Received December 22, 2020, accepted January 22, 2021, date of publication February 3, 2021, date of current version February 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3056614

Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset

ZIADOON KAMIL MASEER[1], ROBIAH YUSOF[1], NAZRULAZHAR BAHAMAN[1], SALAMA A. MOSTAFA[2], AND CIK FERESA MOHD FOOZY[2]
[1]Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Malacca 76100, Malaysia
[2]Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Batu Pahat 86400, Malaysia

Corresponding author: Salama A. Mostafa (salama@uthm.edu.my)

This work was supported by the Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme under Grant FRGS/1/2018/ICT04/UTHM/03/4.

ABSTRACT An intrusion detection system (IDS) is an important protection instrument for detecting complex network attacks. Various machine learning (ML) or deep learning (DL) algorithms have been proposed for implementing anomaly-based IDS (AIDS). Our review of the AIDS literature identifies some issues in related work, including the randomness of the selected algorithms, parameters, and testing criteria, the application of old datasets, or shallow analyses and validation of the results. This paper comprehensively reviews previous studies on AIDS by using a set of criteria with different datasets and types of attacks to set benchmarking outcomes that can reveal the suitable AIDS algorithms, parameters, and testing criteria. Specifically, this paper applies 10 popular supervised and unsupervised ML algorithms for identifying effective and efficient ML–AIDS of networks and computers. These supervised ML algorithms include the artificial neural network (ANN), decision tree (DT), k-nearest neighbor (k-NN), naive Bayes (NB), random forest (RF), support vector machine (SVM), and convolutional neural network (CNN) algorithms, whereas the unsupervised ML algorithms include the expectation-maximization (EM), k-means, and self-organizing maps (SOM) algorithms. Several models of these algorithms are introduced, and the turning and training parameters of each algorithm are examined to achieve an optimal classifier evaluation. Unlike previous studies, this study evaluates the performance of AIDS by measuring the true positive and negative rates, accuracy, precision, recall, and F-Score of 31 ML-AIDS models. The training and testing time for ML-AIDS models are also considered in measuring their performance efficiency given that time complexity is an important factor in AIDSs. The ML-AIDS models are tested by using a recent and highly unbalanced multiclass CICIDS2017 dataset that involves real-world network attacks. In general, the k-NN-AIDS, DT-AIDS, and NB-AIDS models obtain the best results and show a greater capability in detecting web attacks compared with other models that demonstrate irregular and inferior results.

INDEX TERMS Cyberattacks, intrusion detection system, machine learning, supervised and unsupervised learning.

I. INTRODUCTION

As more platforms and applications are being connected to networks, data become increasingly vulnerable to malicious attacks. Accordingly, network security has become an important research topic. Using an intrusion detection system (IDS) is a well-known approach for protecting computer networks [1]. Two popular types of IDS, namely, network- (NIDS) and host-based IDS (HIDS), have been adopted in practice. NIDS monitors network traffic and detects any malicious activity in the network by analyzing the activities of end users [2].

IDS applies two types of detection methods, namely, signature- and anomaly-based methods. On the one hand, signature-based IDS (or HIDS) detects attacks by identifying patterns (i.e., signatures) in IDS [3]. While this method can easily detect known malware and attacks based on their

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

predefined signatures, HIDS cannot easily detect new malware and attacks because of their unknown signatures.

On the other hand, anomaly-based IDS (AIDS) can detect unknown malware and attacks by performing a deep analysis of the transmitted data [4]. Recent studies have developed machine learning (ML) algorithms for enhancing AIDS (ML-AIDS) [5]. These algorithms evaluate the network condition by classifying the processed data into either normal or abnormal classes. They train and test AIDS for detecting attacks and use false alarm rate and accuracy to measure the ability of AIDS in using different datasets. However, most of these datasets are extremely imbalanced in terms of cybersecurity, with the majority (98%) of these datasets being classified as normal, whereas the rest (2%) are classified as attacks [6]. Moreover, most of the available AIDSs use conventional ML methods to tune IDS models. Various evaluation metrics have also been applied to measure the performance of AIDSs [7]. However, only a few studies have examined the multi-class classification issue being faced by AIDS models.

