the objective function of $Ratio_{TPR/FPR}$ in the PSO algorithm.

Next, the mechanism regarding how the PSO algorithm optimizes the weight vector $(u_1, u_2, \cdots, u_{10})$ is described in detail. Numerous variants of the PSO algorithm have been developed over time. Multiple studies have employed the basic PSO algorithm to combine classifiers into ensembles. (Aburomman & Ibne Reaz, 2016; Kausar et al., 2010; Veeramachaneni

et al., 2007; Yang et al., 2009). The basic PSO is sensitive to the rotation of the search space and its performance may become deteriorated (Takahama & Sakai, 2018). In our study, the standard particle swarm optimization 2011 (SPSO2011) algorithm was utilized. The SPSO2011 is a significant improvement of the basic PSO by introducing rotational invariance using a hypersphere (Clerc, 2012). Suppose that a swarm comprising a total of M particles is searching for the optimal position (i.e., weight) in a 10-dimensional space, the position vector and the velocity vector of the mth particle are denoted as $u_m = (u_{m1}, u_{m2}, \ldots, u_{m10})$ and $v_m = (v_{m1}, v_{m2}, \ldots, v_{m10})$, respectively. The 10 elements in the position vector represent a potential set of weights for combining the expert learners. Through each iteration, the SPSO2011 algorithm updates each particle's velocity vector, position vector, and fitness value. The position vector with the maximal fitness value is taken as the optimal weight solution to fuse the ensemble.

More formally, the velocity and position of the mth particle at the $(t+1)$th iteration are updated by using Eqs. (3) and (4), respectively:

$$v_m^{t+1} = \omega v_m^t + h_m^t u_m^t \tag{3}$$

$$u_m^{t+1} = u_m^t + v_m^{t+1} \tag{4}$$

where ω is the inertia weight to avoid swarm explosion, and h_m^t is a random point in the hypersphere with center C_m^t and radius $||C_m^t - u_m^t||$. The random point h_m^t works as a disturbance term to the velocity to ensure the particle can keep exploring the search space, so that the stagnation issue can be avoided (Zhu, Tang, & Yuan, 2017). The center of the hypersphere C_m^t is defined based on current position u_m^t, and two points P_m^t and G_m^t in Eq. (5):

$$C_m^t = \frac{u_m^t + P_m^t + G_m^t}{3} \tag{5}$$

The two points P_m^t and G_m^t are further defined by Eqs. (6) and (7), respectively:

$$P_m^t = u_m^t + c_1 r_1^t (p_m^t - u_m^t) \tag{6}$$

$$G_m^t = u_m^t + c_2 r_2^t (g^t - u_m^t) \tag{7}$$

where p_m^t denotes the best position discovered by mth particle; g^t denotes the best position discovered by all the particles in the swarm; r_1 and r_2 are random vectors uniformly distributed in the interval $[0,1]$; c_1 and c_2 are the particle acceleration coefficient and swarm acceleration coefficient, respectively. The pseudocode for fusing our M-AdaBoost-A-PSO ensemble by the SPSO2011 algorithm is outlined in Fig. 4.

As shown in the pseudocode, a number of PSO parameters need to be defined in the implementation. In Aburomman and Ibne Reaz (2016), the PSO algorithm was applied to search for the optimal ensemble weights to improve network intrusion detection. In their implementation, the local uni-modal sampling (LUS) technique was applied to fine tune the PSO parameters. The LUS algorithm did not generate ensemble weights directly, but instead served as a *meta*-optimizer to find optimal parameter settings for PSO. However, their study concluded that the performance improvement gained by utilizing LUS was not significant enough to justify the additional computational cost. The choices of PSO parameters are problem-dependent and need to account for both the performance improvement and the computational expense. The utilization of a *meta*-optimizer for an automatic parameter tuning would make our system overly complicated and less scalable. In our study, the total number of particles and the stagnation threshold were determined by trial-and-error while other parameters remained to the recommended settings in Zambrano-Bigiarini and Rojas (2013). More specifically, the total number of particles in the swarm was set to 30. Empirical studies have shown that the PSO algorithm has the ability to find optimal solutions with small swarm sizes of 10 to 30 particles (Engelbrecht, 2007). The stagnation threshold 10^{-4} was selected to decide whether the

Input: M-AdaBoost-A-EL expert learners' predictions
swarm size, number of iterations, error tolerance,
stagnation threshold, inertia weight, acceleration coefficients
for particle $m = 1, 2, \ldots, M$ do
initialize position u_m^0 and velocity v_m^0
initialize particle's best position p_m^0 and group's best position g^0
for iteration $t = 1,2, \ldots, T$ or the error tolerance is not met do
for particle $m = 1, 2, \ldots, M$ do
update velocity v_m^t and position u_m^t based on equations (3)-(7)
use position component u_m^t as weights to combine expert learners based on equation (1)
calculate fitness value f based on equation (2)
if $f(u_m^t) > f(p_m^t)$, then update particle's best position by assigning u_m^t to p_m^{t+1}
if $f(p_m^t) > f(g^t)$, then update group's best position by assigning p_m^t to g^{t+1}
Output the best position and the final predictions of M-AdaBoost-A-PSO ensemble

Fig. 4. Pseudocode of ensemble fusion by SPSO2011 algorithm.

swarm must be regrouped to avoid premature convergence. The algorithm stops when the error is below the tolerance 10^{-8} or if the maximum number of iterations reaches 50.

4. Experiments and results

4.1. Experiments

We implemented the M-AdaBoost-A-based ensemble in R by utilizing package "rpart" (Therneau & Atkinson, 2019) for building decision tree base learners and package "HydroPSO" (Zambrano-Bigiarini & Rojas, 2013) for weight optimization to fuse the ensemble. Our proposed system was applied to both the AWID and NSL-KDD datasets. We preprocessed our data and conducted extensive experiments using a wide range of parameter settings, which are detailed in the next two sections.

4.1.1. Experiments with AWID 802.11 data

In this study, the AWID-CLS-R-Train dataset was selected to perform our experiments. The AWID-CLS-R-Train dataset contained 154 features and four classes including the normal class and multiple attack classes, i.e., flooding, impersonation, and injection. The data distribution and the binary class imbalance ratios for the AWID-CLS-R-Train dataset are summarized in Tables 3 and 4.

A total of 1,795,575 samples existed in the AWID-CLS-R-Train dataset, which was much larger than those in the NSL-KDD dataset. Owing to the extremely large data size, the M-AdaBoost-A algorithm without PSO optimization was utilized to classify the AWID dataset. The PSO optimization was used to fuse an ensemble initially; however, the computation of the PSO algorithm was extremely slow for the large AWID dataset. Because the result of the expert learners based on M-AdaBoost-A for the AWID dataset was highly accurate, further performance improvement by fusing the ensemble with the PSO algorithm would be negligible and unnecessary considering the significant add-on computational cost. Instead, an ensemble M-AdaBoost-A-SMV was formed by combining expert learners based on the SMV strategy, which is computationally inexpensive.

