IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 April 2023, accepted 1 May 2023, date of publication 8 May 2023, date of current version 12 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3274474

RESEARCH ARTICLE

Network Intrusion Detection with Two-Phased Hybrid Ensemble Learning and Automatic Feature Selection

ASANKA KAVINDA MANANAYAKA AND SUN SUNNIE CHUNG
Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH 44115, USA
Corresponding author: Asanka Kavinda Mananayaka (a.mananayaka@vikes.csuohio.edu)

ABSTRACT The use of network connected devices has grown exponentially in recent years revolutionizing our daily lives. However, it has also attracted the attention of cybercriminals making the attacks targeted towards these devices increase not only in numbers but also in sophistication. To detect such attacks, a Network Intrusion Detection System (NIDS) has become a vital component in network applications. However, network devices produce large scale high-dimensional data which makes it difficult to accurately detect various known and unknown attacks. Moreover, the complex nature of network data makes the feature selection process of a NIDS a challenging task. In this study, we propose a machine learning based NIDS with Two-phased Hybrid Ensemble learning and Automatic Feature Selection. The proposed framework leverages four different machine learning classifiers to perform automatic feature selection based on their ability to detect the most significant features. The two-phased hybrid ensemble learning algorithm consists of two learning phases, with the first phase constructed using classifiers built from an adaptation of the One-vs-One framework, and the second phase constructed using classifiers built from combinations of attack classes. The proposed framework was evaluated on two well-referenced datasets for both wired and wireless applications, and the results demonstrate that the two-phased ensemble learning framework combined with the automatic feature selection engine has superior attack detection capability compared to other similar studies found in the literature.

INDEX TERMS Feature selection, feature engineering, classification, machine learning, ensemble learning, anomaly detection, intrusion detection system.

I. INTRODUCTION

Network connectivity has become an integral part of our daily lives to the point that even a few minutes of unexpected downtime in connectivity can result in severe disruptions. Wired ethernet networks such as IEEE 802.3 have played a major role in connecting network capable devices for the past several decades and will continue to play a major role for the foreseeable future [1]. The introduction of wireless standards such as IEEE 802.11 further boosted the use of connected devices due to the mobility and portability provided by such networks [2]. The recent emergence of Internet-of-Things (IoT) has further accelerated the growth of wireless networks to the point that the number of network devices is expected to reach 29 billion within the next decade [3].

With this growing popularity, the security and privacy aspects of both wired and wireless networks have gained considerable traction in recent years [4]. These security concerns are typically addressed by a Network Intrusion Detection System (NIDS) that sits at the center of a network performing constant monitoring of ingress and egress packets to detect attacks. Zarpelao et al. [5] classified NIDSs that are in use today into three different categories based on their detection technique: specification-based, signature-based, and anomaly-based. Similar taxonomies can be found in [6], [7], and [8]. A specification-based NIDS works by defining the normal expected behavior of a network and any deviation from this expected behavior is flagged as an

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

anomaly or ''an attack''. They tend to have a low false positive rate because the rules are manually derived. However, the process of manual rule induction to profile the normal system behavior is a major drawback of such systems, especially when used in a larger network. A signature-based NIDS works by detecting signatures of known attacks. It can have a low false positive rate because the signature detection can be fairly accurate for known attacks, provided the feature set is small enough for signature generation. However, manual identification of signatures from high-dimensional network data can be unreliable and prone to human error. Moreover, a signature-based NIDS cannot detect new and unknown attacks (also known as zero-day attacks) as it only flags a matching signature to a previously known attack. Lastly, an anomaly-based NIDS works by primarily employing a machine learning algorithm to detect anomalous behavior of a system. This type of NIDS is far superior to previously mentioned types because: One, it does not require rules to be manually derived. Two, it can detect new and unknown attacks - which is one of the most desired features of a modern NIDS. Almost all new network intrusion detection systems utilize some form of a machine learning based architecture. The work proposed as part of this study will also fall under this branch of intrusion detection systems.

There are several challenges associated with designing an effective machine learning based NIDS. Network devices produce large volumes of complex high-dimensional data which gives meaning to the very definition of big-data. The high dimensionality alone creates several new challenges. For instance, the NIDS has to select the features that provide the best performance without overfitting the model. With high-dimensional data, selecting too many features can lead to issues such as the curse of dimensionality [9]. Therefore, accurate automatic feature selection becomes very important with largescale high-dimensional network data since manual feature selection can be a daunting task that can lead to incorrect feature selection or incorrect rule specification. Automatic feature selection for network data remains to be an active research topic. Another important challenge with designing an effective NIDS is the ability to detect new or unknown attacks. Most new attacks are variations of existing attacks that evolved over time. The threat-landscape is constantly changing with new sophisticated attacks emerging more frequently [10]. For example, the recent WannaCry attack affected more than 230,000 computers in 150 countries including computers belonging to high profile government organizations [11]. Some new attacks can target critical infrastructure that if not protected, can create a national disaster. The Colonial Pipeline attack that took place in 2021 is a prime example of this [12]. Therefore, the NIDS must be designed in such a way to be resilient to these new and emerging attacks. Another challenge associated with a NIDS is the difference in protocols between wired and wireless ethernet networks. Unlike wired networks, wireless networks are open in nature and can be accessed by anyone in the vicinity of the network. This makes wireless networks susceptible to new attacks that are not found in wired ethernet. From a NIDS perspective, this makes designing a unified solution that works in both wired and wireless applications a difficult task.

Several studies have attempted to address the challenges associated with machine learning based NIDSs. A recent study done by Aminanto et al. [8] focused on feature selection and feature engineering to improve attack detection. They showed that in addition to feature selection, adding a feature engineering step can further boost the model performance, especially to detect impersonation type attacks. However, they only applied their method to impersonation attacks essentially turning it into a binary classification problem. They did not expand their method to other attack types. The ability to detect and classify multiple attack types (multiclass) is an important aspect of a NIDS because the mitigation or response can be different from one attack type to another. Unfortunately, multiclass classification suffers from lower accuracy compared to binary class classification. Furthermore, automatic feature selection, which has become an essential step in modern NIDSs, can lead to a decrease in model accuracy if not done properly. Unknown attack detection adds another layer of complexity to the mix. A recent study done by [13] used One-vs-Rest ensemble framework to address some of these challenges with multiclass attack detection. They used a set of nested ensembles with and without boosting to come up with a new ensemble framework that works in both wired and wireless applications. Their best performing model achieved a detection rate of 86% for the wireless application. As such, there is considerable room for improvement. This motivated us to build upon their work and introduce a new ensemble framework to further improve the attack detection rate.

In this study, we propose a Network Intrusion Detection System with a Two-phased Hybrid Ensemble learning algorithm and Automatic Feature Selection (THE-AFS) to detect various cyber-attacks targeted toward network connected devices. To address the high dimensionality of network data, we employ an automatic feature selection framework by leveraging four different machine learning classifiers based on their ability to detect the most significant features. The selected features are supplied to the learning algorithm consisting of multiple learners where each learner is composed of a two-phased prediction algorithm. The first phase is tasked with generating a list of attack candidates whereas the second phase is tasked with narrowing that list to a specific attack type. The first phase is constructed using an adaption of the One-vs-One multiclass ensemble framework trained on normal-to-attack binary classifiers. The second phase is constructed using a set of multiclass classifiers that are trained using every combination of attack classes. The two-phased mechanism within a learner makes this algorithm a hybrid ensemble because the two phases are built using two distinct models that work in stages (the terms phase and stage are used synonymously throughout this study). The two phases go

through a voting mechanism to produce a high detection rate while keeping a low false alarm rate. While there are many One-vs-Rest based intrusion detection algorithms found in the literature [13], [14], [15], an One-vs-One based ensemble learning algorithm for an intrusion detection framework has not been explored enough by the research community. This study aims to fill that gap by successfully adapting an One-vs-One architecture in an ensemble framework for intrusion detection.

Finally, we evaluated the proposed method for both wired (enterprise networks) and wireless (802.11) networks with datasets that contain real traces of attacks. NSL-KDD [16] dataset was used to evaluate the wired application and AWID [17] dataset was used to evaluate the wireless application. Both datasets have separate training and test sets. The test set contains new attacks that are not found in the training set. Even though it can be challenging to detect such attacks in the test set, it provides an opportunity to evaluate the system's capability to detect new attack signatures. The wired application achieved a multiclass detection rate of **0.9431** and a false alarm rate of **0.0005**. The wireless application achieved a multiclass detection rate of **0.9314** and a false alarm rate of **0.0144.** The notable result was in the wireless application where it surpassed the TPR/FPR ratio of other leading studies that attempted to build a generalized model for both wired and wireless ethernet applications. The main contributions of this study are:

- An effective feature engineering method for high-cardinality features followed by an automatic feature selection method for large-scale high dimensional network data.
- A novel two-phased hybrid ensemble learning algorithm based on an enhanced One-vs-One framework for multiclass attack classification.
- A generalized intrusion detection framework that can be utilized in both wired and wireless applications to detect variations of known and unknown attacks, with high detection rate and low false alarm rate compared to other similar work in the literature

The remainder of this paper is organized into the following sections: Section II presents a literature review of other latest architectures used for intrusion detection. Section III presents background on ensemble methods and class binarization methods. Section IV presents detailed implementation of proposed methodologies, including system architecture, feature engineering methods, automatic feature selection methods, and two-phased hybrid ensemble learning method with voting algorithm used by the proposed framework. Section V presents experimental results and comparisons with other leading studies. Finally, Section VI presents the closing remarks.