Many studies use ML algorithms to build a proper IDS. Data should be learned and predicted before applying ML methods. The DARPA and KDD Cup 1999 datasets generated in 1998 and 1999, respectively, have been applied in 42% and 20% of these studies [8]. AIDS uses both datasets for testing and training ML models. However, these datasets suffer from several limitations, including outdated attack versions that may not be representative of real network attacks, inadequate information, and a large number of redundant records that introduce bias to frequency records when training NIDS [9]. NSL-KDD has been proposed to solve some issues being faced by KDD Cup 1999 [10], which also suffers from those problems mentioned in [11]. Tama [12] recently used web attacks to measure the effectiveness of AIDS based on ML classifiers in protecting web applications in the Internet of Things environment. These web attacks include cross-site scripting (XSS) and structured query language (SQL) injection, both of which cause data losses and cannot be detected in either the KDD Cup 1999 or UNSW-NB15 dataset [12].

AIDS models also suffer from performance and evaluation constraints. Therefore, a comparative analysis of these models by using a recently developed dataset is necessary to evaluate their efficiency in distinguishing malicious from benign activities and in identifying different types of attacks [13]. Evaluating these models also facilitates the selection of the most appropriate and accurate turning parameters for developing AIDS. Accuracy plays a crucial role in the assessment because a high false positive (FP) rate can lead to distrust in the system and ignorance of alerts.

This research focuses on modeling ML algorithms for AIDS and investigates some popular supervised ML models, such as an artificial neural network (ANN), support vector machine (SVM), decision tree (DT), random forest (RF), and Naive Bayes (NB) algorithms, as well as unsupervised ML models, such as k-means, self-organizing map (SOM), and expectation–maximization (EM) algorithms, to identify efficient and reliable AIDS methods. The most vital problem in testing and evaluating AIDS models is the application of novel types of attacks. Such a problem can be addressed by using a CICIDS2017 dataset that contains novel attack network traces. The main contribution of this work to the cybersecurity domain lies in two aspects. First, this study presents a comprehensive review of studies on the application of ML in AIDS that have been published over the past four years. These studies are also compared in terms of the classification accuracy and effectiveness of their proposed methods. Second, this study selects different ML-AIDS models for web attack detection that have benchmarking outcomes with low FP and false negative (FN) detection rates and short training and prediction times.

This article is organized as follows. Section II analyzes the most relevant works in ML-AIDS detection. Section III reviews the ML algorithms and the most popular AIDS datasets from a theoretical perspective. Section IV discusses the approaches for evaluating the performance of AIDS and the evaluation metrics used for measuring the effectiveness of classifiers. Section VI tests and evaluates the selected ML algorithms for implementing AIDS. Section VI summarizes the findings and presents some suggestions for future work.

II. RELATED WORK

Many researchers use different datasets to train NIDS models for detecting cybersecurity attacks. Appendix A lists the most significant and most widely cited IDS studies that employ different datasets for testing cybersecurity based on ML that have been published between 2016 and 2019. AIDS is used to evaluate IDS models because of its effectiveness in attack detection [39]. A classification task involves the classification of intrusions as either binary (when the goal is to distinguish normal from abnormal behaviors) or multi-class (when the intrusion is attributed to a specific attack category). This study then proposes an effective classification task for evaluating the latest NIDS models.

The NSL-KDD Cup 1999 dataset is used to analyze intrusion detection (ID), and the features of this dataset are classified into basic, content, traffic, and host. Each of these classifications assesses ID based on the detection rate (DR) and false alarm rate (FAR) [14], [15]. k-means clustering is applied to construct 1,000 clusters with more than 494,020 records and to highlight the association among different types of attacks and protocols utilized in intrusions. ANN is employed to analyze the NSL-KDD dataset [16]. Results show that DR has accurately categorized 81.2% and 72.9% of intrusions and attacks, respectively. Several other studies, such as [17], [18], and [19], have reported a decrease in the efficiency of NIDS models.

