Received 1 July 2024, accepted 6 August 2024, date of publication 14 August 2024, date of current version 26 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3443408

RESEARCH ARTICLE

IDRandom-Forest: Advanced Random Forest for Real-Time Intrusion Detection

MUHAMMAD AZHAR[1], SHAHIDA PERVEEN[2], ASMA IQBAL[2], AND BUMSHIK LEE[3], (Member, IEEE)
[1]Department of Applied Data Science, Hong Kong Shue Yan University, Hong Kong, SAR, China
[2]Department of Computer Science, COMSATS University Islamabad, Islamabad 45550, Pakistan
[3]Department of Information and Communication Engineering, Chosun University, Gwangju 61452, Republic of Korea

Corresponding author: Bumshik Lee (bslee@chosun.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) funded by the Korean Government under Grant 2022R1I1A3065473.

ABSTRACT In the last decade, with the increase in cyberattacks the privacy of network traffic has become a critical issue. Currently, simple network intrusion detection techniques are inefficient in terms of time complexity and are characterized by low detection accuracy and high false alarm rates, whereas techniques using complex algorithms such as recurrent neural network (RNN) and transformer-based deep learning, face challenges of high time complexity, large computational resource usage, and high latency rate in detecting intrusion in real-time traffic. To overcome these issues, we propose an advanced intrusion detection random forest "IDRandom-Forest" for real-time intrusion detection with reduced testing time and with higher accuracy. In this technique, an accuracy sliding window and feature weighting based on stratified feature sampling are introduced to determine the optimal sub-ensemble from the classical random forest. Experimental results demonstrated that the proposed hybrid classification system outperforms current state-of-the-art techniques in terms of accuracy and testing time.

INDEX TERMS Cyberattacks, random forest, real-time intrusion detection systems (IDS), stratified subspace sampling.

I. INTRODUCTION

The volume of network traffic has increased exponentially in recent years with the advancements in communication networks and associated services. Consequently, network packet security has become a major challenge. Furthermore, with the increase in cyberattacks, intrusion detection has become a critical field for researchers.

Traditional network security measures are no longer sufficient to protect against the growing sophistication of cyber threats [1]. Real-time intrusion detection systems (IDS) have emerged as a crucial component of comprehensive cybersecurity strategies, providing the ability to identify and respond to malicious activities in a timely manner.

Intrusion detection methods can be classified into two types: misuse-based and anomaly-based ID methods. A misuse-based ID method, also known as a signature-based ID method, can detect diverse types of previously known attacks. Examples of open-source misuse-based tools have been presented in [2] and [3], while anomaly-based ID methods are used to detect zero-day attacks on a network.

The detection of real-time attacks is a critical issue in intrusion detection systems. The number of real-time threats has increased by five-fold in recent years [4]. Therefore, several researchers proposed methods for such attacks, primarily using machine learning techniques to detect various attacks in real-time [5]. A recent example is RF-SVM [8] technique in which a random forest is fused with binary-class SVM and multiclass SVM for anomaly detection and signature detection. Intrusion detection systems are of two types: host- and network-based IDS. A host-based IDS is installed on a computer or a host and continuously monitors the system for malicious data [6]. Meanwhile, network-based IDS can simultaneously monitor multiple systems within a network [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Ali.

Although various approaches have been developed for both host- and network-based IDS, challenges remain, such as low accuracy of the current IDS, high testing time, high false alarm rate, and high computation cost.

Consequently, it is difficult to formulate an accurate and efficient machine-learning technique to factor network traffic in a real situation.

Over the last decade, deep learning-based techniques have been extensively used for intrusion detection [9], [10], [11] and other fields [12], [13]. Long short-term memory (LSTM) [14], bi-directional LSTM (BLSTM) [15], gated recurrent unit (GRU) [16], and bidirectional GRU (BiGRU) [17] are some of the major recurrent neural network (RNN) techniques used in IDS. Huang et al. [9] proposed a Bidirectional Independent Recurrent Neural Network (BiIndRNN) with a Global Attention (GA) mechanism to improve network anomaly detection and claimed an accuracy of 99.04% while false positive rate of 0.36%, outperforming traditional machine learning and RNN-based techniques.

In addition to these RNN-based techniques, transformer-based intrusion detection techniques [19], [20] such as robust transformer-based intrusion detection system (RTIDS) [21] and a transformer-based approach for intrusion detection (TransIDS) [20] have been proposed, to predict various types of network attacks.

Although transformer-based techniques and RNN outperform conventional machine learning (ML) approaches in the offline mode, the key issues of high time complexity, high latency, and high computational cost persist in the real-time mode. Thus, using these deep learning techniques for real-time intrusion detection is not a feasible solution.

Thus, while the advanced deep learning algorithms discussed above offer better accuracy, they come with high latency and computational costs. In contrast, traditional machine learning methods have low latency, shorter testing times, and require fewer computational resources. However, their accuracy is still not as good, which is crucial for an intrusion detection system.

To resolve the issues of accuracy, testing time, computational power and latency concurrently, a two-level classifier was proposed to perform real-time network intrusion detection [5]; however, this method has three major limitations: 1) **Selection of the number of trees:** It uses the classical random forest algorithm, which does not provide any mechanism for determining the number of decision trees required to generate the random forest. The user-defined number of trees is crucial in real-time applications because in the case of too few trees selected by the user, the accuracy can be highly affected, whereas, in the case of a large number of trees being selected, latency and computational cost increase. 2) **Equal weight for all features:** Classical random forest assigns equal weights to all features of the dataset (created from real-time traffic), irrespective of importance, which affects both accuracy and time complexity by increasing latency. 3) **Extra overhead of Level-1 classifier:** In the case of an optimal sub-ensemble, a Level-1 classifier is not required. Instead, an optimal random forest can be directly used for intrusion detection.

To address the issues of high false alarm rates, low accuracy, and high testing time, we propose an advanced random-forest-based hybrid machine learning technique for real-time intrusion detection. Our technique is inspired by the ensemble pruning via individual contribution (EPIC) proposed by [22] and the stratified sampling technique proposed by [23]. In the proposed technique, real-time network traffic is screened using a decision tree (DT) classifier with a specific threshold level of accuracy. In cases where the confidence is less than the specified threshold, an advanced intrusion detection random forest (IDRandom-forest) is used to screen the traffic data for a specific batch, to detect intrusion with high level of confidence.

Although our idea was inspired by the research of [22] and [23], the proposed algorithm resolves many issues by introducing an accuracy sliding window (ASW) and stratified sampling-based feature weighting to determine the optimal sub-ensemble for intrusion detection in real-time network traffic.

The EPIC algorithm proposed by Lu et.al. [22] only employs the concept of positive and negative contributions of decision trees for the voting system but does not discuss the stopping process of pruning ensemble, which is very important in improving accuracy, reducing latency, and decreasing computation cost during the testing process. This bears significance in our case, that is in intrusion detection from real-time network data. Thus, the proposed method introduces the accuracy sliding window to resolve this issue. Ye et.al. [23] presented the basic idea of stratified feature sampling; however, our method modifies this stratified feature weighting process, to detect intrusion with minimum latency in real-time network traffic. Experimental results show that the optimal sub-ensemble reduces latency (incurring less time in decision making), memory (removing bad trees from the actual sub-ensemble), and computational cost (fewer trees in the sub-ensemble requiring less computation during the voting process), providing higher accuracy by retaining the best decision trees in the sub-ensemble and removing bad trees.

The major contributions of this study are as follows:

- A stratified-sampling-based feature weighting method is introduced in the proposed IDRandom-Forest method, which has not been investigated in the context of intrusion detection.
- A window-based accuracy approach is proposed to identify the optimal sub-ensemble in the classical random forest automatically. This unique contribution of our IDRandom-Forest method significantly improves the efficiency of intrusion detection by reducing testing time and increasing accuracy, highlighting the novelty of our approach.
- The proposed hybrid classification method, IDRandom-Forest, outperforms current state-of-the-art techniques regarding accuracy, computational resources,

latency, and testing time on advanced datasets like UNSW-NB15. The results highlight the originality of our proposed method.

The remainder of the paper is structured as follows: Section II discusses related work in the field of intrusion detection; Section III presents the proposed methodology; Section IV describes the experimental settings and results; Section V concludes the paper.

II. RELATED WORKS

Over the last few decades, researchers have continuously proposed techniques to overcome the issue of intrusion detection in networks.

Recently, several intrusion detection and prevention techniques have been deployed to increase intrusion detection effectiveness. After achieving a balanced dataset using the synthetic minority oversampling technique (SMOTE), Abedin et al [24] proposed a random forest (RF) training approach. In addition to this, experimenting with the KDDCUP'99 incursion dataset, cup [25] indicated that the RF algorithm was 92.39% more accurate than comparative methods. After the resampling of minority samples, RF and SMOTE achieved 92.57% accuracy.