II. LITERATURE REVIEW

Recent studies have proposed various ensemble methods for anomaly-based intrusion detection systems [18], [19], [20], [21], [22]. Aburomman and Reaz [23] discussed the advantages of using ensemble methods for intrusion detection where they pointed out that intrusions can come in various forms and having multiple learners trained on different attack types can increase the overall likelihood of detection. Over the years, ensemble methods have evolved into two distinct categories; homogenous ensembles where the base learners use the same learning technique, and heterogeneous ensembles where the base learners use diverse learning techniques [24]. Heterogeneous ensembles can be further categorized into two methods called stacking and voting based on how the individual learner predictions are combined into a single prediction. A stacking ensemble is comprised of one or more base-models and a final meta-model, where the output from the base-models serve as input to the meta-model [25], [26], [27]. While the base-models learn to produce an intermediate output from the training data, the meta-model learns to predict the true class label from that intermediate output. The most popular method to build a heterogeneous ensemble is voting [28], [29], [30], where the learner predictions go through a voting mechanism to arrive at the final prediction.

Recent studies have proposed various advanced ensemble methods for intrusion detection. A multilevel ensemble was proposed by Zhou et al. [14] where they extended the AdaBoost-A [31] algorithm for multiclass classification using the one-vs-rest strategy. Their ensemble architecture was composed of multiple expert learners and each expert learner was composed of multiple one-vs-rest sub-learners trained as weak base learners. An expert learner makes a prediction by combining the predictions from each sub-learner and then the ensemble makes a prediction by combining the predictions from each expert learner. They introduced two variations of this algorithm based on SVM and Particle Swarm Optimization (PSO) [32].

A sustainable ensemble learning model was proposed by Li et al. [33] where they attempted to solve two problems with current ensemble architectures. One, they introduced a weighting mechanism so that each classifier output is weighted differently for each attack type. The weights were trained by considering the sensitivity of each attack type as opposed to a single weight value for all the classifiers. Two, they introduced a sustainable update mechanism where the model can be retrained with new data while maintaining historical knowledge of the older model. Their proposed method was able to achieve a higher accuracy than classical ensemble architectures.

Louk and Tama [34] proposed a dual ensemble model by constructing the base learners using a second ensemble learning method as opposed to the base learners being a classical machine learning method. They experimented with various state-of-the-art Gradient Boosting Decision Tree (GBDT) models as base learners. They also used a bagging ensemble in series with the GBDT ensemble to improve the performance accuracy. As a result, it was called a dual ensemble architecture. They experimented with three different datasets and in all three instances, their Bagging-GBM method was

able to achieve better accuracy than other methods in the literature.

When it comes to intrusion detection datasets, Aegean Wi-Fi Intrusion Detection Dataset (AWID) released by Kolias et al. [17] is a benchmark dataset for wireless NIDS applications. It opened the door for intrusion detection research into wireless networks by providing packet captures of well-known wireless attacks. The initial experiments performed by the authors discovered that impersonation attacks were the hardest attack type to detect. This motivated several other studies to improve the detection rate of impersonation attacks. In one such study, Aminanto and Kim [35] proposed an unsupervised feature extraction method with a deep learning model using Stacked Auto Encoders (SAE). They also introduced an unsupervised feature selection method by using an Artificial Neural Network (ANN). Important features were selected based on the significance of their weights in the ANN model. The model was able to achieve a detection rate of 85% with binary classification using normal and impersonation traffic only. The same authors performed a follow-on study where they used an SAE and K-means clustering to build a fully unsupervised model [36]. This model was able to improve the detection rate of impersonation attacks to 92%.

A more recent study done by Aminanto et al. [8] used a combination of the SAE feature extraction method and the weighted feature selection method to further improve the detection rate of impersonation attacks. Their final model was able to achieve a detection rate of 99.92%. However, all the aforementioned studies focused on improving the detection rate of a single attack type (impersonation) as a binary classification problem. In general, detecting one attack type as opposed to detecting all attack types tends to have higher accuracy.

Several studies have attempted to improve the multiclass detection rate in AWID wireless dataset. For example, Liu and Chung [37] extended the SAE-based feature extraction method to multiclass classification. Similar to the original study [8], they combined the SAE extracted features with the original feature set and ran them through a feature selection step. However, they added two new steps, Principal Component Analysis (PCA) and Clustering to further reduce the feature set. Their best multiclass detection rate was 79%.

Mikhail et al. [13] proposed a semi-boosted ensemble approach to improve the multiclass detection rate. They studied a complex architecture involving both standard and boosted learners. The standard learners were implemented using decision trees and boosted learners were implemented using Adaboost. A binary sub-ensemble was created using five standard learners and five boosted learners. A collection of these sub-ensembles was used to learn a single class using One-vs-Rest class binarization. They also employed a weight matrix to derive the final prediction value. The weights were calculated using twofold cross-validation on the training set. Their final best model was able to achieve an average detection rate of 86%.

Lopez-Martin et al. [38] proposed a framework based on Radial Basis Activation Function Neural Network (RBFNN) architecture. They worked on streamlining the traditional RBFNN architecture, which involves optimizing three types of parameters as three separate tasks, into a single integrated optimization task that can be extended to more complex architectures consisting of multiple hidden layers. They experimented with both supervised learning and reinforcement learning frameworks. However, they reported their results in terms of accuracy as opposed to detection rate. The best results were achieved using RBFNN reinforcement learning framework with an accuracy of 95.5%.

Lei et al. [39] proposed a deep neural network architecture based on Triangle Area Maps (TAM). TAM was used as a mechanism to derive new features using multi-feature correlation. The constructed TAMs were supplied to a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) to extract both spatial and temporal features. Finally, the extracted features were supplied to an Attention network and then a Deep Neural Network (DNN) to make a prediction. The authors argued that previous studies extracted features from CNN and LSTM models serially resulting in information loss. They addressed this issue by extracting features in parallel and running those features through a fusion method. However, their method was tested using only a small portion (20%) of AWID training and test data.

Several new studies used the hold-out method to evaluate their methods by randomly splitting the AWID training dataset into training and test datasets [40], and [41]. As a result, it is unclear how those models would react to unknown attacks that are only present in the test dataset. Furthermore, the studies found in the literature suffer from a low detection rate especially for the wireless AWID dataset.

III. BACKGROUND

A. ENSEMBLE METHODS

Ensemble methods are often used in literature as a way of improving the classification accuracy by aggregating the results of multiple base classifiers into a single predictor. The base classifiers are independently trained and the final prediction is determined by taking a vote among the base classifiers. Several different methods can be used to construct the base classifier. Some commonly used methods are training set sampling, feature set sampling and class label manipulation. The idea behind training set sampling is to use a subset of the original data to train a base classifier. Bagging (or Bootstrapping) and Boosting are two predominantly used ensemble methods that utilize this approach [42] and [43]. The idea behind feature set sampling is to randomly select a subset of features to train each base classifier. Random Forrest is a popular ensemble method that utilizes this approach [44]. In class label manipulation, a multiclass problem is transformed into a set of binary class problems by creating a set of binary classes from the original multiclass labels using a class binarization algorithm. The transformed

binary classes are then used to train the base learners. In this study, we leverage both bootstrapping and One-vs-One class binarization framework to improve the overall accuracy of multiclass classification.

The reason ensemble methods perform better than a single learner is the way it arrives at the final prediction, also known as the voting mechanism. Assuming the base classifiers within the ensemble are independent, an ensemble algorithm will only misclassify a record if at least half of the base classifiers misclassify that same record. The error rate of the ensemble can be calculated using (1),

$$\sum_{i=1+\frac{N}{2}}^{N} \left(\frac{N}{i}\right) \varepsilon^{i} (1-\varepsilon)^{N-i}, \tag{1}$$

where N is the number of base classifiers in the ensemble and ϵ is the error rate of a single classifier. Let's consider an example where an ensemble is constructed using 25 base classifiers and the error rate of a single base classifier is $\epsilon = 0.35$. If the base classifiers are not independent, then the error rate of the ensemble will remain at 0.35 since every record misclassified by the base classifiers will also be misclassified by the ensemble. However, if the base classifiers are independent then the error rate of the ensemble will be reduced to 0.06.

B. CLASS BINARIZATION

Class binarization is a method used to transform a multiclass problem into a set of equivalent binary-class problems by using a set of classifiers often referred to as base classifiers or base learners. While there are many binarization techniques found in the literature [45], [46], [47], the two most prevailing methods are One-vs-Rest (OVR) and One-vs-One (OVO) [48]. In OVR, a K-class classification problem is transformed into a set of K binary-class problems where each base learner attempts to learn one class from all the other classes. Binary sets are created by treating the class of interest as positive samples and the remaining classes as negative samples. For example, consider a dataset with a sample vector X and K number of class labels. Let class label vector be Y where $Y_i \in \{1, \ldots, K\}$ is the class label for sample X_i where i is the sample index. Let C_k be the classifier trained to detect class label k where $k \in \{1, \ldots, K\}$. Then, for each k in $\{1, \ldots, K\}$, 1) a new class label vector $\overline{Y}$ is created where $\overline{Y}_i = Y_i$ if $Y_i = k$ and $\overline{Y}_i = 0$ otherwise; meaning $\overline{Y}_i = 0$ for any other class label except k. 2) $\overline{Y}$ is used to train the corresponding C_k classifier. To make a prediction, the class label belonging to the classifier that reported the highest confidence score is chosen as the winner. Specifically, $\hat{Y}_i = argmax[C_k(X_i)]$.