An association rule-mining algorithm is implemented to choose the most robust characteristics from the two datasets, and then classifiers are used to evaluate both preciseness and FAR. Results show that UNSW-NB15 has better characteristics than the KDD Cup 1999 dataset. Three IDS benchmark datasets that utilize ML algorithms have also been investigated in the literature [20]. The analysis is conducted by using clustering neural network algorithms that set apart the variances between synthetic and actual traffic. A fast feature selection technique that identifies inferior dataset quality features has also been applied [21]. The variances in random variables are utilized to ascertain the worthiness of features. This study analyzes the currently favored similarity-based algorithms, such as the maximal information compression index, correlation coefficient, and least square regression error, and results in highlighting certain features that are subsequently introduced into NB and k-NN to evaluate the performance of the proposed method. This method eventually outperforms other similarity-based algorithms in terms of computational expenditure.

A rough set of features that are selected based on the KDD Cup 1999 dataset has also been proposed [22]. This set is grounded on the assumptions that the data accuracy level is reduced and that the data pattern perceptibility has improved. Based on these assumptions, factual findings are derived from imperfect data. A selection of features that utilize RF is then obtained [23].

The UNSW-NB15 dataset has also been analyzed by using the Weka instrument [24]. The significant features from this dataset are selected by using various attributes of selection methods, including CfsSubsetEval (attribute evaluator) of the greedy stepwise method and InfoGainAttibuteEval (attribute evaluator) of the ranker method. The optimal selected attributes subset is then employed for the categorization by using several ML algorithms, including RF. The categorization that employs selective features shows improvements in its kappa statistics. Therefore, a weighted feature selection technique is recommended [25] for identifying Wi-Fi impersonation through the AWID dataset [26]. Appendix A lists the different datasets that have been used in the literature for evaluating AIDS. Figure 7 shows the most used dataset to evaluate NIDS by researchers and the publication date. The KDD Cup 1999 (1999) and NSL-KDD (2009) datasets utilize an outdated version of attacks, thereby affecting their reliability. UNSW-NB15 is a recently proposed dataset for training and testing the performance of AIDS models [27], [28], [29]. However, this dataset also has few drawbacks, including unrealistic attacks, reliability issues, and imbalanced properties. CICIDS2017 [30], [31] is considered the most appropriate dataset for training AIDS models due to the profile behavior of its users, which is based on HTTP, HTTPS, FTP, SSH, and email protocols in the application layer. In addition, CICIDS2017 is realistic and utilizes up-to-date attacks [32]. For these reasons, the CICIDS2017 dataset is selected in this research to evaluate AIDSs.

Deep-feature extraction and selection have also been applied for dataset feature reduction. The recommended method achieves 99.918% precision and 0.012% FAR. An analytical study of the CIC-IDS-2017 dataset using distance-based ML techniques [33] has applied the k-NN and k-means algorithms to analyze complexity. Detection systems grounded on the identification of anomalous instances are dependent on AI and ML for anomaly detection and demonstrate an autonomous ability to distinguish normal from abnormal system behavior.

AIDS models have been proposed using ML and DL for classifying lines connection into multi or binary classes [33], [77]. [80]–[82], [85]. In most of the AIDS models in Appendix A, the authors attempt to improve the accuracy of AIDS for performing classification binary task. The real challenge appears in multi-class attacks. Especially, most of the datasets in cybersecurity are formed with imbalanced classes to demonstrate the actual attacks. So, there is a need for powerful metrics to evaluate AIDSs according to the nature of training and testing data. Although, the accuracy improvement reaches up to 99.00 % in some of the existing work. However, relying only on accuracy does not reflect the actual performance of the AIDSs due to the model biased for the majority than minority class. Neglecting the criteria of the false alarm, precision, recall, F1-Score affect the effectiveness of the evaluation. Besides, very few studies focus on the efficiency of AIDS such as training time, testing time and detection time, CPU overload, and memory usage. The performance efficiency criteria are very important with the existence of distributed and high-speed connectivity [86], [88]. On the other hand, data mining techniques perform feature selection by using one of the three approaches, namely, wrapper, filter, and hybrid feature selection [34]. Anomaly detection models might identify the features that are more relevant to identify attacks by using feature selection techniques and some examples are presented in Appendix A. The main objective of applying feature selection algorithms is to improve the attack detection performance of AIDS models and reduce their detection time [35]. Many researchers prefer feature selection over improving the intrusion classifier to solve the false alarm issues in network intrusion detection. However, reducing the features or parameters can result in overfitting or making the classifiers biased toward specific classes of attacks [22]. Moreover, feature selection is unable to detect zero attacks given that the optimal subset of features is selected based on labeled instances and some objective functions that validate the results. While configuring feature selection for new known attacks is not practically applicable [24].