Among other studies, Bhati et al. [26] proposed an extreme gradient boosting ensemble-based IDS. Similar to Bhati, Chan and Guestrin [27] showed that XGBoost with an ensemble-based IDS improves outcomes to boost IDS accuracy and performance. Analyses of the KDDCUP'99 data showed a 99.50% success rate for the suggested approach.

To detect denial of service (DOS) and DDoS attacks, a gain ratio characteristic was suggested for feature selection by Nimbalkar and Kshirsagar [28]. Utilizing 16 and 19 features, they achieved 99.9993% and 99.992% accuracy on the IoT-BoT and KDDCUP'99 datasets, respectively, compared to the original feature set and conventional IDS.

Deep neural networks (DNNs) were developed by Chaudhary [29] to detect malicious IoT. The three most significant datasets used to test the attack detection ability of DNN's were: KDDCUP'99 [25], NSL-KDD [30], and UNSW-NB15 [31]. The accuracy of the suggested DNN approach was 91.50% across all datasets. Kennedy and Eberhart [33] experimented with particle swarm optimization, and Talukder et al. [32] used a feature selection technique in conjunction with an artificial neural network (ANN) classifier to differentiate between normal and abnormal incursion activities. Creating their own models, they achieved an accuracy of 98.00% on the KDDCUP'99 dataset.

Further, Varanasi and Razia [34] reviewed the approaches that use feature correlation (CR) for selecting the most important characteristics and a DNN classifier to develop an intrusion detection system, achieving a success rate of 99.40%. Janiga et al. [35] proposed a self-adaptive technique to increase placement optimization, which reduces time complexity to a large extent on the KDDCUP'99 benchmark dataset using a support vector machine (SVM) algorithm while Boser et al. [36] obtained a detection rate of 94.12% on the same dataset.

In LSTM-related studies, Hu et al. [37] created a novel elemental detection approach for network traffic abnormalities using a convolutional attention LSTM network model. Talukder et al. [38] used a convolutional neural network (CNN) to extract shallow features, which were then subjected to attention-LSTM. They tested their method on the KDDCUP'99 dataset, achieving 98.48% accuracy.

Multiple ML methods have identified invasions on KDDCUP'99. Alqahtani et al. [39] contrasted the efficacy of these methods. Raza et al. [40] used a decision tree with a feature reduction technique to achieve 94% accuracy, surpassing all other algorithms. Kumar et al. [41] advised protecting networks against current DOS, exploits, probes, and general assaults using a misuse-based intrusion detection system. Loh et al. [42] tested the classification and regression tree (CART) models on the UNSW-NB15 dataset.

The C5.0 algorithm (C5) proposed by Quinlan [43], chi-squared automatic interaction detection (CHAID) by Kass et al. [44], and quick, unbiased, efficient statistical tree (QUEST) were evaluated using accuracy, irreproducible discovery rate (IDR), and false acceptance rate (FAR) for each model by Kumar et al. [45]. The C5 model had 99.37% success.

Regarding botnets, Koroniotis et al. [46] developed flow identifier-based network forensics to detect malicious botnet activities in motion. Botnet assaults were identified on the UNSW-NB15 dataset using ML classifiers such as the DT algorithm C4.5, association rule mining (ARM), artificial neural network (ANN), and *naïve Bayes* (NB), thereby determining that ARM, DT, NB, and ANN had accuracy rates of 86.45 %, 93.23 %, 72.73 %, and 63.97 %, and false-positive rates of 13.55%, 6.77 %, 27.27 %, and 36.03 %, respectively.

Kasongo and Sun [47] used various machine learning models to accurately forecast based on the UNSW-NB15 IDS dataset of the UNSW-NB15 dataset. The ML techniques include simplifying decision trees, ANN by McCulloch et al. [48], K-nearest neighbour (KNN) [49], SVM [50], and linear regression (LR) [51]. As expected, binary classification accuracy increased from 88.13% to 90.85%. Accuracy was 90.85% for binary classification and 67.57%, 77.51%, 65.29%, 72.30%, and 53.95% for multiclass classification using DT, ANN, LR, KNN, and SVM, respectively.

Kumar et al. [52] improved ensemble models bagging and boosting and obtained an accuracy of 98.24% and 99.95% on the NSL-KDD and Kyoto 2006 datasets, respectively.

On the CIC-IDS2017 and KDDCUP'99 datasets, Rosay et al. [53] decreased feature counts from 77 to 24 and 41 to 12, respectively. The rule-based classifier based on projective adaptive resonance theory obtained 99.96% accuracy in 133.66 seconds using the CIC-IDS2017 dataset and 99.32% accuracy in 11.22% of the time using the KDDCUP'99 dataset.

To improve the anomaly detection reliability of the intrusion system, Mugabe et al. [54] built a MapReduce-based intrusion detection system, thereby extracting a manageable subset of characteristics from massive datasets. By reducing the training set, parallel input data was created using adaptive and effective feature selection. Their model was 93.90% accurate using 15 features.

Talita and Rustam [55] improved intrusion detection by fuzzy C-means and particle swarm optimization (PSO). Their dataset contained approximately 40 features and 400,000 records. Of over 40 characteristics, 38 were selected using particle swarm optimization (PSO) to save computation time and memory. This method demonstrated a remarkable accuracy of 99.12% compared to others.

A redundant penalty-by-feature (RPFMI) mutual information method was presented by Qin et al. [56] to determine appealing malware detection features. the KDDCUP'99 and Kyoto 2006+ intrusion detection datasets were examined, detecting DOS, user-to-root (U2R), and root-to-local (R2L) attacks with 99.77%, 96.19%, and 97.74% accuracy, respectively.

To categorize network intrusions, Mahhizharuvi et al. [57] built a genetically optimized enhanced multi-relational fuzzy decision tree (EMRFT). The K-nearest neighbor approach was employed to fill in the missing values and a rapid correlation-based feature selection method, to minimize data dimensionality and enhance classifier performance, thereby achieving a binary classification accuracy of 98.27% and multilabel classification accuracy of 96.56%.

To correctly identify malicious traffic, Indrasiri et al. [58] suggested an extra boosting forest (EBF) model employing stacked ensembles to merge the ensemble tree (ET), gradient boosting (GB), and RF models. Local network traffic statistics from UNSW-NB15 and IoTID20 were utilized. For each dataset, features were reduced to 30 using principal component analysis. EBF surpassed comparative methods, achieving the highest accuracy score (98.5%) across all four multilabel classes on the UNSW-NB15 and IoTID20 datasets.

Several tree-based ensemble ML methods have been used to identify malware in portable executable files. Louk and Tama [59] used three open-source datasets mentioned in Yang's paper [60]: Kaggle, BODMAS, and CIC-MalMem-2022, to exhibit algorithm flexibility. Performance differences across algorithms were not statistically significant. The GB machine (GBM), XGBoost, and RF outperformed other tree-based ensemble models in this research, achieving 99.39%, 99.96%, and 100% accuracy, respectively.

A recent example is RF-SVM [8] technique in which random forest is fused with binary-class SVM and multi-class SVM for Anomaly detection and Signature detection. Four different datasets NSL-KDD, ISCX-URL2016, CIC-Darknet2020, and CICDoHBrw2020 are used to validate the accuracy improvement in intrusion detection.

To address the issues of low accuracy and high false alarm rate, Huang et al. [9] recently proposed a Bidirectional Independent Recurrent Neural Network (BiIndRNN) with a Global Attention (GA) mechanism. The BiIndRNN model is designed to effectively capture the bidirectional structural features of network traffic, addressing the gradient vanishing/explosion issues inherent in traditional Recurrent Neural Networks (RNNs). The addition of the GA mechanism allows the model to better extract the interrelations between long-distance and interdependent traffic features to improve the accuracy of anomaly detection. To optimize the performance of the proposed method, the influence of the timestep (number of time steps in each sample) on accuracy and FPR is investigated, and the concept of gain ratio is applied to determine the optimal timestep value. Extensive experiments are conducted on the UNSW-NB15 dataset, comparing the GA-BiIndRNN model with traditional machine learning methods and other RNN variants and claimed an accuracy of 99.04% and an FPR of only 0.36%, outperforming the competing techniques.

III. IDRANDOM FOREST ALGORITHM WITH ACCURACY SLIDING WINDOW (ASW)

To overcome the problems of previous IDSs, we suggest an ensemble pruning technique, IDRandom-Forest, based on ASW and feature weighting, to ensure the discovery of ideal sub-ensembles, thereby overcoming the problems of previous IDS classification tools.