As our framework does not require feature selection, the following data preprocessing was performed to clean up the AWID dataset. As the data contained almost all possible 802.11 fields, redundant features with no variations among the four classes were eliminated as they could not differentiate the classes. Missing values represented by empty fields or question marks were transformed to "NA" values. Meanwhile, hexadecimal number fields were converted to numeric integers after colons were removed. The AWID-CLS-R-Train dataset was randomly split into training and test sets with a split ratio of 50% versus 50%. We performed this random data split with the 50%/50% ratio five times on the full data and created five sets of training/test data. Hence, for a given random data split, both the created training set and the created test set are 50% of the full data without overlapping with each other. Our proposed algorithm M-AdaBoost-A-SMV was performed on these five randomly split training/test sets and for each run, the algorithm was trained on the training set created and tested on the test set created. As a result, there were a total of five M-AdaBoost-A-SMV ensembles trained and tested. Within each M-AdaBoost-A-SMV ensemble, 10 M-AdaBoost-A-EL expert learners were trained and tested. As five random data splits were performed, in order to avoid any bias due to one random data split, the average results were taken over the five test sets for M-AdaBoost-A-SMV and M-AdaBoost-A-EL, respectively. Our proposed system will be benchmarked against the existing standard boosting techniques. To guarantee valid comparison results, each variant of the standard boosting algorithm, including SAMME and AdaBoost.M1, was performed on the same five training/test sets with 10 runs for each training/test set. The average results were also taken over the five test sets for SAMME and AdaBoost.M1, respectively.

Table 3
Data distribution for AWID-CLS-R-Train.

Classes	Samples	Percentage
Flooding	48,484	2.70%
Impersonation	48,522	2.70%
Injection	65,379	3.64%
Normal	1,633,190	90.96%

Table 4
Binary class imbalance ratios for AWID-CLS-R-Train.

Classes	Ratio
Normal vs. Rest	10.1:1
Rest vs. Flooding	36.0:1
Rest vs. Impersonation	36.0:1
Rest vs. Injection	26.5:1

During the training, a number of hyperparameters were specified for M-AdaBoost-A expert learners to classify the AWID dataset. We tested a wide range of parameter values on the training set in our experiments. For example, Fig. 5 plots the maximum depth of the decision trees versus the macro F1 on the training set when the sampling ratio and the number of boosting iterations were fixed at 0.1% and 20, respectively. It is obvious that the one level decision stumps as base learners did not provide the ideal performance. As mentioned previously, the maximum depth of the decision trees was set to five as further performance improvement was insignificant when the depth was greater than five. The boosting sampling ratio and the number of boosting trials were selected empirically as well. Fig. 6 plots the sampling ratio versus the macro F1 on the training set when the maximum depth of the decision trees and the number of boosting iterations were fixed at 5 and 20, respectively. In general, a greater boosting sampling ratio generated a better classification performance. However, the classification performance tended to saturate once the ratio reached a certain number. Overall, the relationship between the sampling ratio and the model performance was not always monotonically increasing. Moreover, the greater the boosting sampling ratio, the higher was the computational cost. Therefore, the final sampling ratio was determined to be 0.1% by balancing the tradeoff between the model performance and the computational cost. Similar observations regarding the number of boosting trials have been experimentally. As shown in Fig. 7 5 to 50 boosting trials were tested, and the value of 20 was selected for the

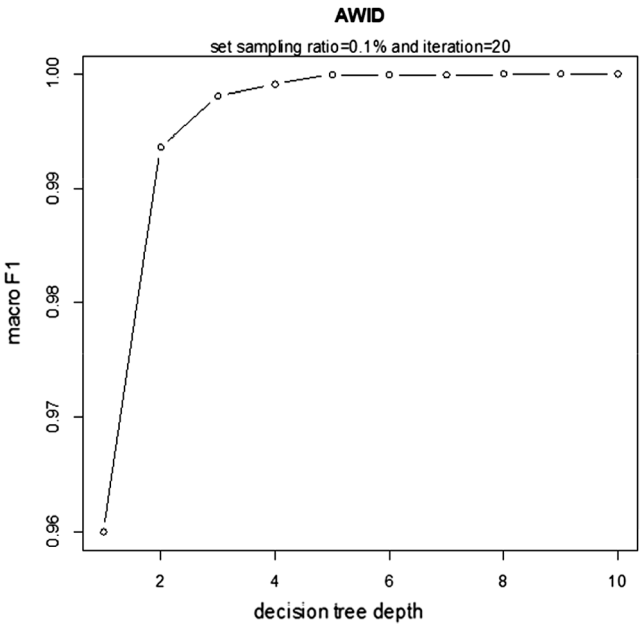


Fig. 5. Plot of decision tree depth vs macro F1 for AWID dataset.

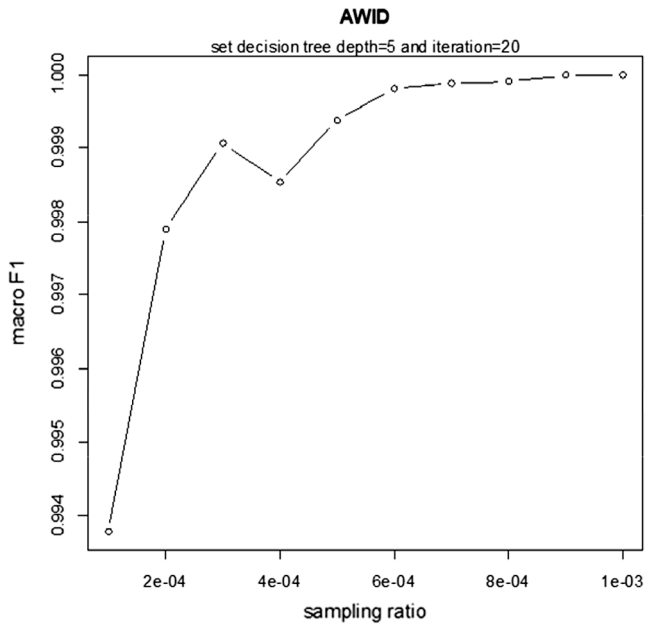


Fig. 6. Plot of sampling ratio vs macro F1 for AWID dataset.

training. To ensure a fair benchmark comparison, for SAMME and AdaBoost.M1, experiments with bootstrap resampling, decision trees with maximum depth of five, and 20 boosting iterations were performed.

4.1.2. Experiments with NSL-KDD dataset

Similar to other studies, we selected a subset of the NSL-KDD dataset, called KDDTrain+ to perform our experiments. The KDDTrain+ dataset contained 125,973 samples, 41 features, and 5 classes, including the normal class and various attack classes, i.e., DoS, probe, R2L, and U2R. The data distribution and the binary class imbalance ratios for the KDDTrain+ dataset are summarized in Tables 5 and 6.