On the other hand, in OVO, a K-class classification problem is transformed into a set of K(K-1)/2 binary-class problems where each base learner attempts to learn one class from another class. Binary sets are created in pairwise fashion. For example, considering the same dataset as above with a sample vector X and K number of class labels, let L be the number of unique class pairs that can be formed where L is equal to K(K-1)/2. Let $C_{n,m}$ be the classifier trained to detect binary class pair n,m where $n,m \in \{1, \ldots, K\}$. Then, for each binary pair n,m, 1) a new subset $\overline{X}$ is created by selecting the samples where $Y_i = n$ or $Y_i = m$. 2) $\overline{X}$ is used to train classifier $C_{n,m}$. To make a prediction, the class that received the highest number of predictions among L classifiers is chosen as the final prediction. In this study, we propose an adaptation to the traditional OVO method for enhanced attack detection.

IV. METHODOLOGY

The overall system architecture with training and testing steps is shown in Fig. 1. The following sections present detailed information on main components in the overall architecture.

A. FEATURE ENGINEERING

In this section, we present several feature engineering techniques that can be used for any intrusion detection dataset given the corresponding base features are present in the dataset.

1) CATEGORICAL FEATURES WITH HIGH CARDINALITY

Most features in a network intrusion dataset can be directly used by a machine learning algorithm with minimal preprocessing such as normalization and binarization. However, certain common features of network data such as categorical features with high cardinality, if used in their raw form can hurt the model accuracy when deployed to the field. Media Access Control (MAC) address is one such feature that needs special processing. Most prior studies simply converted these hexadecimal numbers into their integer equivalents [8] and [13]. However, MAC address is a globally unique value and training on such a unique value (in both train and test sets) would not be beneficial to a prediction algorithm. In fact, it may overestimate the model performance, especially if the same device (with the same MAC address) was used to unleash the attacks between training and test sets. The model may train to identify attacks coming from a particular MAC address and if that same MAC address was used during testing, it may give a false high detection rate. However, it may fail to achieve such results if the attackers' MAC address was changed, which is always the case once the model is deployed to the field. Instead of using the raw MAC address values, we used them to derive three new features and then removed the original features from the dataset.

In the case of AWID dataset, the original MAC address columns were: wlan.ra (receiver address), wlan.da (destination address), wlan.ta (transmitter address) and wlan.sa (source address). A new binary feature called *ReceiverIsDestination* was created by assigning a value of 1 if wlan.ra is equal to wlan.da and 0 otherwise. This means the intermediate receiver and the final destination addresses are the same. Similarly, another binary feature called *TrasmitterIsSource* was created by assigning a value of 1 if wlan.ta is equal to wlan.sa and 0 otherwise. Which means the intermediate transmitter and the original source addresses are the same. A third

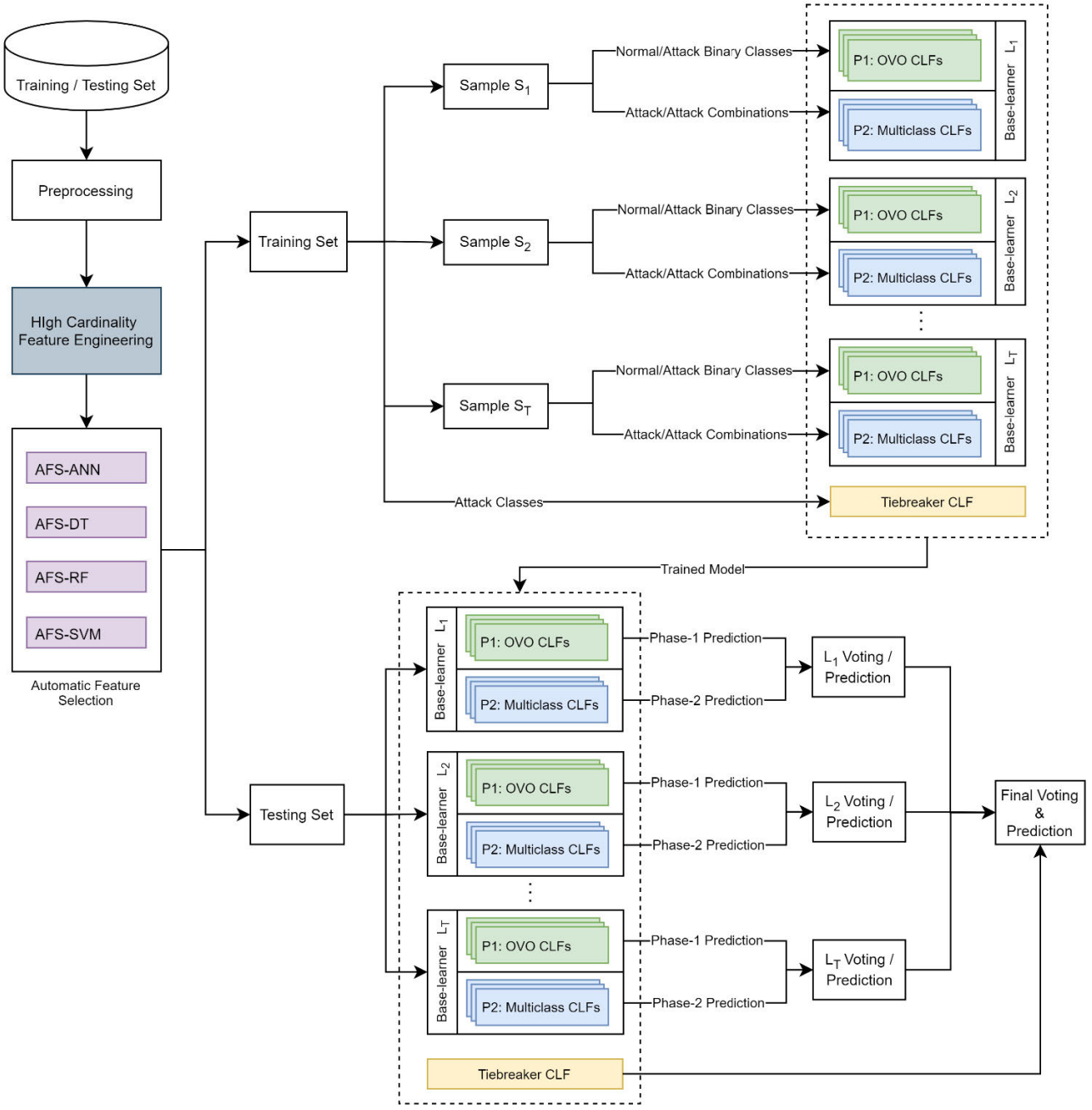

FIGURE 1. Overall architecture of THE-AFS (P1: Phase-1, P2: Phase-2).

binary feature called *Broadcast* was created by assigning a value of 1 if either wlan.ra or wlan.da is set to the broadcast address FF:FF:FF:FF:FF:FF or 0 otherwise.

The Wi-Fi network name (SSID column) is another string feature that needs special preprocessing. Similar to the MAC address columns, these names can vary once deployed to the field. Therefore, we chose to convert the column into binary such that if a SSID name was found, it was assigned a value of 1 and if a SSID name was not found, it was assigned a value of 0. Several other columns such as Wire Equivalent Privacy (WEP) related columns had a very high number of unique values. For example, the Initialization Vector value (in column wlan.wep.iv) used in WEP encryption is randomly generated for every connection and is expected to result in a large number of unique values that may not be beneficial to a prediction model. Therefore, these columns were also converted to binary such that if a value was found it was assigned a value of 1 and if a value was not found, it was assigned a value of 0. The binary value translates to whether a packet was using WEP encryption or not.

2) SEQUENTIAL INTERVAL FEATURES

Network packets produce time series data. As a result, the resulting dataset may contain raw timestamp values

TABLE 1. Problem with model accuracies in AWID dataset when timestamp features are included in the model.

	Predicted Imp.	Predicted Normal
Actual Imp.	1,472	18,607
Actual Normal	46	530,739

	Predicted Imp.	Predicted Normal
Actual Imp.	20,073	6
Actual Normal	15,980	514,805

as features. AWID dataset contained several timestamp columns (frame.time_epoch, frame.time_relative and radiotap.mactime) that needed special handling. In the case of AWID training dataset, there were ∼1.8 million unique timestamp values for the ∼1.8 million packets in the dataset. Therefore, unless new features were derived from the original timestamp value such as, the day of the week, 24H time value, delta time between packets, the raw timestamp values would not be useful to a prediction model.