In the proposed technique, real-time network traffic is screened by IDRandom-Forest to detect intrusions with a high level of confidence. The major contributions of this study include modifications to the random forest classification algorithm, specifically through the introduction of an accurate sliding window and stratified sampling-based feature weighting. These modifications help determine the optimal sub-ensemble from the classical random forest. The IDRandom-Forest framework and its functions are discussed in detail in the following subsections. In subsections III-A and III-B, the stratified subspace sampling-based feature weighting technique and ASW-based ensemble pruning technique are discussed. ASW and feature-weighting are the key steps in determining the optimal sub-ensemble derived from the classical random forest. The optimal sub-ensemble IDRandom-Forest was proven effective in terms of higher accuracy, lower testing time, and reduced latency in detecting intrusion in live data-flow packets. The intrusion detection process is applied only to the first three consecutive packets of traffic flow instead of the entire flow, to accelerate the process.

A. STRATIFIED SUBSPACE SAMPLING-BASED FEATURE WEIGHTING

The classification accuracy performance of the classifier is directly proportional to the correlation between the selected input and target features [23]. Thus, to improve the accuracy of the random forest ensemble model, the accuracy of the decision tree must be improved, which can be achieved by selecting features that have a high correlation with the target

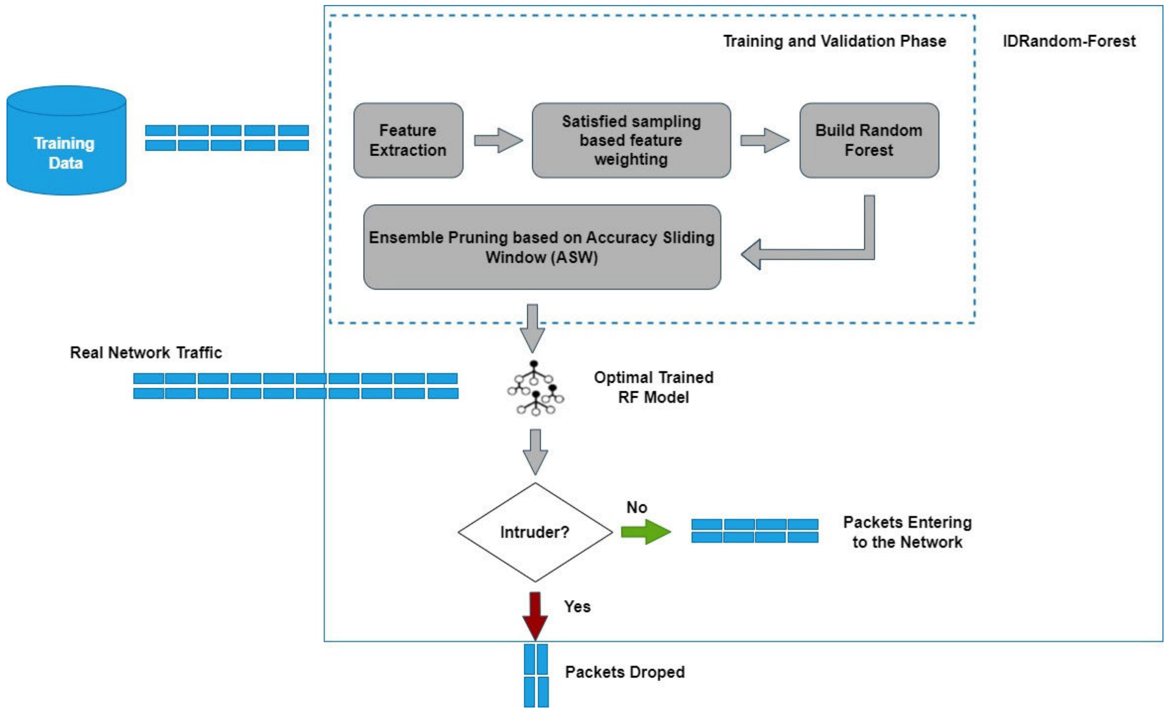
Training and Validation Phase
IDRandom-Forest
Training Data
Feature Extraction
Satisfied sampling based feature weighting
Build Random Forest
Ensemble Pruning based on Accuracy Sliding Window (ASW)
Real Network Traffic
Optimal Trained RF Model
Intruder?
No
Packets Entering to the Network
Yes
Packets Droped

FIGURE 1. **Framework of the proposed IDRandom-Forest classifier.**

variable. The classical random forest method proposed by Breiman [61] randomly selects features with a high probability so that features with low correlation are selected in the subspace. Thus, this problem is solved by weighing the features, whereby features with a low correlation with the target feature are assigned low weights, while those with a high correlation are assigned high weights.

Algorithm 1 outlines the process of assigning weights to the features. The input to Algorithm 1, consists of training data $\boldsymbol{X_{train}}$, non-negative informative function $\boldsymbol{\omega}$, stratification threshold $\boldsymbol{a}$, number of subsets K, number of features in subspace $\boldsymbol{p}$, and the minimum number of instances $\boldsymbol{n_{min}}$. First, the importance of each feature is computed through an informative function $\boldsymbol{\omega}$, which is the Gini index in our case. Subsequently, features are divided into two groups based on the stratification threshold a. Then, K subsets $\{\boldsymbol{X_1}, \boldsymbol{X_2}, \ldots, \boldsymbol{X_k}\}$ are generated through bagging from $\boldsymbol{X_{train}}$ for the K base classifiers (decision trees) to build a random forest. To build each node of the decision tree, features are selected from two feature groups based on proportion and weights provided at the start through informative function $\boldsymbol{\omega}$. Thus, feature selection is not entirely random as in classical random forest. The dataset is split based on the best split computed on the most important feature. The building of decision tree continues until one of the following criteria is satisfied: 1) points belong to the same feature class, or 2) instances are less than $\boldsymbol{n_{min}}$ Finally, all decision trees are combined to generate the random forest ensemble classifier.

B. ENSEMBLE PRUNING WITH ACCURACY SLIDING WINDOW

After applying stratified sampling to the dataset and assigning a weight to each feature according to its importance in detecting the intrusion, the random forest is created similarly to the classical random forest method, but with a feature-weighted dataset. The ASW-based ensemble pruning is subsequently applied, as discussed in detail as follows.

As discussed in relation to EPIC [22], for the two ensembles, the following two properties are observed: 1) When the individual classifiers in two different ensembles have similar accuracy, the more diverse ensemble outperforms the other. 2) In the case of two equally diverse ensembles, the ensemble with the more accurate individual classifiers outperforms the other.

Based on the above mentioned two properties, we propose the ASW-based IDRandom-Forest algorithm, which automatically stops when the optimum sub-ensemble with better accuracy is obtained.

The detailed pruning steps based on the above two properties are explained in Algorithm 2 in which pruning dataset X_{prune}, an ensemble $C_{en} = \{c_1, \ldots, c_k\}$, trained on training dataset X_{train}, are passed as input to the list of ensemble classifiers (decision trees) LC_{en} in decreasing order of contribution. $@_k$ is the output of Algorithm 2.

Let $P = \{p_1, p_2, \ldots, p_M\}$ be a set of M data points, where $p_m = \{(x_m, y_m) \mid m \in [1, M]\}$ is a pair of independent features x_m and label y_m represents the m-th data point; $y_m \in \{y_1, y_2, \ldots, y_L\}$ is a class label that belongs to a set

Algorithm 1 Process of Assigning Weights to Features

Input: training dataset: $\boldsymbol{X_{train}}$, non-negative informative function: $\boldsymbol{\omega}$, stratification threshold: $\boldsymbol{a}$, number of subsets: $\boldsymbol{K}$, number of features in subspace: $\boldsymbol{p}$, minimum number of instances: $\boldsymbol{n_{min}}$

Output: Random forest ensemble: $\boldsymbol{RF}$

```
* For each feature $F_i$ in $X_{train}$:
        Compute its informativeness: $\omega_i = \omega(F_i)$
        Normalize the result: $y_i$ = norm($\omega_i$)
* Divide feature set F into two groups based on the stratification threshold a.
        $F_s$ : Set of features with $y_i \leq a$
        $F_w$ : Set of features with $y_i > a$
* Generate K subsets using bagging.
        $\{X_1, X_2, \ldots, X_k\}$ = bagging ($X_{train}$)
* For each subset $X_i$:
        Build a decision tree: $h_i(X_i)$
        For every node of the Decision Tree Classifier:
          Randomly select $p$ features from the feature subspace
          Select $p_i$ features from $F_s$ and $p_i$ features from $F_w$ (according to the proportions of $F_s$ and $F_w$)
          Split data based on the test function $t$:
              Split the data into right and left nodes.
              Continue building nodes till stopping criteria met:
                  * Points belonging to the same feature class
                  * Instances less than $n_{min}$
Combine the un-pruned classifiers into a random forest ensemble:
$RF = \{c_1(X_1), c_2(X_2), \ldots, c_k(X_k)\}$
```

Algorithm 2 Computing the Contribution of an Individual Classifier Based on Accuracy

Input: pruning dataset X_{prune}, an ensemble $C_{en} = \{c_1, \ldots, c_k\}$ trained on training dataset X_{train}

Output: List LC_{en}

Initialize: A list of vectors $S = \{S^1, S^2, \ldots, S^{M_{prune}} | S^m = [S_1^m, S_2^m, \ldots, S_L^m]\}$, $m \in \{[1, M_{prune}]\}$, where L is the number of class labels, $S_a^m = 0$ (initial number of predictions in label l) on the m-th data point in pruning dataset X_{prune}, M_{prune} is the size of X_{prune}.