Before training the classifiers, the data preprocessing was focused on one-hot encoding to create dummy indicators for categorical variables. In our study, continuous variables were utilized in their original scales

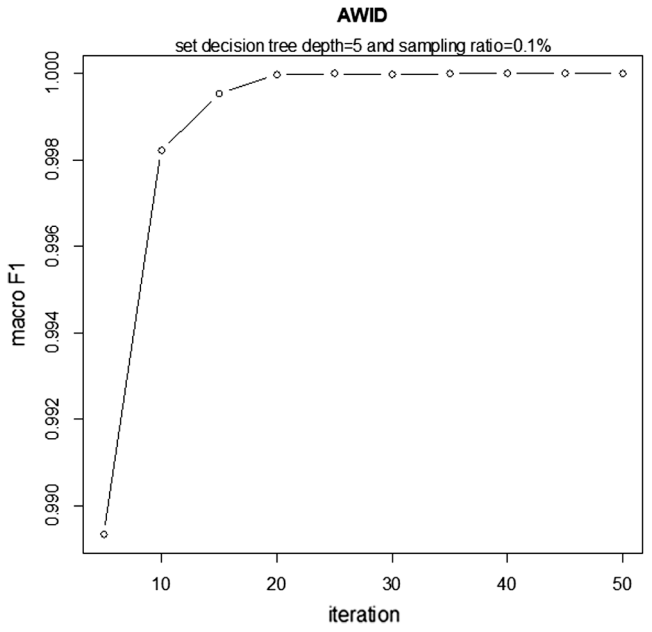


Fig. 7. Plot of iteration vs macro F1 for AWID dataset.

without being standardized. To perform our experiments, the KDDTrain+ dataset was randomly split into training and test sets with the split ratio of 50% versus 50%. We performed the random data split with the 50%/50% ratio five times on the full data and created five sets of training/test data. Hence, for a given random data split, both the created training set and the created test set are 50% of the full data without overlapping with each other. Our proposed algorithms, M-AdaBoost-A-PSO and M-AdaBoost-A-SMV, were performed on these five randomly split training/test sets and for each run, the classification algorithms including the PSO optimization were trained on the training set created and evaluated on the test set created. As a result, there were a total of five M-AdaBoost-A-SMV ensembles and five M-AdaBoost-A-PSO ensembles trained and tested. Within each M-AdaBoost-A-based ensemble, 10 M-AdaBoost-A-EL expert learners were trained and tested. To avoid any bias due to one random data split, the average results were taken over the five test sets for M-AdaBoost-A-SMV, M-AdaBoost-A-PSO, and M-AdaBoost-A-EL, respectively. Our proposed system will be benchmarked against existing standard boosting techniques. To guarantee valid comparison results, each variant of the standard boosting algorithm, including SAMME and AdaBoost.M1, was performed on the same five training/test sets with 10 runs for each training/test set. The average results were also taken over the five test sets for SAMME and AdaBoost.M1, respectively.

A number of hyperparameters were set empirically in the M-AdaBoost-A expert learners for classifying the NSL-KDD dataset. More specifically, the maximum depth of the decision trees was set to five, the boosting sampling ratio was selected as 10%, and the number of boosting trials was set to 50. The selections of these hyperparameters for the algorithm were determined based on considering both efficiency and performance by conducting experiments with a range of parameter values. Figs. 8–10 plot a given hyperparameter versus the macro F1 on the training set when other hyperparameters were fixed at certain values. Similar observations were obtained for the NSL-KDD dataset from these plots compared to the ones for AWID dataset. It is noted again that one level decision stumps as base learners did not perform well and deeper decision trees were required as base learners. As shown in Fig. 9, since the classification performance was more varying with the values of sampling ratio, a relatively greater sampling ratio was preferred to maintain stable performance and better tackle the severe class imbalance in the NSL-KDD dataset. To ensure a fair benchmark comparison, for SAMME and AdaBoost.M1, experiments with bootstrap resampling, decision trees of maximum depth five, and 50 boosting iterations were performed.

4.2. Result evaluation

Compared to the existing standard boosting techniques, our proposed approach aims to achieve performance improvement especially for recognizing the minority class without sacrificing much of the performance on the majority class. Our proposed system was evaluated using a number of metrics. Accuracy is the most typical metric used for assessing the overall effectiveness of a classifier. However, in the network intrusion detection domain, the data are generally dominated by the normal nonintrusive network traffic, while intrusions only represent a small fraction of the overall traffic; a biased classifier that favors the nonintrusive majority class and ignores the intrusive minority

Table 5
KDDTrain + data distribution.

Classes	Samples	Percentage
DoS	45,927	36.46%
Normal	67,343	53.46%
Probe	11,656	9.25%
R2L	995	0.79%
U2R	52	0.04%

Table 6
Binary class imbalance ratios for KDDTrain+.

NSL-KDD dataset Majority vs. minority	Imbalance ratio
Normal vs. Rest	1.1:1
Rest vs. DoS	1.7:1
Rest vs. Probe	9.8:1
Rest vs. R2L	125.6:1
Rest vs. U2R	2421.6:1

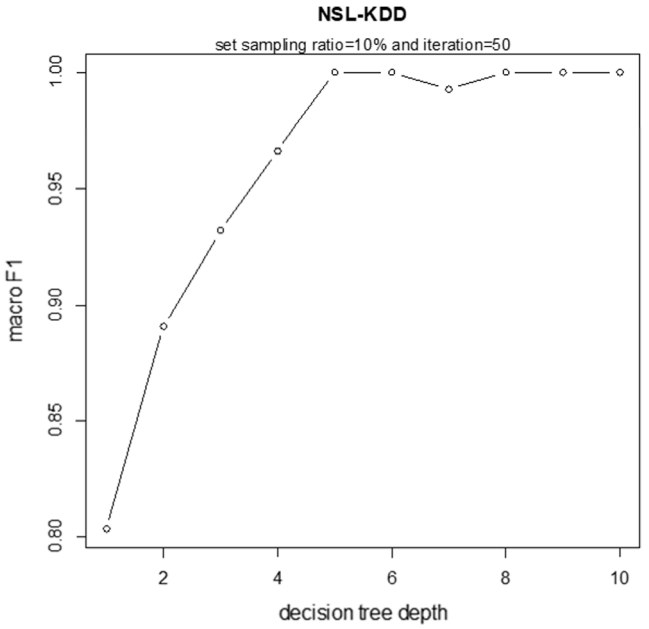


Fig. 8. Plot of decision tree depth vs macro F1 for NSL-KDD dataset.

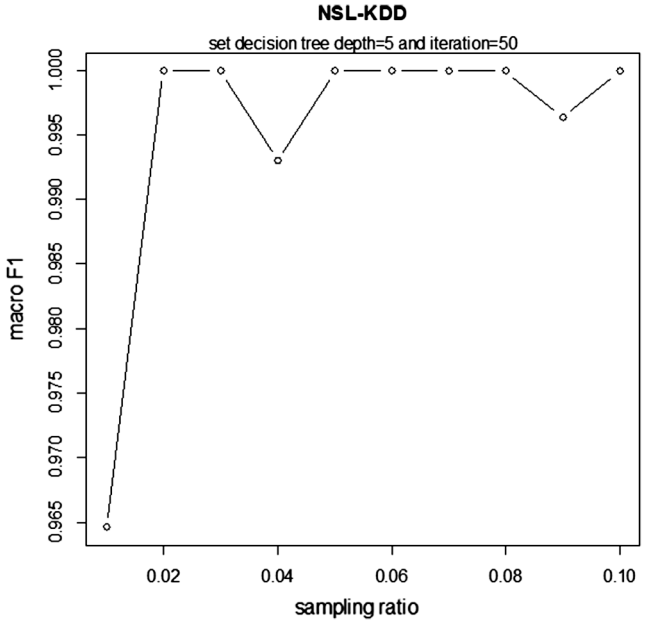


Fig. 9. Plot of sampling ratio vs macro F1 for NSL-KDD dataset.

class may have a high accuracy. It is well known that accuracy alone is insufficient to evaluate the classification performance for network intrusion detection with class imbalance. Hence, in addition to accuracy, metrics such as the TPR, FPR, precision, recall, F1 score, G-mean, and