However, to our surprise, several automatic feature extraction methods [13] and [37] picked one or more timestamp columns as important features that should be included in a model. We performed an extensive study as to why that would be the case and found that timestamp values can illude the model performance due to how the AWID dataset was captured. For example, Table 1 shows the classification results from two models that were trained using the AWID training set and tested using the AWID test set. Both models were filtered to just normal and impersonation traffic for simplicity. Similar to other studies, both models used only 10% of normal traffic during training. The only difference between the two models was the random seed value that was used when sampling the normal traffic. While Model-1 had a very low detection rate of 0.07, Model-2 had a near perfect detection rate of 1. Further analysis showed that the illusion of perfect detection rate was introduced by the timestamp columns. The AWID training dataset was created in a ∼60 minute time period and the test dataset was created in a ∼20 minute time period but they were created back-to-back with a small time gap in between. Therefore, timestamp values such as *frame.time_epoch* will gradually increment from the start of the training set until the end of the test set. After inspecting the Decision Tree for Model-2, we noticed that a decision split was created to classify packets as impersonation when the timestamp value is above a certain value. Since all the timestamp values in the test dataset will always be greater than the timestamp value chosen for the split during training, Model-2 had a very high probability of classifying all the traffic as impersonation. This coupled with other splits in the tree gave Model-2 a very high detection rate but also a relatively high false alarm rate.

Based on this finding, instead of using raw timestamp values, we investigated the possibility of deriving new features that would benefit the prediction accuracy. However, because the training and test datasets were created on the same day within a small time span, transforming timestamp values into more informative features (day of the week or 24H time) did not provide much value. However, the dataset already had a precalculated delta time value as part of the original feature set (*frame.time_delta*), thus we did not have to specifically create it ourselves. To avoid the high variance introduced by raw timestamp values, we simply removed them from the dataset and kept just the delta time values that were derived from the original timestamp features.

B. FEATURE SELECTION

Selecting the most optimal feature set is essential to a machine learning based intrusion detection system because having unnecessary features could lead to several issues as discussed earlier. We implemented a feature selection engine that automatically selects the most important features that correspond to the most significant weights based on four different machine learning algorithms – Decision Tree (DT), Artificial Neural Network (ANN), Random Forrest (RF), and Support Vector Machine (SVM) – as classifiers based on their ability to detect the significance of each input feature using a criterion specific to each classification technique. Each classifier was trained using the entire training dataset, resulting in four candidate feature selection models AFS-DT, AFS-RF, AFS-ANN, and AFS-SVM, each producing a set of features in the order of their weight significance.

In the case of AFS-DT and AFS-RF, we ranked all the features selected by the classifier based on Gini importance and then the most important features were selected based on a predefined threshold. In the case of AFS-ANN, we calculated the importance of each feature by taking the absolute sum of all the weights between features and all the neurons in the first layer of ANN. This is shown in (2).

$$X_j = \sum_{i=1}^{h} \left|\omega_{ij}\right|, \tag{2}$$

where j is the feature index and i is the neuron index in the first layer of ANN. The features with a higher absolute sum were chosen based on a predefined threshold. For AFS-SVM, we selected the most important features using the Recursive Feature Elimination (RFE) process adopted by [8] and [37]. Finally, the two-phased learning algorithm was evaluated using features selected by each candidate model independently.

C. TWO-PHASED ENSEMBLE ARCHITECTURE

The proposed system employs a hybrid multiclass ensemble method consisting of T base learners where each base learner is trained using a random sample (with replacement) drawn from the original dataset. The number of base learners is a hyperparameter supplied to the model. The overall system architecture with training and testing steps is shown in Fig. 1. The model is trained using training data and then tested using previously unseen test data. Both datasets go through the same preprocessing, feature engineering and feature selection steps before being applied to the model. Each base learner is a hybrid of two classification methods that work in stages. The purpose of the first stage is to produce a list of attack

TABLE 2. Classifier combinations in phase-2 using 4 attack classes.

Number of Attack Classes Picked	Number of Classifiers	Attack Class Combinations
2	$\binom{4}{2} = 6$	$CF^{Atk_1-Atk_2}$, $CF^{Atk_1-Atk_3}$, $CF^{Atk_1-Ak_4}$, $CF^{Atk_2-Atk_3}$, $CF^{Atk_2-Atk_4}$, $CF^{Atk_3-Atk_4}$
3	$\binom{4}{3} = 4$	$CF^{Atk_1-Atk_2-Atk_3}$, $CF^{Atk_1-Atk_2-Atk_4}$, $CF^{Atk_2-Atk_3-Atk_4}$, $CF^{Atk_1-Atk_3-Atk_4}$,
4	$\binom{4}{4} = 1$	$CF^{Atk_1-Atk_2-Atk_3-Atk_4}$

TABLE 3. Distribution of attack and normal traffic in AWID training and test datasets.

Attack Type	AWID-CLS-R-Trn Count	AWID-CLS-R-Tst Count
flooding	48,484	8,097
impersonation	48,522	20,079
injection	65,379	16,682
normal	1,633,190	530,785
Total	**1,795,575**	**575,643**

candidates whereas the purpose of the second stage is to narrow that list down to a specific attack class. The number of classifiers in each stage is dependent on the number of classes in the dataset.

The first stage is built using a variation of the OVO method. A typical OVO method is built using $K(K-1)/2$ binary classifiers where K is the number of classes in the dataset. The proposed method is built using the OVO method but with only *normal-attack* binary classifiers. The *attack-attack* binary classifiers are not used in this stage. The result is an OVO method with $(K-1)$ classifiers. For example, a dataset containing five classes, one class being normal and four classes being attack where the attack classes are labeled as $Attack_1$, $Attack_2$, $Attack_3$ and $Attack_4$, the resulting four binary classifiers are: $[CF_1^{Normal-Attack}\, CF_2^{Normal-Attack}, CF_3^{Normal-Attack}\, CF_4^{Normal-Attack}]$. The training process involves drawing a random sample from the original dataset and creating binary subsets containing only the normal class and one attack class. These subsets are then used to train the binary classifiers.

The second stage is built using multiclass classification where multiple sets of classifiers are built using class combinations. The goal of this stage is to narrow down to a specific attack type from a list of attack candidates passed down from the first stage. Therefore, the normal class is not considered in this stage. This concept can be better illustrated by using the previous example with four attack classes. Table 2 lists all attack combinations that can be used to build the classifiers. For a dataset with K classes where one class is normal and $(K-1)$ classes are attacks, the number of classifiers used in the second stage can be expressed using (3).

$$\left(\frac{K-1}{2}\right) + \left(\frac{K-1}{3}\right) + \ldots + \left(\frac{K-1}{K-1}\right) = \sum_{i=2}^{K-1} \left(\frac{K-1}{i}\right), \tag{3}$$

The training process involves drawing a random sample from the original dataset and creating subsets of data containing class combinations. These subsets are then used to train the corresponding multiclass classifier. In the previous example with four attack classes, eleven subsets are created to train the list of classifiers shown in Table 2. The total number of classifiers used in both stages can be expressed using (4). For a system with T number of base learners, the total number of classifiers for the entire model can be calculated using (5). A visual representation of the base learner architecture is shown in Fig. 2. Unlike One-vs-Rest architectures explored in other studies in the literature where the learning algorithm attempts to learn one attack class from all the other attack classes, our One-vs-One two-phased architecture attempts to learn one attack class from another attack class resulting in increased multiclass classification accuracy.

$$(K-1) + \sum_{i=2}^{K-1} \left(\frac{K-1}{i}\right), \tag{4}$$

$$T\left[(\mathrm{K}-1) + \sum_{i=2}^{K-1} \left(\frac{K-1}{i}\right)\right], \tag{5}$$

D. PREDICTION ALGORITHM

Before the algorithm can be explained in detail, it is important to introduce the terminology used throughout this study. Considering a system architecture with T number of base learners, the OVO classifiers are denoted by $CF_{j,c}^{OVO}$; where j is the base learner index and c is the attack class. The multiclass classifiers are denoted by $CF_{j,z}^{MLT}$; where j is the base learner index and z is the set of attack classes that were used to train this classifier. For example, let's consider a system with five base learners and four attack classes labeled as $\{Atk_1, Atk_2, Atk_3 \text{ and } Atk_4\}$. The OVO classifier in base learner index-2 that was trained using the binary classes normal and Atk_3, is denoted by CF_{2,Atk_3}^{OVO}. Similarly, the multiclass classifier in base learner index-2 that was trained using Atk_1, Atk_2 and Atk_3 is denoted by $CF_{2,z=[Atk_1,Atk_2,Atk_3]}^{MLT}$. If the system was trained using N observations, an OVO classifier trained using the i^{th} observation is denoted by $CF^{OVO}(x_i)_{j,c}$. Similarly, a multiclass classifier trained using the i^{th} observation is denoted by $CF^{MLT}(x_i)_{j,z}$. classifier.