Pruning:

```
1. For each $c_k$ in $C$:
     for each $p_b$ in $X_{prune}$:
         get $c_k(x_b)$, # $c_k$'s predictions on $p_b$;
         $S^b_{c_k x_b} = S^b_{c_k x_b} + 1$
2. For each $c_k$ in $C$:
     $\alpha_{k_b} = 1$ if $c_k(x_b) = y_b$ and $c_k(x_b)$ belongs to the minority group; otherwise, 0.
     $\beta_{k_b} = 1$ if $c_k(x_b) = y_b$ and $c_k(x_b)$ belongs to the majority group; otherwise, 0.
     $\theta_{k_b} = 1$ if $c_k(x_b) \neq y_b$, otherwise 0.
         $@_k = \sum_{b=1}^{M} \left(\alpha_{k_b}(2S^b_{maj} - S^b_{c_k(x_b)}) + \beta_{k_b} S^b_{sec} + \theta_{k_b}(S^b_{correct} - S^b_{c_k(x_b)} - S^b_{maj})\right)$
     append pair ($c_k$, $@_k$) to List $LC_{en}$.
Return $LC_{en}$ in decreasing order of $@_k$
```

of L number of output class labels; $C = \{c_1, \ldots, c_k\}$ is a set of K classifiers, where $c_k(x_m)$ denotes the prediction of the k-th classifier on the m-th data point. S is a set of vectors $S^m S = \{S^1, S^2, \ldots, S^M \mid; S^m = [S_1^m, S_2^m, \ldots, S_L^m]\}$; In S^m, S_l^m is the total number of predictions for the l-th label (belonging to $\{y_1, y_2, \ldots, y_L\}z$ set) of the m-th data point of the ensemble combined with majority-voting, and L is the total number of output class labels. In addition, $@_{en}$ is used to represent the accuracy of the entire ensemble, with $@_k$ denoting the accuracy of the single classifier k. To measure the diversity of classifiers within an ensemble, various metrics such as Q and k statistics [62] have been proposed. An important

Algorithm 3 Pruning Process Based on Accuracy Sliding Window

Input: training dataset X_{train}, pruning dataset X_{prune}, testing dataset X_{test}, Number of trees to build nTree, Accuracy sliding window length Φ.

Output: An optimal sub-ensemble c that is pruned based on the pruning dataset X_{prune}

```
Initialize:
$C_{en}$ = nTree
Ensemble Accuracy $@_{en} = 0$
Current Accuracy Sliding Window count $\Phi_{current} = 0$
$C_{en}$= generate_RF(nTree, $X_{train}$)          # Random forest generation based on $X_{train}$
while (nTree) do
    $LC_{en}$ = Compute_Contribution ($C_{en}, X_{prune}$)     # Compute the contributions using Algorithm 1
    $@_{en}$ = Accuracy ($LC_{en}$)
    Remove the tree at the end of the list $C_{en}$ with the lowest individual contribution $@_{k\_lowest}$
    $LC_{en}$--)= Compute_Accuracy ($C_{en}$--), $X_{prune}$)
    $@_{current} \geq$ Accuracy ($LC_{en}$--)
    If ($@_{current} > @_{en}$) then
            nTree--;
        Else
            $\Phi_{current}$ ++ ;
            If ($\Phi_{current}$ == 3)
            Break
Compute F1-score and Accuracy based on testing dataset $X_{test}$ on an optimal sub-ensemble $C_{opt}$
```

measure, the 0/1 loss-based diversity measure, was proposed by Ho [63] and used by Lu et al. [22]. Because our method is based on 0/1 based diversity measure metrics, two important properties are discussed.

According to the 0/1 loss diversity, $M^{(01)}$ denotes the number of network traffic packets incorrectly predicted by c_k but correctly predicted by c_b, where c_k and c_b are the two classifiers in the ensemble. $M^{(10)}$ is the opposite of $M^{(01)}$. Thus, the diversity index $\varepsilon_{k,b}$ between classifiers c_k and c_b of the ensemble is defined as in (1).

$$\varepsilon_{k,b} = \frac{M^{(01)} + M^{(10)}}{M} \tag{1}$$

where $\varepsilon_{k,b}$ is the ratio of the sum of the number of network traffic packets correctly predicted by the classifiers to the total number of network traffic packets.

The second property involves calculating the total diversity contribution of an individual classifier to its ensemble. Total diversity contribution D_k of an individual classifier can be computed as shown in (2).

$$D_k = \sum\nolimits_{b=1}^{K} \varepsilon_{k,b} \tag{2}$$

Thus, $\boldsymbol{D_k}$ is the sum of the diversity contributions of $\boldsymbol{c_k}$ using all other K classifiers.

Based on the two properties mentioned in (1) and (2), it can be inferred that accurate predictions of the network packets contribute positively, whereas inaccurate predictions contribute negatively. Thus, it can be inferred that accurate predictions of the network packets from the minority group make more positive contributions than accurate predictions of the network packets from the majority group. In addition, inaccurate predictions about network packets from the majority group make more negative contributions than inaccurate predictions about network packets from the minority group. These properties were proven in [22]. These rules can be visualized in Fig. 2. Thus, based on the results shown in Fig. 2, the contribution of an individual classifier based on accuracy is defined as in (3).

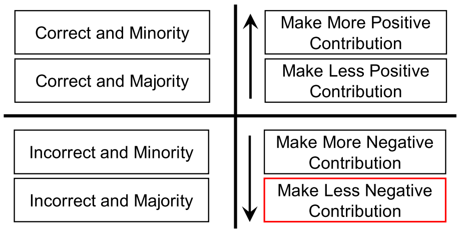
Correct and Minority
Correct and Majority
Make More Positive Contribution
Make Less Positive Contribution
Incorrect and Minority
Incorrect and Majority
Make More Negative Contribution
Make Less Negative Contribution

FIGURE 2. Standards for assessing prediction rules.

$$@_k = \sum\nolimits_{b=1}^{M} @_k^b \tag{3}$$

where $@_k$ (individual classifier contribution based on accuracy) of c_k is the sum of contributions to packet p_b.

If $c_k(x_b) = y_b$, that is, classifier c_k predicts p_b correctly, and $c_k(x_b)$ belongs to the minority group, then the degree of positive contribution of an individual classifier based on accuracy $@_k^b$ is defined as in (4).

$$@_k^b = 2S_{maj}^b - S_{c_k(x_b)}^b \tag{4}$$

where S^b_{maj} denotes the number of majority votes on the network packet p_b, and $S^b_{c_k(x_b)}$ denotes the total number of predictions by classifier c_k on packet p_b, defined as $c_k(x_b)$.

Classifier c_k in the minority group of packets p_b indicates that p_b is difficult to classify and is incorrectly classified by most classifiers. Thus, classifier c_k, which predicts packet p_b correctly belongs to the minority group. Thus, this classifier should not be pruned from the ensemble because of its importance to a specific packet p_b. As $@_k$ is calculated based on predictions of all packets p_b: $b \in [1, M]$, the removal of classifier $\mathrm{c_k}$ is decided based on the $@_k$.

If $c_k(x_b) \neq y_b$, i.e., classifier c_k predicts p_b incorrectly (y_b is the ground truth), then the degree of negative contribution of an individual classifier based on the accuracy $@^b_k$ is defined as in (5).

$$@^b_k = S^b_{correct} - S^b_{c_k(x_b)} - S^b_{maj} \tag{5}$$

where $S^b_{correct}$ is the number of accurate votes for p_b;

TABLE 1. UNSW-NB15 dataset divided into training and testing datasets.

Class	Training	Testing	Total
Normal	17,194	4299	21,493
Fuzzers	572	143	715
Generic	686	171	857
Exploits	401	100	501
Shell code	104	25	129
Reconnaissance	169	42	211
Dos	255	64	319
Total	19,381	4844	24,225

TABLE 2. CICIDS2017 dataset divided into training and testing datasets.