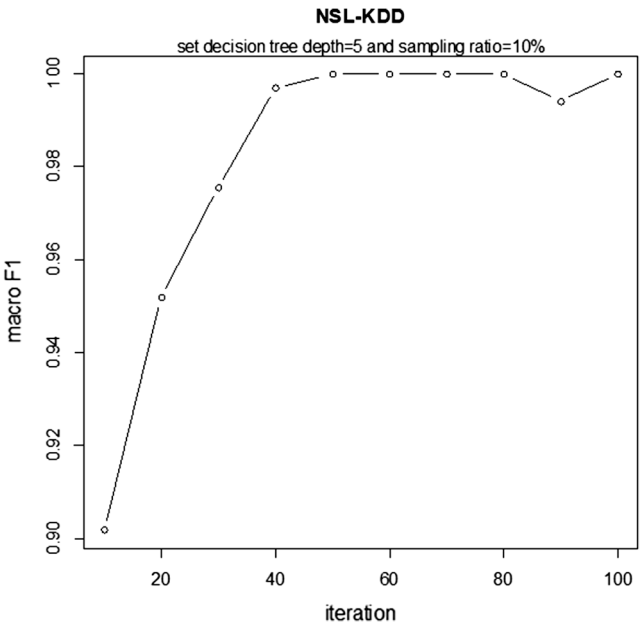


Fig. 10. Plot of iteration vs macro F1 for NSL-KDD dataset.

$Ratio_{TPR/FPR}$ were utilized to comprehensively assess the performance on both the nonintrusive majority and intrusive minority classes in our study.

More formally, the TPR, FPR, precision, recall, F1 score, G-mean, $Ratio_{TPR/FPR}$, and accuracy are defined in the formulas below:

$$TPR(Recall) = \frac{TP}{TP + FN} \tag{8}$$

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{11}$$

$$G - mean = \left(\prod_{k=1}^{K} recall_k\right)^{1/K} \tag{12}$$

$$Ratio_{TPR/FPR} = \frac{\sum_{k=1}^{K} TPR_k/K}{\sum_{k=1}^{K} FPR_k/K} \tag{13}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

The TPR and FRP calculate the proportions of correct and incorrect positive identifications, respectively. Precision measures the proportion of the predicted positive instances that are the actual TPs, and recall measures the proportion of the actual TP instances that are predicted to be positive. The F1-measure incorporates both precision and recall to balance the tradeoff. The per-class precision, per-class recall, and per-class F1 can be calculated for each class first; subsequently, the macro precision, macro recall, and macro F1 can be derived using the average per-class precision, per-class recall, and per-class F1 over all the classes, respectively.

G-mean is defined as the geometric mean of recall values of all the classes. If the minority classes have a low recall, then the G-mean value will be reduced. In extreme cases, if one of the minority classes is entirely unrecognized by the classifier, then G-mean will be zero. A high G-mean shows that a classifier can balance the recognition among different classes and the minority classes are not ignored. Hence, G-

mean measures the balanced performance of a learning algorithm among classes and is suitable for datasets involving class imbalance. In addition to these metrics, as mentioned in the previous section, the measurement $Ratio_{TPR/FPR}$, which is defined as the ratio between the average TPR and the average FPR over all the classes, is utilized to evaluate the overall effectiveness of the classification algorithm.

4.2.1. Results for AWID dataset

For the AWID dataset, the average metric measurements based on multiple runs of M-AdaBoost-A-SMV, M-AdaBoost-A-EL, SAMME, and AdaBoost.M1 are summarized in Tables 7 and 8. Table 7 includes the overall performance metrics, such as accuracy, $ratio_{TPR/FPR}$, macro precision, macro recall, macro F1, and G-mean. Table 8 includes the per-class performance metrics, such as the TPR and FPR for each class.

We first focused on the performance comparison between our proposed expert learner M-AdaBoost-A-EL and the standard boosting techniques, including SAMME and AdaBoost.M1. The G-mean distributions from multiple runs were compared among M-AdaBoost-A-EL, SAMME, and AdaBoost.M1 using boxplots. As shown in the boxplots of Fig. 11, the maximum, minimum, median, and mean of G-mean measurements for M-AdaBoost-A-EL were all higher than those of SAMME and AdaBoost.M1. Hence, our expert learner M-AdaBoost-A-EL performed better in balancing the recognition among different classes. More importantly, a higher G-mean implied that the minority classes were not neglected because otherwise, the G-mean value would be reduced. More specifically, compared with AdaBoost.M1, our expert learner M-AdaBoost-A-EL outperformed AdaBoost.M1 in terms of each of the overall performance metrics and the per-class performance metrics. Compared with SAMME, our M-AdaBoost-A-EL performed better in terms of the $ratio_{TPR/FPR}$, macro recall, and G-mean. In terms of per-class performance, compared with SAMME, our M-AdaBoost-A-EL yielded higher TPRs for all three minority classes and lower FPRs for most classes except flooding. More notably, M-AdaBoost-A-EL balanced the TPRs and FPRs across multiple classes and yielded a significantly higher $ratio_{TPR/FPR}$ than those of SAMME and AdaBoost.M1.

Next, we focused on the performance comparison between the expert learner M-AdaBoost-A-EL and the multiboosting ensemble M-AdaBoost-A-SMV. While our expert learner M-AdaBoost-A-EL performed extremely well on the AWID dataset, our proposed multiboosting ensemble M-AdaBoost-A-SMV further improved the accuracy and stability of the classification performance. As observed, our M-AdaBoost-A-SMV outperformed M-AdaBoost-A-EL for each of the overall performance metrics and the per-class performance metrics, which was considered significant. It is noteworthy that both our M-AdaBoost-A-SMV and M-AdaBoost-A-EL achieved zero FPRs for the injection class.

Finally, among all the algorithms compared, our M-AdaBoost-A-SMV achieved the highest accuracy, $ratio_{TPR/FPR}$, macro recall, G-mean, and TRPs for all three minority classes; and the lowest FPRs for classes impersonation, injection, and normal. Our proposed approach M-AdaBoost-A-SMV not only improved the overall classification accuracy, but also improved the performance for recognizing minority classes. Furthermore, SAMME demonstrated the highest macro precision and macro F1. However, SAMME also indicated the highest TPR for the normal and the lowest FPR for the flooding class. Relative to our proposed approach, it is likely that SAMME will not detect class flooding well with bias favoring class normal. In cases when SAMME detected instances belonging to class flooding, they are generally reliable and hence result in a higher macro precision and macro F1. In conclusion, for the AWID dataset, our proposed approach M-AdaBoost-A-SMV performed the best in recognizing minority classes, maximizing TPRs, and minimizing FPRs across multiple classes.

4.2.2. Results for NSL-KDD dataset

For the NSL-KDD dataset, the average metric measurements based on multiple runs of M-AdaBoost-A-PSO, M-AdaBoost-A-SMV, M-AdaBoost-A-EL, SAMME, and AdaBoost.M1 are summarized in Tables 9 and 10. Table 9 includes the overall performance metrics, such as accuracy, $ratio_{TPR/FPR}$, macro precision, macro recall, macro F1, and G-mean. Table 10 includes the per-class performance metrics, such as the TPR and FPR for each class.