The review of the current studies of AIDS shows that all the existing studies are not meant for benchmarking AIDS models using ML or DL but focus on improving the detection of particular algorithms. Furthermore, the existing models are yet to reflect optimal performance. Table 1 has points out the various aspect and shortcomings of several existing studies that are related to the dataset selection, evaluation metric selection, and validation issues.

Table 1 shows that most of the existing studies only rely on accuracy to evaluate the performance of AIDSs. Also, there is a validation issue in which very few existing studies have compared their work with the state of the art AIDS models. Subsequently, the literature lacks a standard benchmarking methodology for AIDSs. Therefore, there is a need for a clear

TABLE 1. A critical review of the existing studies.

Ref.	Dataset issue	Evaluation metric issue	Validation issue	Remarks
[36]	√	*	*	The accuracy is used to evaluate without considering the confusion matrix and time. No proper validation is performed.
[37]	√	*	*	Binary classification is used and without considering the confusion matrix and time. No proper validation is performed.
[38]	*	*	*	Outdate attacks and without considering the confusion matrix and time. No proper validation is performed.
[39]	*	*	√	Outdate attacks, the accuracy is used to evaluate without considering the confusion matrix and time.
[40]	√	*	*	The accuracy is used to evaluate without considering the confusion matrix and time. No proper validation is performed.

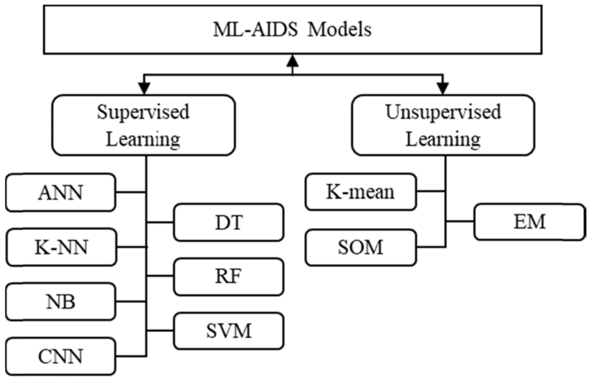


FIGURE 1. The ML-AIDS models.

benchmarking methodology for testing and evaluating the performance of the ML and DL-based AIDS models.

III. METHODS AND MATERIALS

An ML or DL algorithm can be trained by using various procedures, including supervised and unsupervised learning. Supervised learning performs classification based on data instances that are marked in the training phase. Supervised learning algorithms include ANN, DT (both types c4.5, ID3), k-NN, NB, RF, SVM, and CNN. Meanwhile, unlabeled data instances are detected via unsupervised learning, with clustering being the dominant learning method. Unsupervised learning algorithms include k-means clustering, EM clustering, and SOM as shown in Figure 1.

A. SUPERVISED LEARNING

Seven supervised ML algorithms are evaluated in ML-AIDS. The basic concepts of these algorithms are described as follows.

1) ANN

An ANN may be understood in visual terms as a weighted directed graph that comprises nodes and edges [41]. Those artificial neurons and directed edges with weights (that is, the strongest neurons) represent the linkages among artificial neurons. The output of a neuron serves as the input to other neurons. These neurons accept the input out of the world as a vector, which is similar to a certain pattern or image. The weights are adjusted while training the ANN to address classification issues. The ANN architecture comprises an input, output, and hidden layers, with each layer comprising neurons. The input layer accepts the input from the outside world, whereas the output layer reacts to the input introduced into the input layer based on its own learning capacity. The hidden layer serves as a mediator between the input and output layers and changes the input in a certain way to utilize the output layer. These layers may be partially or completely linked. In this study, a multilayer perceptron technique with backpropagation learning is applied. Figure 2 shows the general ANN architecture (I-H-O) for the C class, where I, H, and O represent the number of input, hidden layer, and output nodes, respectively.