Class	Training	Testing	Total
DDoS	89,503	38,517	128,020
Normal	318,345	136,245	454,590
DoS hulk	161,680	69,282	230,962
Bot	1,390	577	1,967
FTP-patator	5,526	2,414	7,940
SSH-patator	4,117	1,781	5,898
Port scan	111,229	47,694	158,923
DoS slowlris	4,050	1,745	5,795
DoS slowhttptest	3,819	1,611	5,430
DoS GoldenEye	7,193	3,101	10,294
Total	706,852	302,967	1,009,819

$S^b_{c_k(x_b)}$ denotes the total number of predictions by a classifier c_k on packet p_b, defined as $c_k(x_b)$, and S^b_{maj} denotes the number of majority votes on the network packet p_b. Thus, $(S^b_{correct} - S^b_{c_k(x_b)})$ indicates the degree of negative contribution based on the difference between the accurate number of votes for the ground truth and the inaccurate number of votes for $c_k(x_b)$. By combining (3), (4), and (5), the individual contribution of the classifier c_k can be computed as in (6).

$$@_k = \sum\nolimits_{b=1}^{M} \Big(\alpha_{k_b}(2S^b_{maj} - S^b_{c_k(x_b)}) + \beta_{k_b} S^b_{sec} + \theta_{k_b}(S^b_{correct} - S^b_{c_k(x_b)} - S^b_{maj})\Big) \tag{6}$$

where $\alpha_{k_b} = 1$ if $c_k(x_b) = y_b$ and $c_k(x_b)$ is in the minority group. Otherwise, $\alpha_{k_b} = 0$ and $\beta_{k_b} = 1$ if $c_k(x_b) = y_b$ and $c_k(x_b)$ is the majority. Else, $\beta_{k_b} = 0$. Similarly, $\theta_{k_b} = 1$, if $c_k(x_b) \neq y_b$. Otherwise, $\theta_{k_b} = 0$.

Algorithm 3 shows the overall ASW-based process for stopping pruning. Considering input training dataset $\boldsymbol{X_{train}}$, pruning dataset $\boldsymbol{X_{prune}}$, testing dataset $\boldsymbol{X_{test}}$, number of trees to build (nTree), and ASW length Φ, it returns an optimal sub-ensemble $\boldsymbol{C_{opt}} = \{\boldsymbol{c_1, \ldots, c_T} | \boldsymbol{T} \leq \boldsymbol{k}\}$ as output which is pruned based on the pruning dataset $\boldsymbol{X_{prune}}$. The training dataset $\boldsymbol{X_{train}}$ is used to build the initial random forest before pruning based on the input parameter nTree (number of trees to build). ASW length Φ is used to set the threshold for the stopping criterion. The contribution of each decision tree in a random forest is built using the training dataset $\boldsymbol{X_{train}}$ for the computations, and then, the pruning dataset is used to prune the trees one by one based on the contribution of the decision tree. After removing each worst tree from the ensemble, the accuracy difference is computed by ASW. If the accuracy decreases continuously by removing three trees, the pruning process is stopped. The last three trees are added back to the ensemble again, as discussed in Algorithm 3.

Thus, the output of our proposed model is an optimal sub-ensemble pruned according to the accuracy threshold criterion set by Algorithm 3 with the impact (contribution) of each base classifier (decision tree inside random forest ensemble) calculated as discussed in Algorithm 2.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

To validate the proposed method, IDRandomForest, we conducted numerous experiments using various experimental settings, and the experimental results were compared with those of the current state-of-the-art techniques in terms of accuracy, testing time, and latency.

A. DATASET AND EXPERIMENTAL SETTINGS

The proposed approach was examined using other algorithms, such as classical random forest, RF-SVM [8], deep learning-based blindRNN [9] and HAST-IDS [65]. The experiments were conducted on Intel ® Xeon ® Silver 4210 based processor with a base clock speed of 2.20 GHz and graphics cards of 3090 Titan Series with 24 GB of GDDR6X memory and 10496 CUDA cores. The TensorFlow library was used for the CNN of HAST-IDS and for the blindRNN, whereas classical random forest, and the proposed advanced random forest hybrid deep learning technique were implemented using Scikit-learn, NumPy, and Pandas. The HAST-IDS and blindRNN simultaneously utilize GPU and CPU.

For comparison, we used the UNSW-NB15 [31] and CICIDS2017 [66] datasets. The CICIDS2017 dataset is based

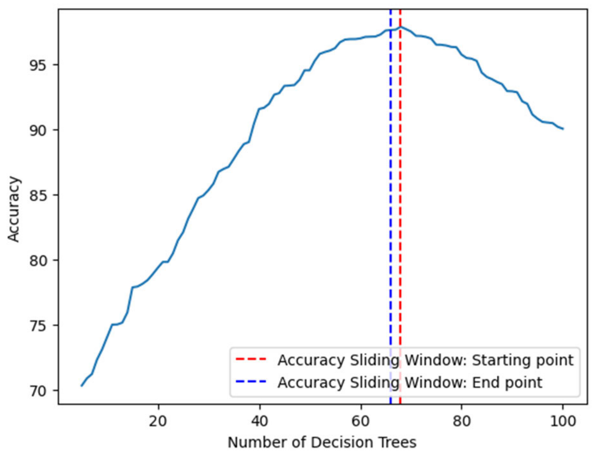

FIGURE 3. Ensemble pruning with stratified feature sampling on UNSW-NB15 dataset; ASW-based highest accuracy is 97.3%.

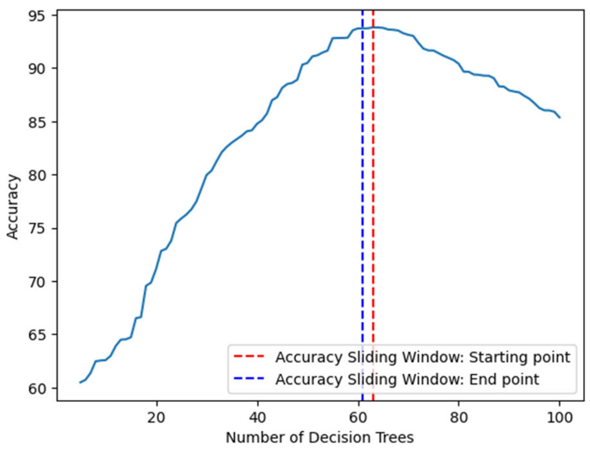

FIGURE 5. Ensemble pruning without stratified feature sampling based on UNSW-NB15 dataset; highest accuracy is 94.2%.

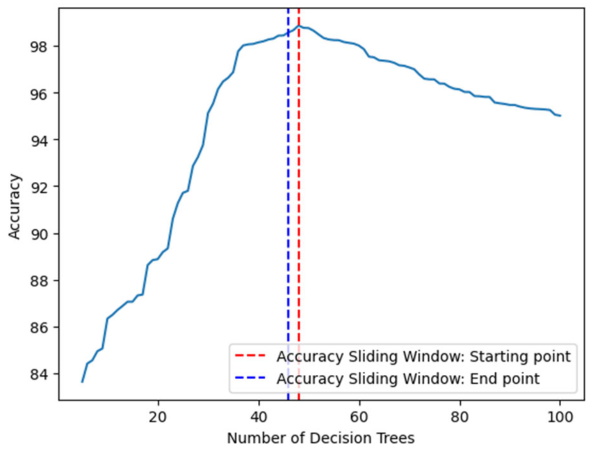

FIGURE 4. Ensemble pruning with stratified feature sampling on CICIDS2017 dataset; ASW-based highest accuracy is 98.9%.

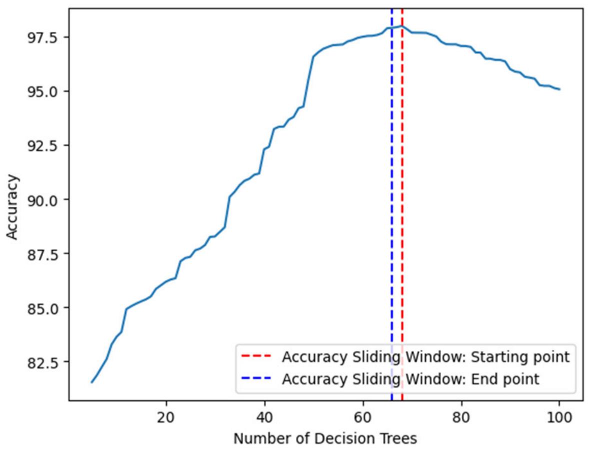

FIGURE 6. Ensemble pruning without stratified feature sampling based on CICIDS2017 dataset; highest accuracy is 97.9%.

on DDoS and DOS attacks. Only seven classes were used to measure progress, one for regular and the others for assault classes. Classes such as worms, open ports, and analyses were removed because of the extremely small number of samples. The UNSW-NB15 and CIC-IDS2017 datasets were used, instead of KDDCUP99 and NSL-KDD (a better version of KDDCUP99) because they are new datasets and are extensively used in recent studies. Although KDDCUP99 and NSL-KDD were widely used in the past for evaluating intrusion detection systems, they have some limitations, such as an unrealistic distribution of attack types, redundant records, outdated patterns not conforming to the latest typical network attacks, and less network diversity.