We first focused on the performance comparison between our proposed expert learner M-AdaBoost-A-EL and the standard boosting techniques, including SAMME and AdaBoost.M1. The G-mean distributions from multiple runs were compared among M-AdaBoost-A-EL, SAMME, and AdaBoost.M1 using boxplots. As shown in the boxplots of Fig. 12, the maximum, minimum, median, and mean of G-mean measurements for M-AdaBoost-A-EL were all higher than those of SAMME and AdaBoost.M1. Hence, our expert learner M-AdaBoost-A-EL performed better in balancing the recognition among different classes. More importantly, a higher G-mean implied that the minority classes were not neglected because otherwise, the G-mean value would be reduced. More specifically, when compared with AdaBoost.M1, our M-AdaBoost-A-EL performed better in terms of accuracy, $ratio_{TPR/FPR}$, macro recall, macro F1, and G-mean. In terms of per-class performance, compared with AdaBoost.M1, our M-AdaBoost-A-EL yielded higher TPRs across all classes and lower FPRs for most classes except U2R. Compared with SAMME, our M-AdaBoost-A-EL performed better in terms of accuracy, $ratio_{TPR/FPR}$, macro recall, and G-mean. In terms of per-class performance, compared with SAMME, our M-AdaBoost-A-EL yielded higher TPRs across all classes and lower FPRs for most classes except DoS and U2R. More notably, M-AdaBoost-A-EL balanced the TPRs and FPRs across multiple classes and yielded a significantly higher $ratio_{TPR/FPR}$ than those of SAMME and AdaBoost.M1.

Next, we focused on the performance comparison between the expert learner M-AdaBoost-A-EL and the multiboosting ensembles including M-AdaBoost-A-PSO and M-AdaBoost-A-SMV. While our expert learner M-AdaBoost-A-EL performed extremely well on the NSL-KDD dataset, both our proposed multiboosting ensembles M-AdaBoost-A-PSO and M-AdaBoost-A-SMV further improved the accuracy and stability of the classification performance. As observed, our M-AdaBoost-A-SMV performed better than M-AdaBoost-A-EL for each of the overall performance metrics. In terms of per-class performance, compared with M-AdaBoost-A-EL, M-AdaBoost-A-SMV yielded higher TPRs across all the classes and lower FPRs for most classes except normal. Hence, our multiboosting ensemble M-AdaBoost-A-SMV demonstrated the expected performance improvement over the individual expert learners of M-AdaBoost-A-EL. Using the PSO algorithm, M-AdaBoost-A-PSO aims to achieve further improvements for recognizing minority classes, maximizing TPRs, and minimizing FPRs across multiple classes. As noted, M-AdaBoost-A-PSO outperformed M-AdaBoost-A-EL in terms of each of the overall performance metrics and the per-class performance metrics, which was considered significant.

Table 7
Average overall performance on AWID dataset.

	Accuracy	$ratio_{TPR/FPR}$	Macro.Precision	Macro.Recall	Macro.F1	G.mean
M-AdaBoost-A-SMV	**0.9999722**	**48,935**	0.9997949	**0.9999257**	0.9998603	**0.9999257**
M-AdaBoost-A-EL	0.9999375	33,322	0.9994860	0.9998837	0.9996848	0.9998837
SAMME	0.9999721	23,157	**0.9998933**	0.9998421	**0.9998677**	0.9998421
AdaBoost-A.M1	0.9998767	9833	0.9991896	0.9996857	0.9994375	0.9996857

The best result obtained for each metric among the algorithms is highlighted in bold.

Table 8
Average per-class performance on AWID dataset.

		Flooding	Impersonation	Injection	Normal
M-AdaBoost-A-SMV	TPR	**0.9999918**	**0.9997455**	**0.9999878**	0.9999777
	FPR	0.0000201	**0.0000025**	**0.0000000**	**0.0000639**
M-AdaBoost-A-EL	TPR	0.9999679	0.9996405	0.9999829	0.9999437
	FPR	0.0000534	0.0000037	0.0000000	0.0000762
SAMME	TPR	0.9997993	0.9995994	0.9999817	**0.9999880**
	FPR	**0.0000070**	0.0000040	0.0000006	0.0001841
AdaBoost-A.M1	TPR	0.9995311	0.9994211	0.9998906	0.9999000
	FPR	0.0000680	0.0000093	0.0000163	0.0003565

The best result obtained for each metric among the algorithms is highlighted in bold.

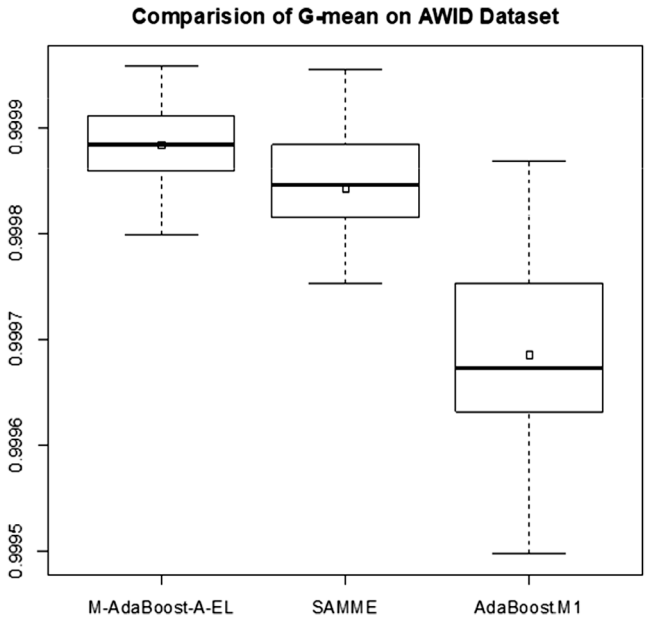


Fig. 11. Boxplots of G-mean measurements for AWID data.

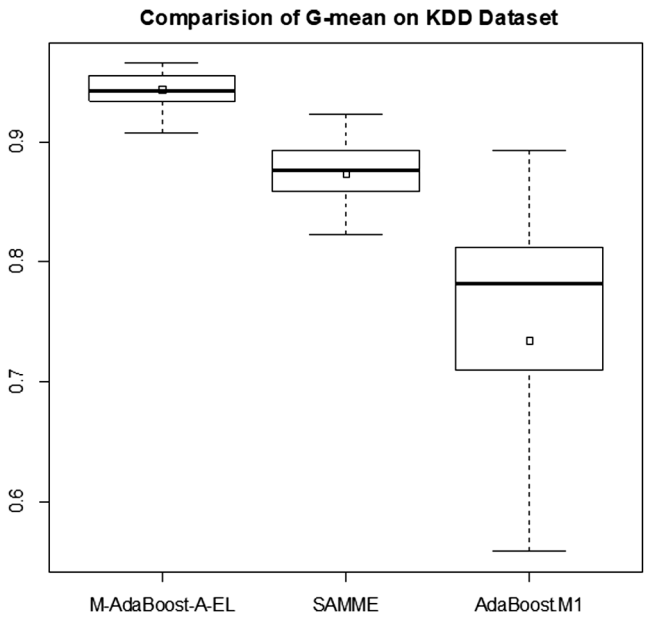


Fig. 12. Boxplots of G-mean measurements for NSL-KDD data.

Table 9
Average overall performance on NSL-KDD dataset.