As described in Algorithm 1, the prediction algorithm makes an intermediate prediction at each base learner level and then a final prediction by taking a vote among all the base learners. At the base learner level, the voting mechanism goes through two phases. The first phase is designed to get the prediction from OVO classifiers. If all the OVO classifiers predict an observation as normal, then the intermediate prediction for that base learner is classified as normal. The second phase is not used in this scenario. If at most one OVO

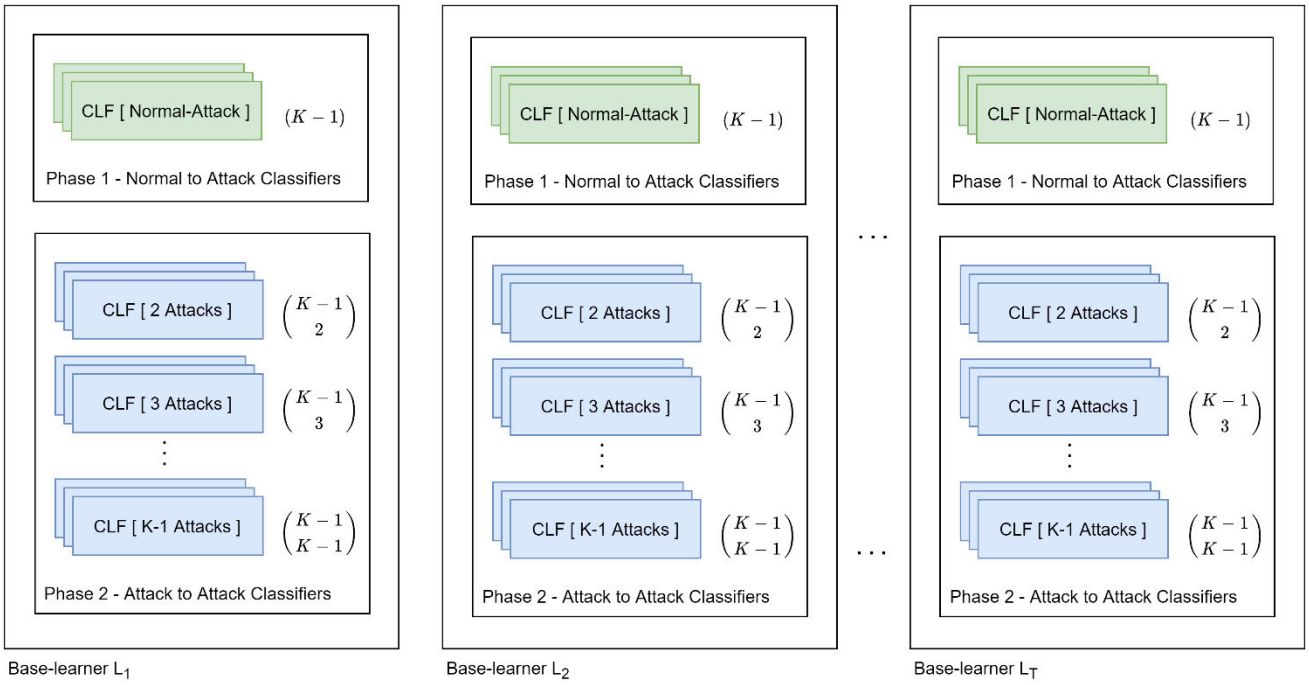

FIGURE 2. Architecture of Two-phased Hybrid Ensemble (THE) learning algorithm.

Algorithm 1 Two-Phased Hybrid Ensemble Learning Algorithm

Let T be total number of BaseLearners in the Ensemble
Let M be min number of BaseLeane rs *required to classify as Attack*
Let K be number of classes
Let $(K-1)$ *be the number of attack classes*
Let CF^{OVO}_{j,Atk_a} *be Normal to Attack OVO classifier for Attack=a*
Let $CF^{MLT}_{j,Phase1}$ be an Attack combination multiclass classifier
LetL_j be the prediction for base leaner in j^{th} index

Base Learner Prediction

For BaseLearner $j = 1$ *to* T:
  # Get Phase1 Prediction
  For Attack $a = 1$ *to*$(K-1)$:
    # Is the OVO classifier trained to detect Atk_a predicting as Atk_a?
    If $CF^{OVO}_{j,Atk_a} == Atk_a$:
      $Phase1\,[] += Atk_a$
  if count$(Phase1) == 0$:
    $L_j =$ Normal # None of the OVO classifiers predicted as Attack
  else count $(Phase1) == 1$:
    $L_j = Phase1[]$ # Only 1 OVO classifier predicted as Attack
  # Get Phase2 Prediction
  else:
    # More than 1 OVO classifier predicted as Attack.
    # Get final Attack type using the corresponding multiclass classifier.
    $L_j = CF^{MLT}(x_i)_{j,Phase1}$

Ensemble Prediction L_1- L_T

Aggregate predictions from base learners into a dictionary of attack counts
$\text{AtkCounts}[Atk_a] = \sum_{j=1}^{T} I(L_j = Atk_a)$, for each attack a
Get highest predicted attack label
$Atk_a = argmax(AtkCounts)$
Predict as Atk_a if the highest attack count exceeds a predefined threshold

$$Y = \begin{cases} Atk_a; & if\,(AtkCounts\,[Atk_a] > M \\ Normal; & else \end{cases}$$

classifier predicts an observation as *Attack*, then that attack type is assigned as the prediction value for that base learner. Again, the second phase is not used in this scenario. Lastly, if at least two OVO classifiers predict an observation as *Attack*, then those prediction values are passed to the second phase to determine the final prediction. Accumulating the attack predictions from OVO classifiers into a list is shown in (6).

$$Phase1(x_i)\,[] \overset{Atk_a}{\longleftarrow} \theta\left(CF^{OVO}(x_i)_{j,Atk_a} = Atk_a\right), \quad (6)$$

where $a = 1, 2, 3, \ldots, (K-1)$ and θ returns the attack label when the OVO binary prediction is attack.

As mentioned earlier, the second phase prediction is performed using the multiclass classifier that was trained on attack types passed down from the first phase. For example, if $Phase1 = [Atk_1, Atk_2, Atk_4]$, then $CF^{MLT}(x_i)_{j,z=[Atk_1,Atk_2,Atk_4]}$ is used to get the prediction value for the second phase which is also the final prediction value for that base learner. The base learner prediction algorithm is stated in (7)

$$L_j(x_i) = \begin{cases} Normal, & if\,CF^{OVO}(x_i)_{j,Atk_a} = Normal, \\ Atk_a, & if \left\langle \begin{matrix} CF^{OVO}(x_i)_{j,Atk_a} = Atk_a AND \\ All other\, CF^{OVO}(x_i)_j = Normal \end{matrix} \right\rangle \\ Atk_a, & if \left\langle CF^{MLT}(x_i)_{j,Phase1} = Atk_a \right\rangle, \end{cases} \quad (7)$$

for all a where $a = 1, 2, 3, \ldots, (K-1)$. The first condition in (7) represents the case where base learner prediction is normal when all the *normal-attack* OVO classifiers predict an

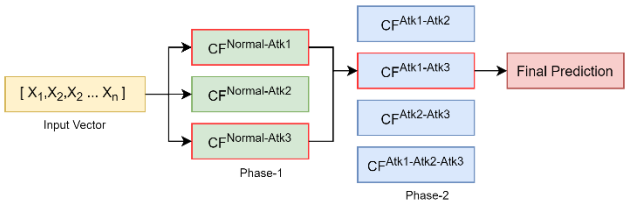

FIGURE 3. Voting example using three attack classes. Phase-1 predicts as Attack1 and Attack3. Therefore, $CF^{Atk1-Atk3}$ is selected in Phase-2 to get the final prediction.

observation as normal. The second condition represents the case where base learner prediction is Atk_a if only one OVO classifier predicts an observation as attack, but all the other OVO classifiers predict the observation as normal. The third condition represents the case where base learner prediction is Atk_a if the first phase prediction list is passed to the second phase and the multiclass classifier predicts the observation as attack. An example prediction flow with three attack classes is shown in Fig. 3. For input vector $[X_1, X_2, X_3 \ldots X_n]$, Phase-1 OVO classifiers trained with $Normal - Atk_2$ predicts as normal but both $Normal - Atk_1$ and $Normal - Atk_3$ predicts as Atk_1 and Atk_3 respectively. Therefore, Atk_1 and Atk_3 is passed down to Phase-2, at which point the multiclass classifier trained with $Atk_1 - Atk_3$ is used to get the final prediction.

Once each base learner has a prediction value, the next step in the algorithm is to derive a final prediction value for the observation. The algorithm creates a dictionary of attack counts, where the attack type is the key and the number of base learner predictions for that attack type is the value. This is shown in (8).

$$AtkCounts\,[Atk_a] = \sum_{j=1}^{T} I(L_j\,(x_i) = Atk_a), \tag{8}$$

where $a = 1, 2, 3, \ldots, (K-1)$ and the identity function I return 1 when base learner j prediction is equal to Atk_a. Next, the algorithm finds the attack label with the highest attack count using (9).

$$Atk_{Label} = argmax(AtkCounts), \tag{9}$$

In rare cases where the highest reported attack count is a tie between two or more attack classes, a tiebreaking classifier trained just on attack classes (excluding normal class) is used to break the tie and produce a single attack class. Lastly, the final prediction value is determined based on whether the highest reported attack count exceeds a predefined threshold. In other words, the prediction is classified as Atk_a if at least M base learners classify a sample as Atk_a. Otherwise, the prediction is classified as normal. The value of M is a hyperparameter supplied to the model and it is determined empirically in this study. The final prediction equation is shown in (10).

$$Y\,(x_i) = \begin{cases} Atk_{Label}; & if\,(AtkCounts\,[Atk_{Label}] > M \\ Normal; & else, \end{cases} \tag{10}$$

The complete voting and prediction algorithm at both base learner and final ensemble level is shown in Algorithm 1.