In contrast, UNSW-NB15 and CIC-IDS2017 were generated in a controlled environment to simulate real-world network traffic, and both datasets contain a wide range of network attacks and normal traffic. These two datasets were specifically designed for evaluating IDS and intrusion prevention system (IPS) solutions and comprise more accurate and diverse samples and attacks.

In addition to the sender and receiver IPs, only the first 100 bytes of the data packet of each flow were used to perform quick processing at Level-1 with the decision tree classifier, and padding with null values was applied for packets of less than 108 bytes. Under uncertainties regarding the threshold value set for intrusion detection, Level-2 classifiers were used, where the one-hot encoding technique was employed to create 230 features, and all features were normalized to be within [0, 1]. We did not apply any feature selection methods; instead, we used a feature weighting method to weigh the features.

The dataset was divided in an 8:2 ratio using the Python library sklearn, in which the proportionality of classes is the most important feature. Eighty percent of the data was used for training and the remaining 20 percent for testing.

The UNSW-NB15 dataset contained a total of 24225 samples, whereas the CICIDS2017 dataset contained 1009819 samples. The datasets were divided into 80 percent training and 20 percent testing. The samples in each class of the UNSW-NB15 and CICIDS2017 datasets are listed in

Tables 1 and 2, respectively. Because the stratified feature sampling technique was used for feature weighting, in the Level-2 classifier, no features were omitted from Level-1 or Level-2.

B. ASW-BASED ENSEMBLE PRUNING RESULTS FOR IDRandomForest

This section discusses the results of the ensemble pruning process of IDRandomForest. The classical random forest algorithm requires the generation of the number of decision trees "nTree." Thus, the user randomly sets a value for the "nTree" parameter without estimating the optimal number of required decision trees in the ensemble.

As the classical random forest randomly selects features (m) out of a total number of features (M) in the dataset, the best feature and best split are chosen from the randomly selected "m" number of features without considering the importance of features for classifying the problem at hand. To improve the feature selection process, we introduced a stratified feature sampling process in our proposed algorithm to assign weights to the features according to their relevancy in terms of importance.

As random forests use a voting system at the end, randomly built decision trees have both types of base classifiers. Some decision trees positively contribute to the voting system, whereas remaining bad decision trees negatively contribute to the final voting in classifying intrusion. Thus, we use specific processes, which are shown in Algorithms 1 to Algorithm 3, to remove the decision trees with negative contributions. In the end, we obtain a sub-ensemble consisting of decision trees only with positive contributions. Thus, voting with these best trees provides higher accuracy than the randomly built classical random forest. As the sub-ensemble size is much smaller than the actual ensemble size, testing time is also reduced. Thus, Figs. 3 to 6 show the improvement of the testing accuracy due to the pruning of each decision tree with a negative contribution.

Fig. 3 shows ensemble pruning (with stratified feature sampling) on the UNSW-NB15 dataset, where the initial accuracy of the ensemble is 89.5 %. Ensemble pruning through ASW improves the accuracy to 97.3%. The red line indicates the start point of the ASW, whereas the endpoint is indicated by the blue line, which is the highest accuracy point. As shown in Fig. 3, the accuracy increases gradually until the removal of the 32nd tree, after which the accuracy starts to decline; according to the sliding window process, the optimal ensemble with 68 trees is finalized and used as the Level-2 classifier on the UNSW-NB15 dataset. The accuracy declines until the ensemble constitutes only five trees. The accuracy of the sub-ensemble is shown for validation purposes with accuracy never improving after ASW confirms that the highest accuracy of 97.3% is achieved.

Fig. 4 describes ensemble pruning (with stratified feature sampling) similarly to that shown in Fig. 3; however, the experiment was performed on the CICIDS2017 dataset, where the accuracy of the ensemble at the beginning of the ensemble pruning process was 94.91%. Although the initial accuracy was improved, our ensemble pruning process increased the accuracy to 98.8%, an increase of approximately 3.89%. As shown in Fig. 4, the accuracy increases gradually until the 52nd tree is removed, after which the accuracy starts to decline. According to the sliding window process, the optimal ensemble with 48 trees is finalized and used as the Level-2 classifier on the CICIDS2017 dataset. As discussed previously, the accuracy decline process continues until the ensemble has only five trees. The accuracy of the sub-ensemble is meant only for validation purposes; accuracy does not improve after ASW confirms that the highest accuracy of 98.9been achieved.

Fig. 5 shows ensemble pruning (without stratified feature sampling) on the UNSW-NB15 dataset, where the initial accuracy of the ensemble is 85.5 %. Ensemble pruning through ASW improves the accuracy to 93.7%. The red line indicates the start point of the ASW, whereas the endpoint is indicated by the blue line, which is the highest accuracy point. As shown in Fig. 5, the accuracy increases gradually until the removal of the 38th tree, after which the accuracy starts to decline; according to the sliding window process, the optimal ensemble with 61 trees is finalized and used as the Level-2 classifier on the UNSW-NB15 dataset. The accuracy declines until the ensemble constitutes only five trees. The accuracy of the sub-ensemble is shown for validation purposes with accuracy never improving after ASW confirms that the highest accuracy of 93.7% is achieved.

Fig. 6 describes ensemble pruning (without stratified feature sampling) similarly to that shown in Fig. 5; however, the experiment was performed on the CICIDS2017 dataset, where the accuracy of the ensemble at the beginning of the ensemble pruning process was 95.11%. Although the initial accuracy was improved, our ensemble pruning process increased the accuracy to 97.5%, an increase of approximately 2.39%. As shown in Fig. 6, the accuracy increases gradually until the 34th tree is removed, after which the accuracy starts to decline. According to the sliding window process, the optimal ensemble with 36 trees is finalized and used as the Level-2 classifier on the CICIDS2017 dataset.

As discussed previously, the accuracy decline process continues until the ensemble has only five trees. The accuracy of the sub-ensemble is meant only for validation purposes; accuracy does not improve after ASW confirms that the highest accuracy of 97.5% has been achieved.

Tables 3 and Table 4 compare the proposed algorithms, IDRF and IDRF (SFS--), with other state-of-the-art algorithms using various evaluation metrics: false alarm rate, precision, recall, F1-Score, accuracy, and test time. Notably, IDRF and IDRF (SFS--) outperform the classical random forest, HAST-100, and HAST-300 in five metrics: precision, recall, accuracy, F1-Score, and false alarm rate. Although the precision, recall, and F1 score of HAST-300 are better than those of IDRF (SFS--), in the trade-off between accuracy

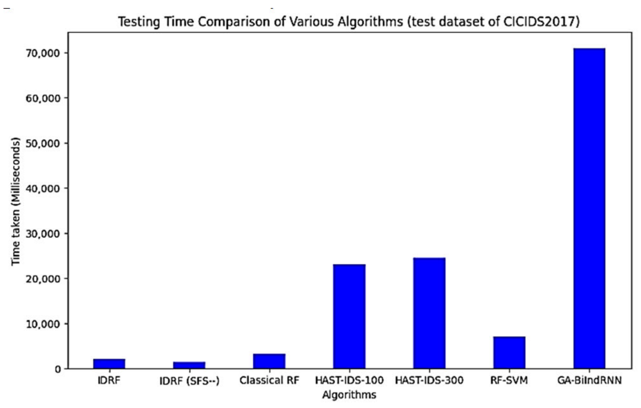

FIGURE 7. Testing time comparison of various algorithms (test dataset UNSW-NB15).

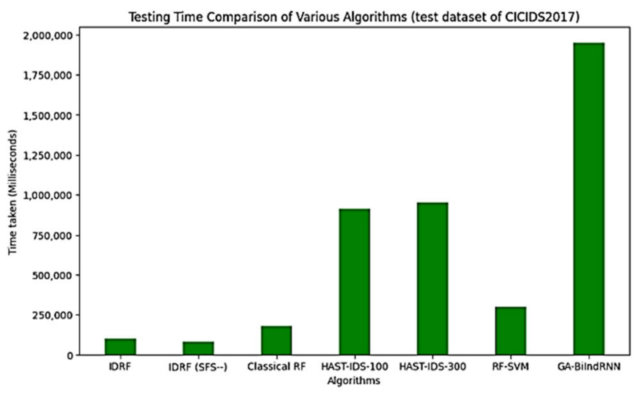

FIGURE 8. Testing time comparison of various algorithms (test dataset CICIDS2017).

TABLE 3. Performance of various algorithms on UNSW-NB15 dataset using different evaluation metrics (Unit: %).