	Accuracy	$ratio_{TPR/FPR}$	Macro.Precision	Macro.Recall	Macro.F1	G.mean
M-AdaBoost-A-PSO	0.9989617	**3123**	0.8804321	**0.9556466**	0.9164817	**0.9519959**
M-AdaBoost-A-SMV	**0.9989744**	3000	0.8895698	0.9513090	**0.9193693**	0.9469366
M-AdaBoost-A-EL	0.9984829	2271	0.8516219	0.9480481	0.8972157	0.9432146
SAMME	0.9967603	815	0.9389187	0.8927059	0.9149879	0.8741181
AdaBoost.M1	0.9952543	503	**0.9484873**	0.8188080	0.8777104	0.7341172

The best result obtained for each metric among the algorithms is highlighted in bold.

Table 10
Average per-class performance on NSL-KDD dataset.

		DoS	Normal	Probe	R2L	U2R
M-AdaBoost-A-PSO	TPR	0.9996863	0.9991743	**0.9977438**	**0.9734142**	**0.8082146**
	FPR	**0.0001199**	**0.0008460**	**0.0001225**	0.0000352	0.0004225
M-AdaBoost-A-SMV	TPR	**0.9997123**	**0.9992159**	0.9976724	0.9725894	0.7873549
	FPR	0.0001249	0.0009689	0.0001331	**0.0000288**	0.0003462
M-AdaBoost-A-EL	TPR	0.9995712	0.9985835	0.9968351	0.9700799	0.7751710
	FPR	0.0001503	0.0009567	0.0001562	0.0000467	0.0007887
SAMME	TPR	0.9994571	0.9978744	0.9900137	0.8920087	0.5841754
	FPR	0.0001389	0.0044150	0.0006771	0.0004128	0.0000734
AdaBoost.M1	TPR	0.9992153	0.9983700	0.9808348	0.7954674	0.3201526
	FPR	0.0003246	0.0078444	0.0008010	0.0001232	**0.0000416**

The best result obtained for each metric among the algorithms is highlighted in bold.

Finally, among all the algorithms compared, M-AdaBoost-A-SMV achieved the highest accuracy, macro F1, and TPRs for classes DoS and normal; and the lowest FPRs for class R2L. Moreover, among all the algorithms compared, M-AdaBoost-A-PSO achieved the highest $ratio_{TPR/FPR}$, macro recall, G-mean, TPRs for classes probe, R2L, and U2R; and the lowest FPRs for classes DoS, normal, and probe. It is interesting to note that AdaBoost.M1 had the lowest FPR for class U2R. However, AdaBoost.M1 also had the lowest TPR for class U2R. This indicates that AdaBoost.M1 ignored class U2R and seldom generated positive predictions of U2R, resulting in both the lowest TPR and FPR for class U2R. In conclusion, for the NSL-KDD dataset, our proposed approach M-AdaBoost-A-PSO performed the best in recognizing minority classes, maximizing TPRs, and minimizing FPRs across multiple classes.

While our proposed ensemble system achieved nearly perfect results across all classes for the AWID dataset, the results of the NSL-KDD dataset appeared less ideal compared with those of the AWID dataset. The NSL-KDD dataset is considered more challenging owing to the more severe class imbalance caused by class U2R. Unlike the relatively new AWID dataset, the NSL-KDD dataset has been studied extensively. To justify the significance of our results for the NSL-KDD dataset, our proposed system was further compared with other approaches reported in the literature. Table 11 collects study results produced by various researchers on the NSL-KDD dataset and ranks them in terms of the overall effectiveness measured based on $ratio_{TPR/FPR}$. With a significantly higher $ratio_{TPR/FPR}$, our proposed ensembles M-AdaBoost-A-PSO and M-AdaBoost-A-SMV were superior to methods in previous studies. In particular, for the minority classes, our system performed remarkably well with both high TPRs and low FPRs achieved simultaneously.

The experiments conducted on both AWID and NSL-KDD datasets demonstrated that M-AdaBoost-A-PSO and M-AdaBoost-A-SMV are distinguished by model complexity, classification performance and computational cost. Table 12 lists the average training time of our experiments over five data splits on a computer with a 2.6 GHz processor and 16 GB of RAM. The performance improvement gained by M-AdaBoost-A-PSO over M-AdaBoost-A-SMV is achieved with additional computational expense. M-AdaBoost-A-PSO may serve as a preferred solution for learning moderately large datasets, whereas M-AdaBoost-A-SMV may serve as a preferred solution for learning extremely large datasets.

5. Discussion

We herein introduced a novel machine learning system based on M-AdaBoost-A to detect network intrusions more effectively in both traditional enterprise and AWID wireless networks. As described, our system did not require data preprocessing, such as feature selection or variable scaling; therefore, data information loss was avoided. To our best knowledge, this is the first study utilizing M-AdaBoost-A to mitigate class imbalance in network intrusion detection. Unlike many other network intrusion detection studies, which failed to provide specific details regarding the model performance per class, our proposed system was comprehensively evaluated based on various metrics across multiple classes. Compared with the standard techniques, our approach achieved excellent performance consistently across multiple classes in two distinct network intrusion detection domains. More importantly, our proposed system outperformed other systems in recognizing minority classes, maximizing TPRs, and minimizing FPRs across multiple classes. Furthermore, through literature review, many studies have combined classifiers in an ad-hoc manner or selected a single classifier; in our study, we employed various strategies, including combining classifiers algorithmically through optimization. Our proposed ensemble system performed better than individual classifiers. In conclusion, our proposed system is a robust framework for learning large-scale imbalanced multiclass data for network intrusion detection. For future studies, we will focus on comprehensively evaluating our approach against other specialized boosting techniques, including SMOTEBoost and various cost-sensitive boosting for mitigating class imbalance in the network intrusion detection domain. We also plan to extend our proposed system to other cyber security domains, such as malware detection. Additionally, numerous variants of the PSO algorithm can be compared for fusing the ensemble. The impact of the parameter choices for the PSO algorithm on fusing the ensemble can be studied as well.

Table 12
Average training time.

Dataset	Algorithm	Training time (minutes)
NSL-KDD	M-AdaBoost-A-SMV	27.29
NSL-KDD	M-AdaBoost-A-PSO	37.23
AWID	M-AdaBoost-A-SMV	51.68

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Table 11
Comparison with other studies.

		DoS	Normal	Probe	R2L	U2R	$ratio_{TPR/FPR}$
M-AdaBoost-A-PSO	TPR	0.9996863	0.9991743	0.9977438	0.9734142	0.8082146	**3123**
(proposed)	FPR	0.0001199	0.0008460	0.0001225	0.0000352	0.0004225	
M-AdaBoost-A-SMV	TPR	0.9997123	0.9992159	0.9976724	0.9725894	0.7873549	3000
(proposed)	FPR	0.0001249	0.0009689	0.0001331	0.0000288	0.0003462	
WTPR-SEMI-P	TPR	0.9996000	0.9987000	0.9959000	0.9598000	0.6000000	1687
(Mikhail et al., 2019)	FPR	0.0003000	0.0015000	0.0003000	0.0001000	0.0005000	
MARK-ELM	TPR	0.9996000	0.9989000	0.9742000	0.9494000	0.6287000	1517
(Fossaceca et al., 2015)	FPR	0.0002000	0.0019000	0.0003000	0.0005000	0.0001000	
OVA-SVM	TPR	0.9990000	0.9960000	0.9920000	0.9870000	0.7390000	673
(Thaseen and Kumar, 2017)	FPR	0.0010000	0.0040000	0.0010000	0.0010000	0.0000000	
EID3	TPR	0.9990000	0.9990000	0.9980000	0.9997000	0.9980000	594
(Jaiganesh et al., 2014)	FPR	0.0003000	0.0008000	0.0039000	0.0022000	0.0012000	
ID3	TPR	0.9990000	0.9990000	0.9970000	0.9350000	0.4910000	244
(Jaiganesh et al., 2014)	FPR	0.0003000	0.0010000	0.0055000	0.0098000	0.0015000	
NB/DT	TPR	0.9976000	0.9984000	0.9975000	0.9935000	0.9947000	75
(Farid et al., 2010)	FPR	0.0003000	0.0005000	0.0028000	0.0622000	0.0010000	
ANN	TPR	0.9380000	0.9880000	0.8980000	0.9190000	0.8660000	65
(AkashdeepI & Kumar, 2017)	FPR	0.0004000	0.0655000	0.0014000	0.0028000	0.0005000	
Hybrid Artificial Bee	TPR	0.9991000	0.9899000	0.9989000	0.9937000	0.9989000	58
(Mazini et al., 2019)	FPR	0.0010000	0.0150000	0.0150000	0.0340000	0.0210000	