V. EVALUATION AND DISCUSSION

A. DATASET

The proposed methodology was tested on the Aegean Wi-Fi Intrusion Detection Dataset (AWID) which is widely used in the literature as a benchmark dataset for Wi-Fi intrusion detection. It was created in a lab environment by emulating a Wi-Fi setup consisting of both stationary and mobile clients found in a typical wireless network. The stationary clients included a desktop PC and a smart TV. The mobile clients included two laptops, two smartphones and one tablet. One mobile client (laptop) was used to generate intrusions by leveraging commonly used penetration tools. The authors emulated and captured 15 different attack types using this setup. Then they grouped those 15 attacks into 3 attack classes based on the similarities of the attack method, namely - flooding, injection and impersonation. The 3 attack classes plus the normal class make this a multiclass classification problem with 4 imbalanced classes. Each row in the dataset is a network packet on the wireless network, and 154 network attributes (columns) were captured per row. They released two separate datasets, one for training purposes and one for testing purposes. The two datasets were not created by splitting a single dataset, but rather using two separate capturing sessions, which makes the data more realistic. Another important fact is that the testing set contains attacks that are not found in the training set. This is where traditional learning algorithms perform poorly with this dataset because the learning algorithms usually fail to identify previously unseen attacks. However, it is an important aspect of an intrusion detection problem because an effective NIDS has to be capable of detecting new or unknown attacks that evolve over time, often times as variations of known attacks. For this study, we used the AWID-CLS-R-Trn dataset to train the model and AWID-CLS-R-Tst dataset to test it. The heavily imbalanced distribution of the attack and normal classes between the training and testing datasets is shown in Table 3.

To broaden the testing efforts of our proposed methodology and to prove that it can be generalized to both wired and wireless applications, we brought in a second dataset NSL-KDD [16] which is commonly used in the literature as an intrusion detection dataset for wired Ethernet. The attacks in this dataset were emulated in a military environment where they unleashed 39 different attack types while operating the network as if it were a true military Local Area Network (LAN). The attack types were grouped into 4 categories, namely – dos, r2l, u2r and probing. The 4 attack classes

TABLE 4. Distribution of attack and normal traffic in NSL-KDD dataset.

Attack Type	Count
dos	45,927
probe	11,656
r2l	995
u2r	52
normal	67,343
Total	125,973

TABLE 5. Experiment results based on feature selection method.

Dataset	Feature Selection Method	DR	FAR
AWID	THE-AFS-ANN	0.9295	0.0145
	THE-AFS-DT	0.9251	0.0153
	THE-AFS-RF *	0.9264	**0.0140**
	THE-AFS-SVM	**0.9314**	0.0144
NSL-KDD	THE-AFS-ANN	**0.9494**	0.0006
	THE-AFS-DT *	0.9431	**0.0005**
	THE-AFS-RF	0.9489	0.0006
	THE-AFS-SVM	0.9354	0.0006

* The final candidate selected for each dataset

plus the normal class makes this a multiclass classification problem with 5 imbalanced classes. Each row of data in the dataset is a network packet in the LAN network represented by 41 attributes (columns). Following the same approach as the previous study [13], we used the KDDTrain+ dataset with a random 50/50 split for training and test data. The imbalanced distribution of the attack and normal classes in KDDTrain+ is shown in Table 4.

B. DATASET PRE-PROCESSING

The AWID dataset required several pre-processing steps to transform the data into a format that can be used by a machine learning algorithm. Several columns had missing values indicated by a string value of "?". Prior studies replaced these missing values with an integer value of 0 [8] or -1 [13]. In our study, we chose to replace them with a -1 so that it does not collide with an actual 0 value in a binary column. Several columns had hexadecimal values which needed to be converted to their integer equivalent. Furthermore, to address the class imbalance problem, we experimented with several techniques such as Random Under Sampling (RUS), Random Over Sampling (ROS) and Synthetic Minority Over-sampling Technique (SMOTE) [49]. We adopted RUS because it was able to achieve the best performance. Following prior studies [8], the dataset was balanced by randomly selecting 10% of the majority class (normal traffic) instances.

C. SYSTEM EVALUATION

The evaluation metrics must be carefully chosen because of the highly imbalanced nature of the dataset where normal packets vastly outnumber the attack packets. Metrics such as accuracy can produce misleading results because the majority class can easily overshadow the minority class giving a false sense of model performance. In our study, we used two of the most well referenced evaluation metrics for class imbalanced data in intrusion detection systems: Detection Rate (DR) and False Alarm Rate (FAR) [50]. DR which is also called the True Positive Rate (TPR) is defined as the percentage of packets that are correctly classified for a given class. DR is calculated using (11). For an attack class, True Positive (TP) is defined as the number of attack packets correctly classified as attack and False Negative (FN) is defined as the number of attack packets incorrectly classified as normal. FAR which is also known as False Positive Rate (FPR) is defined as the percentage of packets that are incorrectly classified for a given class. FAR is calculated using (12). For an attack class, FP is defined as the number of normal packets incorrectly classified as attack and TN is defined as the number of normal packets correctly classified as normal packets. The goal of a NIDS should be to maximize DR and minimize FAR.

$$Detection\ Rate\ (DR) = TPR = \frac{TP}{TP + FN}, \tag{11}$$

$$False\ Alarm\ Rate\ (FAR) = FPR = \frac{FP}{FP + TN}, \tag{12}$$

D. EXPERIMENTAL RESULTS

The proposed model was evaluated using all four feature selection methods discussed earlier. Our preliminary studies showed that around 30 and 38 features produced the best detection rates for AWID and NSL-KDD datasets respectively. This is consistent with findings by [13]. With each feature selection method, the model hyperparameters were iterated to find the most optimized model that performed best with each dataset. The three model hyperparameters were: sample size denoted by S, number of base learners denoted by T, and the minimum percentage of base learners required to classify a packet as attack which is denoted by M. The sample size S was iterated from 10% to 90% in 10% increments. For each sample size S, the model was tested using 5 different T values (T = 10, 25, 50, 75, and 100 learners). For each S and T value, M was iterated from 1 to T in 10% increments.

The best result for each feature selection method is shown in Table 5. For the AWID dataset, SVM based feature selection achieved the highest Detection Rate (DR) while RF based feature selection achieved the lowest False Alarm Rate (FAR). We picked RF based feature selection as the final candidate (THE-AFS-RF) because of achieving a low FAR of **0.0140** while maintaining a high DR of **0.9264**. The best THE-AFS-RF result was achieved using the model parameters: T=75, S=30%, M=1. An M value of 1 indicates that for a packet to be normal, all base learners within the ensemble must classify that packet as normal. On the other hand, a single base learner classifying a packet as $Attack_a$ will result in

TABLE 6. Evaluation of proposed method with other leading approaches using AWID dataset.

Study	Norm.	Flo.	Imp.	Inj.	Ave.	TPR/FPR [Rank]
THE-AFS-RF *	0.9912	**0.7157**	**0.9989**	**0.9999**	**0.9264**	**66.17**
	0.0474	0.0022	0.0065	**0.0000**	**0.0140**	
SemBst [13]	0.9610	0.7014	0.7912	0.9999	0.8634	24.81
	0.0981	0.0044	0.0364	0.0003	0.0348	
MltEns [51]	**0.9999**	0.6803	0.0732	0.9997	0.6883	5.83
	0.4726	**0.0000**	**0.0000**	**0.0000**	0.1181	
J48 [17]	0.9996	0.6846	0.0641	**0.9999**	0.6871	5.81
	0.4726	**0.0000**	0.0001	**0.0000**	0.1183	
PRelu [52]	0.9980	0.5748	0.9850	0.8272	0.8463	-
	-	-	-	-	-	
SAE [35]	0.9986	0.3155	0.6518	0.9996	0.7414	-
	-	-	-	-	-	

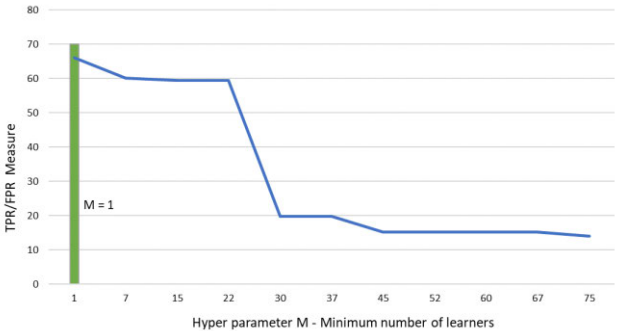

FIGURE 4. AWID model performance when T = 75, S = 30% and M is varied from 1 to 75. The best model performance was observed when M=1.

the entire ensemble classifying the packet as $Attack_a$. This behavior can be further illustrated by plotting the TPR/FPR measure for several T values as shown in Fig. 4. As mentioned before, the highest TPR/FPR value when M=1 indicates that the attacks found in AWID dataset are much harder to detect and even a single learner detecting a packet as attack tends to have a high probability of being an actual attack. The low FPR value further solidifies this argument. Another observation is that TPR/FPR value has a drop when M is around 25. Further investigation revealed that this drop is caused by the failure to detect impersonation attacks. As M increased, more impersonation attacks were predicted as normal traffic and as a result, the TPR of impersonation decreased and FPR of normal increased with a net result of decreased overall TPR/FPR. This finding is consistent with other studies in the literature which state impersonation attacks were the hardest to detect [8] and [17].