Algorithms	Precision	Recall	F1-Score	Accuracy	False alarm rate
IDRF	96.2	96.8	96.5	96.5	3.8
IDRF (SFS--)	92.3	91.5	91.9	93.7	7.7
Classical RF	88.3	90.2	89.2	88.1	11.7
HAST-IDS (100)	93.6	92.2	92.9	92.2	6.4
HAST-IDS (300)	93.9	92.6	93.2	92.6	6.1
RF-SVM	94.7	93.8	94.2	94.3	5.3
GA-BiIndRNN	96.7	96.5	96.3	96.5	3.3

TABLE 4. Performance of various algorithms on CICIDS2017dataset using different evaluation metrics (Unit: %).

Algorithms	Precision	Recall	F1-Score	Accuracy	False alarm rate
IDRF	95.9	96.2	96.0	96.1	4.1
IDRF (SFS--)	93.1	94.2	93.6	97.5	6.9
Classical RF	92.1	94.3	93.2	93.1	7.9
HAST-IDS (100)	90.1	92.6	91.3	92.2	9.9
HAST-IDS (300)	90.9	91.2	91.0	93.6	9.1
RF-SVM	92.9	94.7	93.6	93.8	7.1
GA-BiIndRNN	94.9	96.5	95.4	95.8	5.1

and testing time, IDRF (SFS--) outperforms, as latency is significant for real-time intrusion detection.

Comparing our proposed method with the deep learning-based GA-BlindRNN, our method performs better in accuracy and F measure on the CICIDS2017 dataset while performing slightly less well on the UNSW-NB15 dataset. However, GA-BlindRNN has a significant issue with testing time, which is a crucial factor for real-time intrusion detection. In summary, the IDRF performs well overall in terms of latency, accuracy, testing time, and F1 measure, validating our method. The false alarm rate is also included to provide a better understanding of the results.

Fig. 7 compares the testing time of IDRF and IDRF(SFS--) with those of other algorithms such as classical random forest, HAST-IDS (100), HAST-IDS (300), RF-SVM and GA-BiIndRNN on the test dataset UNSW-NB15. Clearly, the proposed IDRF(SFS--) incurs only 1,451ms, while IDRF incurs 2,110ms. The time taken by both IDRF and IDRF(SFS--) is less than that of classical random forest (3,200ms) and RF-SVM (7,011ms) and significantly lower than those of other methods such as HAST-IDS (100) and HAST-IDS (300), which incur 23,040, and 24,510ms, respectively. Although GA-BlindRNN slightly outperformed in accuracy and F-measure, it is significantly larger in terms of testing time (71,043, 34 times more than our proposed IDRF). In summary, the average single sample classification time for the proposed IDRF(SFS--) is 0.299ms, while IDRF has 0.435ms, which is less than classical random forest (0.66ms) and RF-SVM (1.45ms). Furthermore, they are significantly faster than the other methods, such as HAST-IDS (100) and HAST-IDS (300), which incur 4.75ms and 5.05ms, respectively.

In addition, we compared the time taken by the proposed methods with those of other methods on the test partition of the CICIDS2017 dataset, which is shown in Fig. 8. The proposed IDRF(SFS--) incurred the least time, followed by IDRF, displaying the same trend observed with the UNSW-NB15 dataset. The proposed IDRF(SFS--) consumed only 84,032ms, while IDRF took 99,110ms. The time consumed by both IDRF and IDRF(SFS--) was less than that of classical random forest (181,530ms), RF-SVM (301,130ms), and significantly less than HAST-IDS (100), HAST-IDS (300) and GA-BlindRNN which incurred 912,930ms, 951,043ms and 1,951,043ms, respectively. Notably, the accuracy of IDRF was better than that of IDRF(SFS--), and the time taken by IDRF(SFS--) was the lowest. In the average single sample classification time comparison, the proposed IDRF(SFS--) took 0.277ms. In contrast, IDRF took 0.327ms, which is less than classical random forest (0.599ms), RF-SVM (0.993ms) and significantly faster than the other methods such as HAST-IDS (100), and HAST-IDS (300), and GA-BlindRNN which incur 3.01ms, and 3.14ms, and 6.4ms, respectively.

Both versions of the proposed algorithm showed promising results compared to other state-of-the-art methods.

V. CONCLUSION

Current network intrusion detection techniques are inefficient in terms of testing time and detection accuracy. Simple intrusion detection has low detection accuracy and a high false alarm rate, whereas complex algorithms such as RNN- and transformer-based deep learning techniques, have high time complexity and require considerable computational resources. To overcome these issues, in this study, we propose an advanced random forest for real-time intrusion detection in less time with high accuracy. ASW and feature weighting (based on stratified feature sampling) are used to determine the optimal sub-ensemble from the classical random forest. The experimental results demonstrate that the proposed hybrid classification system outperforms current state-of-the-art techniques in terms of accuracy and latency.

REFERENCES

[1] S. X. Wu and W. Banzhaf, "The use of computational intelligence in intrusion detection systems: A review," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 1–35, Jan. 2010.

[2] M. Roesch, "Snort: Lightweight intrusion detection for networks," *Lisa*, vol. 99, no. 1, pp. 229–238, 1999.

[3] O. Erdogan and P. Cao, "Hash-AV: Fast virus signature scanning by cache-resident filters," *Int. J. Secur. Netw.*, vol. 2, no. 1, p. 50, 2007.

[4] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 833–844.

[5] W. Seo and W. Pak, "Real-time network intrusion prevention system based on hybrid machine learning," *IEEE Access*, vol. 9, pp. 46386–46397, 2021.

[6] V. R. Varanasi and S. Razia, "Intrusion detection using machine learning and deep learning," *Int. J. Recent Technol. Eng.*, vol. 1, pp. 1–24, Nov. 2019.

[7] H. P. S. Sasan and M. Sharma, "Intrusion detection using feature selection and machine learning algorithm with misuse detection," *Int. J. Comput. Sci. Inf. Technol.*, vol. 8, no. 1, pp. 17–25, Feb. 2016.

[8] P. Dey and D. Bhakta, "A new random forest and support vector machine-based intrusion detection model in networks," *Nat. Acad. Sci. Lett.*, vol. 46, no. 5, pp. 471–477, Oct. 2023.

[9] H. Li, H. Ge, H. Yang, J. Yan, and Y. Sang, "An abnormal traffic detection model combined BiIndRNN with global attention," *IEEE Access*, vol. 10, pp. 30899–30912, 2022.

[10] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Appl. Sci.*, vol. 9, no. 20, p. 4396, Oct. 2019.

[11] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, Jan. 2021, Art. no. e4150.

[12] M. S. Ali, M. Azhar, S. Masood, B. Lee, T. Iqbal, and A. Amjad, "Efficient video summarization with hydra attentive vision transformer," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2023, pp. 196–201.

[13] K. Hussain, M. Azhar, B. Lee, A. Iqbal, M. Affan, and S. U. Khan, "ASAnalyzer: Attention based sentiment analyzer for real-world sentiment analysis," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2023, pp. 184–189.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[15] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.

[16] R. A. Ahmad, M. Azhar, and H. Sattar, "An image captioning algorithm based on the hybrid deep learning technique (CNN+GRU)," in *Proc. Int. Conf. Frontiers Inf. Technol. (FIT)*, Dec. 2022, pp. 124–129.

[17] S. Zia, M. Azhar, B. Lee, A. Tahir, J. Ferzund, F. Murtaza, and M. Ali, "Recognition of printed Urdu script in nastaleeq font by using CNN-BiGRU-GRU based encoder–decoder framework," *Intell. Syst. Appl.*, vol. 18, May 2023, Art. no. 200194.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[19] R. Kozik, M. Pawlicki, and M. Choras, "A new method of hybrid time window embedding with transformer-based traffic data classification in IoT-networked environment," *Pattern Anal. Appl.*, vol. 24, no. 4, pp. 1441–1449, Nov. 2021.

[20] P. Wang, X. Wang, Y. Song, J. Huang, P. Ding, and Z. Yang, "TransIDS: A transformer-based approach for intrusion detection in Internet of Things using label smoothing," in *Proc. 4th Int. Conf. Comput. Eng. Appl. (ICCEA)*, Apr. 2023, pp. 216–222.

[21] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "RTIDS: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022.

[22] Z. Lu, X. Wu, X. Zhu, and J. Bongard, "Ensemble pruning via individual contribution ordering," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2010, pp. 871–880.

[23] Y. Ye, Q. Wu, J. Zhexue Huang, M. K. Ng, and X. Li, "Stratified sampling for feature subspace selection in random forests for high dimensional data," *Pattern Recognit.*, vol. 46, no. 3, pp. 769–787, Mar. 2013.

[24] M. Z. Abedin, C. Guotai, P. Hajek, and T. Zhang, "Combining weighted SMOTE with ensemble learning for the class-imbalanced prediction of small business credit risk," *Complex Intell. Syst.*, vol. 9, no. 4, pp. 3559–3579, Aug. 2023.