Note: The results of our proposed approaches were calculated as the average results over five test sets.

CRediT authorship contribution statement

Ying Zhou: Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing. **Thomas A. Mazzuchi:** Conceptualization, Methodology, Investigation, Writing - review & editing, Supervision. **Shahram Sarkani:** Conceptualization, Methodology, Investigation, Writing - review & editing, Supervision.

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.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments which were very helpful in improving the paper.

References

Abadeh, M. S., Mohamadi, H., & Habibi, J. (2011). Design and analysis of genetic fuzzy systems for intrusion detection in computer networks. *Expert Systems with Applications, 38*(6), 7067–7075. https://doi.org/10.1016/j.eswa.2010.12.006.

Aburomman, A. A., & Ibne Reaz, M. B. (2016). A novel SVM-kNN-PSO ensemble method for intrusion detection system. *Applied Soft Computing Journal, 38*, 360–372. https://doi.org/10.1016/j.asoc.2015.10.011.

AkashdeepI, Manzoor I., & Kumar, N. (2017). A feature reduced intrusion detection system using ANN classifier. *Expert Systems with Applications, 88*, 249–257. https://doi.org/10.1016/j.eswa.2017.07.005.

Alotaibi, B., & Elleithy, K. (2016). A majority voting technique for Wireless Intrusion Detection Systems. 2016 IEEE Long Island Systems, Applications and Technology Conference, LISAT 2016, April. https://doi.org/10.1109/LISAT.2016.7494133.

Aminanto, M. E., & Kim, K. (2017). Detecting impersonation attack in wifi networks using deep learning approach. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10144 LNCS, 136–147. https://doi.org/10.1007/978-3-319-56549-1_12.

AWID Data Set for Network-Based Intrusion Detection Systems. (accessed August, 2019) <icsdweb.aegean.gr/awid/>, March 2016.

Beheshti, Z., & Shamsudding, S. M. (2013). A review of population-based meta-heuristic algorithms. *International Journal of Advances in Soft Computing and its Applications, 5*(1), 1–35.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science), 2838*, 107–119. https://doi.org/10.1007/978-3-540-39804-2_12.

Chen, Z., Qian, P., & Chen, Z. (2009). Application of PSO-RBF neural network in network intrusion detection. In *Proceedings of the 3rd international symposium on intelligent information technology application* (pp. 362–364).

Clerc, M. (2012). Standard particle swarm optimisation. HAL open access archive.

Cortes, C., Mohri, M., & Syed, U. (2014). Deep boosting. In *Proceedings of the 31st international conference on machine learning* (pp. 1179–1187).

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction*. Wiley.

Fan, W., Stolfo, S., Zhang, J., & Chan, P. (1999). Adacost: Misclassification cost-sensitive boosting. In *Proceedings 16th international conference on machine learning* (pp. 97–105).

Farid, D. M., Harbi, N., & Rahman, M. Z. (2010). Combining naive bayes and decision tree for adaptive intrusion detection. *International Journal of Network Security & Its Applications, 2*(2), 12–25. https://doi.org/10.5121/ijnsa.2010.2202.

Farnaaz, N., & Jabbar, M. A. (2016). Random forest modeling for network intrusion detection system. *Procedia Computer Science, 89*, 213–217. https://doi.org/10.1016/j.procs.2016.06.047.

Fernández-Baldera, A., Buenaposada, J. M., & Baumela, L. (2018). BAdaCost: Multiclass boosting with costs. *Pattern Recognition, 79*, 467–479. https://doi.org/10.1016/j.patcog.2018.02.022.

Fossaceca, J. M., Mazzuchi, T. A., & Sarkani, S. (2015). MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection. *Expert Systems with Applications, 42*(8), 4062–4080. https://doi.org/10.1016/j.eswa.2014.12.040.

Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the 13th international conference on machine learning* (pp. 148–156).

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis, 38*(4), 367–378.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C, 42*(4), 463–484.

Hu, W., Hu, W., & Maybank, S. (2008). AdaBoost-based algorithm for network intrusion detection. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 38*(2), 577–583. https://doi.org/10.1109/TSMCB.2007.914695.

Jaiganesh, V., Mangayarkarasi, S., & Sumathi, P. (2014). An efficient algorithm for network intrusion detection system. *International Journal of Computer Applications, 90*(12), 12–16. https://doi.org/10.5120/15771-4100.

Joshi, M. V., Agarwal, R. C., & Kumar, V. (2002). Predicting rare classes: Can boosting make any weak learner strong?. In *Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 297–305).

Joshi, M. V., Kumar, V., & Agarwal, R. C. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. *Proceedings – IEEE International Conference on Data Mining, ICDM, 257–264*. https://doi.org/10.1109/icdm.2001.989527.

Kausar, A., Ishtiaq, M., Jaffar, M. A., & Mirza, A. M. (2010). Optimization of ensemble based decision using PSO. In *Proceedings of the world congress on engineering* (p. 10).

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the IEEE international joint conference on neural networks* (pp. 1942–1948).

Koc, L., Mazzuchi, T. A., & Sarkani, S. (2012). A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Systems with Applications, 39*(18), 13492–13500. https://doi.org/10.1016/j.eswa.2012.07.009.

Kolias, C., Kambourakis, G., Stavrou, A., & Gritzalis, S. (2016). Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Communications Surveys and Tutorials, 18*(1), 184–208. https://doi.org/10.1109/COMST.2015.2402161.

Landesa-Vázquez, I., & Alba-Castro, J. L. (2012). Shedding light on the asymmetric learning capability of AdaBoost. *Pattern Recognition Letters, 33*(3), 247–255.

Landesa-Vázquez, I., & Alba-Castro, J. L. (2013). Double-base asymmetric AdaBoost. *Neurocomputing, 118*, 101–114.

Li, K., Zhou, G., Zhai, J., Li, F., & Shao, M. (2019). Improved PSO_AdaBoost ensemble algorithm for imbalanced data. *Sensors (Switzerland), 19*(6). https://doi.org/10.3390/s19061476.

Ma, J., Liu, X., & Liu, S. (2008). A New Intrusion Detection Method Based on BPSO-SVM. In *Proceedings of the international symposium on computational intelligence and design* (pp. 473–477).