For the NSL-KDD dataset, ANN based feature selection achieved the highest Detection Rate (DR) while DT based feature selection achieved the lowest False Alarm Rate (FAR). We picked DT based feature selection as the final candidate (THE-AFS- DT) because of achieving a low FAR

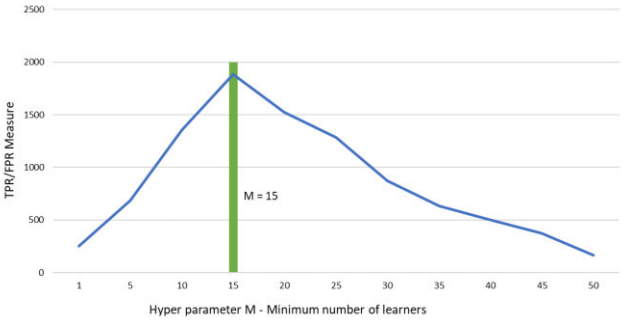

FIGURE 5. NSL-KDD model performance when T = 50, S = 70% and M is varied from 1 to 50. The best model performance was observed when M=15.

TABLE 7. Evaluation of proposed method with other leading approaches using NSL-KDD dataset.

Study	Norm.	DoS	Probe	R2L	U2R	Ave.	TPR/FPR [Rank]
THE-AFS-DT *	0.9976	0.9996	0.9988	0.9607	0.7586	0.9431	**1,886**
	0.0012	**0.0002**	0.0006	0.0004	0.0003	**0.0005**	
Semi-Bst [13]	0.9987	0.9996	0.9959	0.9598	0.6000	0.9108	1,822
	0.0015	0.0003	**0.0003**	**0.0001**	0.0005	**0.0005**	
MRK-ELM [53]	0.9989	0.9996	0.9742	0.9494	0.6287	0.9102	1,517
	0.0019	**0.0002**	**0.0003**	0.0005	0.0001	0.0006	
OVA-SVM [54]	0.9960	0.9990	0.9920	0.9870	0.7390	0.9426	673
	0.0040	0.0010	0.0010	0.0010	**0.0000**	0.0014	
EID3 [55]	**0.9990**	0.9990	0.9980	**0.9997**	0.9980	**0.9987**	587
	0.0008	0.0003	0.0039	0.0022	0.0012	0.0017	
ID3 [55]	0.9990	0.9990	0.9970	0.9350	0.4910	0.8842	246
	0.0010	0.0003	0.0055	0.0098	0.0015	0.0036	
NB/DT [56]	0.9984	0.9976	0.9975	0.9935	0.9947	0.9963	74
	0.0005	0.0003	0.0028	0.0622	0.0010	0.0134	
ANN [57]	0.9880	0.9380	0.8980	0.9190	0.8660	0.9218	65
	0.0655	0.0004	0.0014	0.0028	0.0005	0.0141	
Hybrid [58]	0.9899	0.9991	**0.9989**	0.9937	**0.9989**	0.9961	58
	0.0150	0.0010	0.0150	0.0340	0.0210	0.0172	

* Proposed Study

of **0.0005** while maintaining a high DR of **0.9431**. The best THE-AFS-DT result was achieved using the model parameters: T=50, S=70%, M=15. Unlike the AWID dataset, the NSL-KDD dataset was producing a high FAR when M is closer to 1, resulting in a lower TPR/FPR score. On the other hand, it was producing a lower TPR when M is closer to 50, again resulting in a lower TPR/FPR with higher M values. Another way to look at this behavior is, when M=1, an incorrect prediction by a single classifier (within the ensemble) will result in the entire ensemble making an incorrect prediction, hence the high FPR. As M increases, more classifiers are needed to detect an attack and as a result, it becomes harder for the entire ensemble to detect an attack, hence, the low

TABLE 8. Evaluation of proposed method with other studies using AWID dataset and multiclass classification. results presented in terms of accuracy and f1 scores.

Study	Accuracy Overall	Accuracy Macro	F1 Macro	F1 Micro
THE-AFS-RF *	**0.9879**	**0.9939**	**0.9190**	**0.9879**
RL-NIDS [41]	0.9572	-	0.7027	0.9447
DRL+RBFNN [38]	0.9540	-	-	0.938
SBN [13]	0.9526		0.8209	-

* Proposed Study

TPR. As shown in Fig. 5, this behavior produced a downward parabolic curve for the NSL-KDD dataset.

Next, we compared our experimental results to other leading approaches published in the literature. We used the same Macro-Average TPR/FPR ranking criteria adopted by [13]. The AWID comparison results are shown in Table 6. Compared to other approaches, our best performing model THE-AFS-RF was not only able to achieve the highest TPR but also the lowest FPR, resulting in the highest TPR/FPR rank of **66.17**. The NSL-KDD comparison results are shown in Table 7. The best performing model THE-AFS-DT was able to maintain the same low FPR value of 0.005 as the previous leading study but increased the TPR from 0.9108 to 0.9431 resulting in a slightly higher TPR/FPR rank of **1,886**. Several recent studies reported their results in terms of overall accuracy and F1 scores for the AWID dataset. For completeness, we compared our results with some of the newer studies as well and the results are presented in Table 8.

VI. CONCLUSION

In this study, we proposed THE-AFS, a machine learning based Network Intrusion Detection System (NIDS) with Two-Phased Ensemble learning and Automatic Feature Selection for detecting various known and unknown attacks in high dimensional network data. The proposed automatic feature selection engine can be used by any high dimensional network dataset to identify the most significant features needed for multiclass attack classification. Furthermore, we identified drawbacks of existing feature selection methods for network data and introduced new feature engineering techniques that improve attack detection. The proposed framework is constructed using a hybrid of two learning algorithms. While there were many One-vs-Rest based learning algorithms found in the literature, it was evident that not enough research was done on One-vs-One based ensemble learning algorithms for intrusion detection. Our two-phased architecture built with One-vs-One framework is capable of learning one attack type from another attack type, resulting in higher multiclass classification accuracy. The proposed framework was tested on both wired and wireless networks using two well referenced datasets and the results were compared against other leading studies in the literature. THE-AFS-DT model performed best for the wired application with a detection rate of **0.9431** and a false alarm rate of **0.0005**. THE-AFS-RF model performed best for the wireless application with a detection rate of **0.9314** and a false alarm rate of **0.0144**. The wireless application outperformed other leading studies in the literature that attempted to build a generalized model that works in both wired and wireless applications.

REFERENCES

[1] F. Obite, E. T. Jaja, G. Ijeomah, and K. I. Jahun, "The evolution of Ethernet Passive Optical Network (EPON) and future trends," *Optik*, vol. 167, pp. 103–120, Aug. 2018.
[2] B. Bellalta, L. Bononi, R. Bruno, and A. Kassler, "Next generation IEEE 802.11 wireless local area networks: Current status, future directions and open challenges," *Comput. Commun.*, vol. 75, pp. 1–25, Feb. 2016.
[3] E. J. Oughton, W. Lehr, K. Katsaros, I. Selinis, D. Bubley, and J. Kusuma, "Revisiting wireless internet connectivity: 5G vs Wi-Fi 6," *Telecommun. Policy*, vol. 45, no. 5, Jun. 2021, Art. no. 102127.
[4] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the Internet of Things: A comprehensive investigation," *Comput. Netw.*, vol. 160, pp. 165–191, Sep. 2019.
[5] B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. De Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017.
[6] R. Mitchell and I.-R. Chen, "A survey of intrusion detection in wireless network applications," *Comput. Commun.*, vol. 42, no. 3, pp. 1–23, Apr. 2014.
[7] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, 2013.
[8] M. E. Aminanto, R. Choi, H. C. Tanuwidjaja, P. D. Yoo, and K. Kim, "Deep abstraction and weighted feature selection for Wi-Fi impersonation detection," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 3, pp. 621–636, Mar. 2018.
[9] G. V. Trunk, "A problem of dimensionality: A simple example," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 3, pp. 306–307, Jul. 1979.
[10] M. Akbanov, V. G. Vassilakis, and M. D. Logothetis, "Ransomware detection and mitigation using software-defined networking: The case of WannaCry," *Comput. Electr. Eng.*, vol. 76, pp. 111–121, Jun. 2019.
[11] C. Adams, "Learning the lessons of WannaCry," *Comput. Fraud Secur.*, vol. 2018, no. 9, pp. 6–9, Jan. 2018.
[12] A. Cooke, K. Renaud, D. Spence, and C. Tankard, "US authorities recover most of colonial pipeline ransom," *Netw. Secur.*, vol. 2021, no. 6, pp. 1–2, 2021. [Online]. Available: https://www.magonlinelibrary.com/doi/full/10.1016/S1353-4858%2821%2900057-X
[13] J. W. Mikhail, J. M. Fossaceca, and R. Iammartino, "A semi-boosted nested model with sensitivity-based weighted binarization for multi-domain network intrusion detection," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 3, pp. 1–27, May 2019.
[14] Y. Zhou, T. A. Mazzuchi, and S. Sarkani, "M-AdaBoost-A based ensemble system for network intrusion detection," *Exp. Syst. Appl.*, vol. 162, Dec. 2020, Art. no. 113864.
[15] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016.
[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[17] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st. Quart., 2016.
[18] B. A. Tama and S. Lim, "Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100357.
[19] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *J. Netw. Comput. Appl.*, vol. 66, pp. 1–16, May 2016.
[20] Y. Zhou, G. Cheng, S. Jiang, and M. Dai, "Building an efficient intrusion detection system based on feature selection and ensemble classifier," *Comput. Netw.*, vol. 174, Jun. 2020, Art. no. 107247.
[21] M. N. Adnan and M. Z. Islam, "Forest PA: Constructing a decision forest by penalizing," *Exp. Syst. Appl.*, vol. 89, pp. 389–403, Dec. 2017.
[22] N. F. Haq, A. R. Onik, and F. M. Shah, "An ensemble framework of anomaly detection using Hybridized Feature Selection Approach (HFSA)," in *Proc. SAI Intell. Syst. Conf. (IntelliSys)*, Nov. 2015, pp. 989–995.