[25] K. D. D. Cup. (1999). *The UCI KDD Archive*. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[26] B. S. Bhati, G. Chugh, F. Al-Turjman, and N. S. Bhati, "An improved ensemble based intrusion detection technique usingXGBoost," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 6, Jun. 2021, Art. no. e4076.

[27] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

[28] P. Nimbalkar and D. Kshirsagar, "Feature selection for intrusion detection system in Internet-of-Things (IoT)," *ICT Exp.*, vol. 7, no. 2, pp. 177–181, Jun. 2021.

[29] P. Chaudhary, B. B. Gupta, and A. K. Singh, "Adaptive cross-site scripting attack detection framework for smart devices security using intelligent filters and attack ontology," *Soft Comput.*, vol. 27, no. 8, pp. 4593–4608, Apr. 2023.

[30] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 2, pp. 1848–1853, 2013.

[31] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[32] H. Lin, "DeepShield: A hybrid deep learning approach for effective network intrusion detection," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 7, pp. 1–21, Jul. 2023.

[33] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95-Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[34] V. Varanasi and S. Razia, "Network intrusion detection using machine learning, deep learning–A review," in *Proc. 4th Int. Conf. Smart Syst. Inventive Technol. (ICSSIT)*, Jan. 2022, pp. 1618–1624.

[35] D. Janiga, R. Czarnota, J. Stopa, and P. Wojnarowski, "Self-adapt reservoir clusterization method to enhance robustness of well placement optimization," *J. Petroleum Sci. Eng.*, vol. 173, pp. 37–52, Feb. 2019.

[36] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, Pittsburgh, PA, USA, Jul. 1992, pp. 144–152.

[37] F. Hu, S. Zhang, X. Lin, L. Wu, N. Liao, and Y. Song, "Network traffic classification model based on attention mechanism and spatiotemporal features," *EURASIP J. Inf. Secur.*, vol. 2023, no. 1, pp. 1–6, Jul. 2023.

[38] M. A. Talukder, K. F. Hasan, M. M. Islam, M. A. Uddin, A. Akhter, M. A. Yousuf, F. Alharbi, and M. A. Moni, "A dependable hybrid machine learning model for network intrusion detection," *J. Inf. Secur. Appl.*, vol. 72, Feb. 2023, Art. no. 103405.

[39] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb, "A survey on Android malware detection techniques using machine learning algorithms," in *Proc. 6th Int. Conf. Softw. Defined Syst. (SDS)*, Jun. 2019, pp. 110–117.

[40] S. A. R. Shirazi, S. Shamim, A. H. Khan, and A. Anwar, "Intrusion detection using decision tree classifier with feature reduction technique," *Mehran Univ. Res. J. Eng. Technol.*, vol. 42, no. 2, p. 30, Mar. 2023.

[41] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection system: Analysis on UNSW-NB15 data set and the real time online dataset," *Cluster Comput.*, vol. 23, no. 2, pp. 1397–1418, Jun. 2020.

[42] W. Y. Loh, "Classification and regression trees," *Wiley Interdiscipl. Reviews, Data Mining Knowl. discovery*, vol. 1, no. 1, pp. 14–23, Jan. 2011.
[43] J. R. Quinlan, *C4. 5: Programs for Machine Learning*. Amsterdam, The Netherlands: Elsevier, 2014.
[44] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *Appl. Statist.*, vol. 29, no. 2, p. 119, 1980.
[45] V. Kumar, A. K. Das, and D. Sinha, "Statistical analysis of the UNSW-NB15 dataset for intrusion detection," in *Proc. CIPR*, 2019, pp. 279–294.
[46] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.
[47] S. M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," *J. Big Data*, vol. 7, no. 1, pp. 1–20, Dec. 2020.
[48] W. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biol.*, vol. 52, nos. 1–2, pp. 99–115, 1990.
[49] E. Fix, *Discriminatory Analysis: Nonparametric Discrimination, Consistency Properties*, vol. 1. Wright-Patterson, OH, USA: USAF school of Aviation Medicine, 1985.
[50] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory*, Jul. 1992, pp. 1–27.
[51] S. M. Stigler, "Gauss and the invention of least squares," in *The Annals of Statistics*. New York, NY, USA: JSTOR, 1981, pp. 465–474.
[52] G. Kumar, "An improved ensemble approach for effective intrusion detection," *J. Supercomput.*, vol. 76, no. 1, pp. 275–291, Jan. 2020.
[53] A. Rosay, E. Cheval, F. Carlier, and P. Leroux, "Network intrusion detection: A comprehensive analysis of CIC-IDS2017," in *Proc. 8th Int. Conf. Inf. Syst. Secur. Privacy*, 2022, pp. 25–36.
[54] E. Mugabo, Q. Y. Zhang, A. Ngaboyindekwe, V. P. N. Kwizera, and V. E. Lumorvie, "Intrusion detection method based on mapreduce for evolutionary feature selection in mobile cloud computing," *Int. J. Netw. Secur.*, vol. 23, no. 1, pp. 106–115, 2021.
[55] A. S. Talita and Z. Rustam, "Klasifikasi masalah intrusion detection system dengan menggunakan metode fuzzy C-means dan Laplacian score," *Jurnal Ilmiah Komputasi*, vol. 15, pp. 19–26, Sep. 2016.
[56] X.-C. Qin, M. Shi, J.-H. Tian, X.-D. Lin, D.-Y. Gao, J.-R. He, J.-B. Wang, C.-X. Li, Y.-J. Kang, B. Yu, D.-J. Zhou, J. Xu, A. Plyusnin, E. C. Holmes, and Y.-Z. Zhang, "A tick-borne segmented RNA virus contains genome segments derived from unsegmented viral ancestors," *Proc. Nat. Acad. Sci. USA*, vol. 111, no. 18, pp. 6744–6749, May 2014.
[57] P. Mahhizharuvi, "An effective intrusion detection system using enhanced multi relational fuzzy tree," *Turkish J. Comput. Math. Educ. (TURCOMAT)*, vol. 12, pp. 3152–3159, Jan. 2021.
[58] O. Olayemi Petinrin, F. Saeed, X. Li, F. Ghabban, and K.-C. Wong, "Malicious traffic detection in IoT and local networks using stacked ensemble classifier," *Comput., Mater. Continua*, vol. 71, no. 1, pp. 489–515, 2022.
[59] M. H. L. Louk and B. A. Tama, "Tree-based classifier ensembles for PE malware analysis: A performance revisit," *Algorithms*, vol. 15, p. 332, Jun. 2022.
[60] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh, and G. Wang, "BODMAS: An open dataset for learning based temporal analysis of PE malware," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2021, pp. 78–84.
[61] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[62] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Mach. Learn.*, vol. 40, pp. 139–157, Jul. 2000.
[63] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.
[64] C. Guo, Y. Ping, N. Liu, and S.-S. Luo, "A two-level hybrid approach for intrusion detection," *Neurocomputing*, vol. 214, pp. 391–400, Nov. 2016.
[65] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial–temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
[66] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 4, pp. 479–482, 2018.

MUHAMMAD AZHAR received the B.S. degree in computer science from the National University of Computer and Emerging Sciences, Islamabad, Pakistan, in 2011, the M.S. degree in computer science from Sejong University, Seoul, South Korea, in 2014, and the Ph.D. degree in machine learning and data mining from Shenzhen University, Shenzhen, China, in 2020. He was a Research Professor at Chosun University, from 2022 to 2023. He is currently an Assistant Professor with the Department of Applied Data Science, Hong Kong Shue Yan University, SAR, China. His research interests include natural language processing and computer vision through conventional machine learning and deep learning.

SHAHIDA PERVEEN received the B.C.S. degree from Punjab College Burewala, the M.C.S. degree from Virtual University Islamabad, and the M.S. degree from COMSATS University Islamabad. Her research interests include IoT intrusion detection and machine learning, among others.

ASMA IQBAL received the B.S. degree from GC University Faisalabad, Pakistan, in 2017, and the M.S. degree in computer science from COMSATS University Islamabad, Pakistan, in 2023. She is currently a Lecturer with the Department of Computer Science, COMSATS University Islamabad, Sahiwal Campus, Pakistan. Her research interests include conventional machine learning, deep learning, and computer vision.

BUMSHIK LEE (Member, IEEE) received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2000, and the M.S. and Ph.D. degrees in information and communications engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2006 and 2012, respectively. He was a Research Professor with KAIST, in 2014, and a Postdoctoral Scholar with the University of California at San Diego, San Diego, CA, USA, from 2012 to 2013. He was a Principal Engineer with the Advanced Standard Research and Development Laboratory, LG Electronics, Seoul, from 2015 to 2016. He is currently a Professor with the Department of Information and Communications Engineering, Chosun University, South Korea. His research interests include video processing, video security, and medical image processing.