Malik, A. J., Shahzad, W., & Khan, F. A. (2015). Network intrusion detection using hybrid binary PSO and random forests algorithm. *Security and Communication Networks, 8*(16), 2646–2660. https://doi.org/10.1002/sec.508.

Masnadi-Shirazi, H., & Vasconcelos, N. (2007). Asymmetric boosting. In *Proceedings 24th international conference machine learning* (pp. 609–619).

Mazini, M., Shirazi, B., & Mahdavi, I. (2019). Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms. *Journal of King Saud University – Computer and Information Sciences, 31*(4), 541–553. https://doi.org/10.1016/j.jksuci.2018.03.011.

Michailidis, E., Katsikas, S. K., & Georgopoulos, E. (2008). Intrusion detection using evolutionary neural networks. In *Proceedings – 12th Pan-Hellenic conference on informatics*. https://doi.org/10.1109/PCI.2008.53.

Mikhail, J. W., Fossaceca, J. M., & Iammartino, R. (2019). A semi-boosted nested model with sensitivity-based weighted binarization for multidomain network intrusion detection. *ACM Transactions on Intelligent Systems and Technology, 10*(3). https://doi.org/10.1145/3313778.

Mokarian, A., Faraahi, A., & Delavar, A. G. (2013). False positives reduction techniques in intrusion detection systems-A review. *International Journal of Computer Science and Network Security, 13*(10), 128–134.

Mukherjee, I., & Schapire, R. E. (2013). A theory of multiclass boosting. *Journal of Machine Learning Research, 14*(1), 437–497.

Mukkamala, S., Sung, A. H., & Abraham, A. (2005). Intrusion detection using an ensemble of intelligent paradigms. *Journal of Network and Computer Applications, 28*(2), 167–182. https://doi.org/10.1016/j.jnca.2004.01.003.

NSL-KDD Data Set for Network-Based Intrusion Detection Systems. (accessed August, 2019) <https://www.unb.ca/cic/datasets/nsl.html/>, March 2009.

Pfahringer, B. (2000). Winning the KDD99 classification cup. *ACM SIGKDD Explorations Newsletter, 1*(2), 65. https://doi.org/10.1145/846183.846200.

Rodda, S., & Erothi, U. S. R. (2016). Class imbalance problem in the network intrusion detection systems. In *Proceedings of the international conference on electrical, electronics, and optimization techniques* (pp. 2685–2688).

Ryu, D., Jang, J., & Baik, J. (2017). A transfer cost-sensitive boosting approach for cross-project defect prediction. *Software Quality Journal, 25*, 235–272. https://doi.org/10.1007/s11219-015-9287-1.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2008). Resampling or reweighting: A comparison of boosting implementations. *Proceedings – International Conference on Tools with Artificial Intelligence, ICTAI, 1*, 445–451. https://doi.org/10.1109/ICTAI.2008.59.

Srinoy, S. (2007). Intelligence system approach for computer network security. In *Proceedings of the fourth IASTED Asian conference on communication systems and networks* (pp. 89–95).

Sun, Y., Kamel, M. S., Wong, A. K. C., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition, 40*(12), 3358–3378. https://doi.org/10.1016/j.patcog.2007.04.009.

Takahama, T., & Sakai, S. (2018). The velocity updating rule according to an oblique coordinate system with mutation and dynamic scaling for particle swarm optimization. *Artificial Life and Robotics, 23*, 618–627. https://doi.org/10.1007/s10015-018-0498-y.

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *IEEE symposium on computational intelligence for security and defense applications*. https://doi.org/10.1109/CISDA.2009.5356528.

Thaseen, I. S., & Kumar, C. A. (2017). Intrusion detection model using fusion of chi-square feature selection and multi class SVM. *Journal of King Saud University – Computer and Information Sciences, 29*(4), 462–472. https://doi.org/10.1016/j.jksuci.2015.12.004.

Therneau, T., & Atkinson, B. (2019). rpart: Recursive partitioning and regression trees. R package version 4.1-15.

Tian, W., & Liu, J. (2010). Network intrusion detection analysis with neural network and particle swarm optimization algorithm. In Proceeding of 2010 Chinese control and decision conference (pp. 1749–1752).

Ting, K. M. (2000). A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the 17th international conference on machine learning* (pp. 983–990).

Tu, C. J., Chuang, L. Y., Chang, J. Y., & Yang, C. H. (2006). Feature selection using PSO-SVM. In *Lecture notes in engineering and computer science* (pp. 138–143).

Veeramachaneni, K., Yan, W., Goebel, K., & Osadciw, L. (2007). Improving classifier fusion using particle swarm optimization. *Proceedings of IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, 128–135.

Viola, P., & Jones, M. (2002). Fast and robust classification using asymmetric adaboost and a detector cascade. *Advances in Neural Information Processing System, 2*, 1311–1318.

Wang, J., Hong, X., Ren, R. R., & Li, T. H. (2009). A Real-time intrusion detection system based on PSO-SVM. In *Proceedings of the international workshop on information security and application* (pp. 319–321).

Wang, S., & Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 42*(4), 1119–1130. https://doi.org/10.1109/TSMCB.2012.2187280.

Webb, I. G. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning, 40*(2), 980–991. http://users.monash.edu/~webb/Files/SAVEDPDFS/Webb00a.pdf.

Xia, H., & Hoi, S. C. (2013). MKBoost: A framework of multiple kernel boosting. *IEEE Transactions on Knowledge and Data Engineering, 25*(7), 1574–1586. https://doi.org/10.1109/TKDE.2012.89.

Xu, R., Rui, A., & Xiao, F. (2011). Research Intrusion Detection Based PSO-RBF Classifier. In *Proceeding of IEEE 2nd international conference on software engineering and service science* (pp. 104–107).

Yang, L. Y., Zhang, J. Y., & Wang, W. J. (2009). Selecting and combining classifiers simultaneously with particle swarm optimization. *Information Technology Journal, 8*(2), 241–245.

Yang, Y., Zheng, K., Wu, C., & Yang, Y. (2019). Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors (Switzerland), 19*(11). https://doi.org/10.3390/s19112528.

Zainal, A., Maarof, M. A., & Shamsuddin, S. M. (2007). Feature selection using rough-DPSO in anomaly intrusion detection. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) LNCS, 4705*(PART 1), 512–524. https://doi.org/10.1007/978-3-540-74472-6_42.

Zambrano-Bigiarini, M., & Rojas, R. (2013). A model-independent Particle Swarm Optimisation software for model calibration. *Environmental Modelling & Software, 43*, 5–25.

Zhang, H., Gao, H. H., & Wang, X. Y. (2008). Quantum Particle swarm optimization based network intrusion feature selection and detection. In *Proceedings of the 17th world congress the international federation of automatic control* (pp. 12312–12317).

Zhou, T., Li, Y., & Li, J. (2009). Research on intrusion detection of SVM based on PSO. In *Proceedings of the international conference on machine learning and cybernetics* (pp. 1205–1209).

Zhu, Z., Tang, B., & Yuan, J. (2017). Multirobot task allocation based on an improved particle swarm optimization approach. *International Journal of Advanced Robotics Systems, 14*(3), 1–22.

Zhu, J., Zou, H., Rosset, S., & Hastie, T. (2009). Multi-class AdaBoost. *Statistics and Its Interface, 2*(3), 349–360. https://doi.org/10.4310/sii.2009.v2.n3.a8.