[23] A. A. Aburomman and M. B. I. Reaz, "A survey of intrusion detection systems based on ensemble and hybrid classifiers," *Comput. Secur.*, vol. 65, pp. 135–152, Mar. 2017.
[24] G. Kumar, K. Thakur, and M. R. Ayyagari, "MLEsIDSs: Machine learning-based ensembles for intrusion detection systems—A review," *J. Supercomput.*, vol. 76, no. 11, pp. 8938–8971, Nov. 2020.
[25] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K. Kwak, "An enhanced anomaly detection in web traffic using a stack of classifier ensemble," *IEEE Access*, vol. 8, pp. 24120–24134, 2020.
[26] Y. Cheng, Y. Xu, H. Zhong, and Y. Liu, "Leveraging semisupervised hierarchical stacking temporal convolutional network for anomaly detection in IoT communication," *IEEE Internet Things J.*, vol. 8, no. 1, pp. 144–155, Jan. 2021.
[27] M. Milliken, Y. Bi, L. Galway, and G. Hawe, "Multi-objective optimization of base classifiers in StackingC by NSGA-II for intrusion detection," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.
[28] Y. Wang, Y. Shen, and G. Zhang, "Research on intrusion detection model using ensemble learning methods," in *Proc. 7th IEEE Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Aug. 2016, pp. 422–425.
[29] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3854–3861.
[30] S. A. Ludwig, "Intrusion detection of multiple attack classes using a deep neural net ensemble," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–7.
[31] K. Li, G. Zhou, J. Zhai, F. Li, and M. Shao, "Improved PSO_AdaBoost ensemble algorithm for imbalanced data," *Sensors*, vol. 19, no. 6, p. 1476, Mar. 2019.
[32] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Nov. 1995, pp. 1942–1948.
[33] X. Li, M. Zhu, L. T. Yang, M. Xu, Z. Ma, C. Zhong, H. Li, and Y. Xiang, "Sustainable ensemble learning driving intrusion detection model," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 4, pp. 1591–1604, Jul./Aug. 2021.
[34] M. H. L. Louk and B. A. Tama, "Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system," *Exp. Syst. Appl.*, vol. 213, Mar. 2023, Art. no. 119030.
[35] M. E. Aminanto and K. Kim, "Detecting impersonation attack in WiFi networks using deep learning approach," in *Proc. WISA*, 2016, pp. 136–147.
[36] M. E. Aminanto and K. Kim, "Improving detection of Wi-Fi impersonation by fully unsupervised deep learning," in *Proc. WISA*, 2017, pp. 212–223.
[37] J. Liu and S. S. Chung, "Automatic feature extraction and selection for machine learning based intrusion detection," in *Proc. IEEE SmartWorld, Ubiquitous Intell. Comput., Adv. Trusted Comput., Scalable Comput. Commun., Cloud Big Data Comput., Internet People Smart City Innov. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, Aug. 2019, pp. 1400–1405.
[38] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas, and B. Carro, "Network intrusion detection based on extended RBF neural network with offline reinforcement learning," *IEEE Access*, vol. 9, pp. 153153–153170, 2021.
[39] S. Lei, C. Xia, Z. Li, X. Li, and T. Wang, "HNN: A novel model to study the intrusion detection based on multi-feature correlation and temporal-spatial analysis," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 4, pp. 3257–3274, Oct. 2021.
[40] R. Kumar, A. Malik, and V. Ranga, "An intellectual intrusion detection system using hybrid hunger games search and remora optimization algorithm for IoT wireless networks," *Knowl.-Based Syst.*, vol. 256, Nov. 2022, Art. no. 109762.
[41] W. Wang, S. Jian, Y. Tan, Q. Wu, and C. Huang, "Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions," *Comput. Secur.*, vol. 112, Jan. 2022, Art. no. 102537.
[42] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[43] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. 13th Int. Conf. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
[44] T. K. Ho, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, Aug. 1995, pp. 278–282.
[45] B. Quost and S. Destercke, "Classification by pairwise coupling of imprecise probabilities," *Pattern Recognit.*, vol. 77, pp. 412–425, May 2018.
[46] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," 1995, *arXiv: 9501101*.
[47] J. Furnkranz, "Pairwise classification as an ensemble technique," in *Proc. 13th Eur. Conf. Mach. Learn.*, 2002, pp. 97–110.
[48] T. M. Khoshgoftaar, K. Gao, and N. H. Ibrahim, "Evaluating indirect and direct classification techniques for network intrusion detection," *Intell. Data Anal.*, vol. 9, no. 3, pp. 309–326, Jun. 2005.
[49] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jul. 2018.
[50] O. Y. Al-Jarrah, O. Alhussein, P. D. Yoo, S. Muhaidat, K. Taha, and K. Kim, "Data randomization and cluster-based partitioning for botnet intrusion detection," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1796–1806, Aug. 2016.
[51] B. Alotaibi and K. Elleithy, "A majority voting technique for wireless intrusion detection systems," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf. (LISAT)*, Apr. 2016, pp. 1–6.
[52] V. L. L. Thing, "IEEE 802.11 network anomaly detection and attack classification: A deep learning approach," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2017, pp. 1–6.
[53] J. M. Fossaceca, T. A. Mazzuchi, and S. Sarkani, "MARK-ELM: Application of a novel multiple kernel learning framework for improving the robustness of network intrusion detection," *Exp. Syst. Appl.*, vol. 42, no. 8, pp. 4062–4080, May 2015.
[54] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, 2016.
[55] V. Jaiganesh, S. Mangayarkarasi, and P. Sumathi, "An efficient algorithm for network intrusion detection system," *Int. J. Comput. Appl.*, vol. 90, no. 12, pp. 12–16, Mar. 2014.
[56] D. M. Singh, N. Harbi, and M. Z. Rahman, "Combining naive Bayes and decision tree for adaptive intrusion detection," *Int. J. Netw. Secur. Appl.*, vol. 2, no. 2, pp. 12–25, Apr. 2010.
[57] A. Sharma, I. Manzoor, and N. Kumar, "A feature reduced intrusion detection system using ANN classifier," *Exp. Syst. Appl.*, vol. 88, pp. 249–257, Dec. 2017.
[58] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 31, no. 4, pp. 541–553, Oct. 2019.

ASANKA KAVINDA MANANAYAKA received the B.S. degree in electrical engineering from Louisiana State University, Louisiana, USA, in 2007, and the M.S. degree in computer engineering from Cleveland State University, OH, USA, in 2014, where he is currently pursuing the Ph.D. degree. He was a Global Leader of industrial automation and information with Rockwell Automation, USA, a fortune 500 company, where he has been with the company, since 2008, and currently works as a Product Security Engineer responsible for the security of industrial automation controllers. His current research interests include cyber security, network intrusion detection with machine learning, and big data analytics.

SUN SUNNIE CHUNG received the M.S. and Ph.D. degrees in computer science from Case Western Reserve University, Cleveland, USA, in 1999 and 2006, respectively. She was a Research Associate of the research and development of optimizer of big data analytics systems with Teradata Corporation (Previously in AT&T), Silicon Valley, CA, USA, from 2007 to 2012, and the research and development of optimizer of enterprise data warehouse systems with NCR (Part of IBM Now), from 2006 to 2007. She is currently an Associate Professor of practice with the Department of Electrical Engineering and Computer Science, Cleveland State University. She is also leading five research groups with the Big Data Analytics and Cyber Security Laboratory, focusing on text analysis with natural language processing (NLP), context aware medical question answering system, machine learning-based network intrusion detection systems, cyber security and privacy in cloud-based the IoT applications, and real time opinion analysis of social networks. She holds numerous publications in prestigious journals and conferences and serves as a review committee member of prestigious IEEE Journals and ACM conferences.