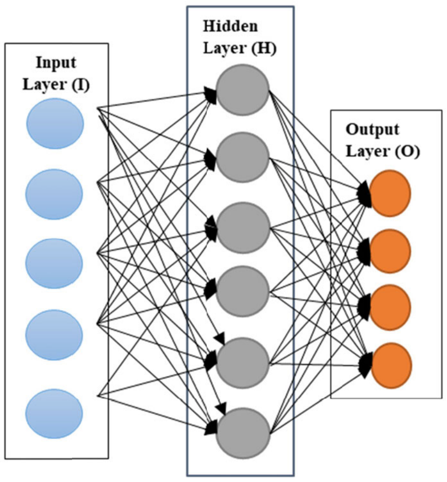


FIGURE 2. The ANN architecture.

2) DT

A DT often uses supervised learning algorithms to solve ML classification issues. Tree models, also known as classification trees, are used in situations where the target variable can accept discrete values as inputs. The components of a DT include leaves, branches, and nodes. While leaves represent the labels for each class, the branches represent the collection of attributes that result in the class labels. These branches function with discrete and continuous data. The DT algorithm divides the samples into two or more homogeneous sets according to an ultimate important splitter in input determinants. However, DT faces an overfitting issue that is dealt with via implementing bagging and boosting algorithms. DT functions efficiently over discrete data. Figure 3 presents the common architecture of a DT [42].

3) K-NN

According to [43], k-NN is an instance-based learning and classification method whose fundamentals lie in its distance

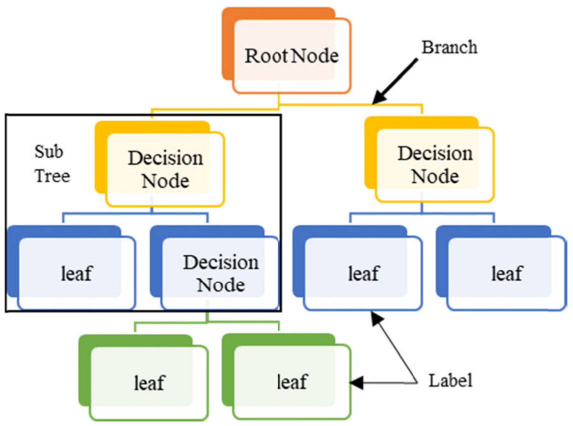


FIGURE 3. The DT architecture.

function, which computes the correlations or differences among pairs of instances or points. Some other alternative distance measures are also utilized in k-NN, including the Euclidean distance, which is a typical

$$\text{D(a,b)} = \sqrt{\sum\nolimits_{i=1}^{r} (ai - bi)^2}, \tag{1}$$

where a_i is the i_{th}-featured element of instance a, bi is the i_{th}-featured element of instance b, and r represents the entire quantity of dataset features. Euclidean distance is a non-parametric measure that lacks any conjecture of the fundamental data dissemination. The construct of the model is ascertained from the dataset, and this approach has been proven beneficial given that a huge bulk of data are derived from actual datasets without adhering to mathematical conjectures. Meanwhile, a lazy algorithm entails the construction of a model that does not require the training of data points. All training data are utilized during the testing stage, which is slower and more expensive than the training stage. In the worst-case scenario, k-NN requires additional time, memory, and training data storage to scan all data points.

4) NB

NB methods use a group of probabilistic classifiers that are created by implementing the Bayes theorem. These methods consider the naive conjectures of independence between each pair of features or attributes [44]. In processing the training data, the NB can rival the latest sophisticated methods within its domain, including SVM and ANN. NB can be easily trained by using a supervised learning structure. In numerous realistic implementations, the maximal probability technique is applied to calculate the parameters of NB models. In other words, the NB model can function with the refusal of Bayesian probability or by using any Bayesian technique. Bayes theorem can be formulated as

$$p\,(A \mid B) = \frac{p(\text{A}|\text{B})P\,(A)}{p\,(B)}, \tag{2}$$

where A represents the active target attribute or dependent event, B denotes the active predictor attribute or prior event, P (A) represents the prior probability of A, P(A|B) represents the posterior probability of B, and P(B|A) represents the probability of B if hypothesis A holds true.

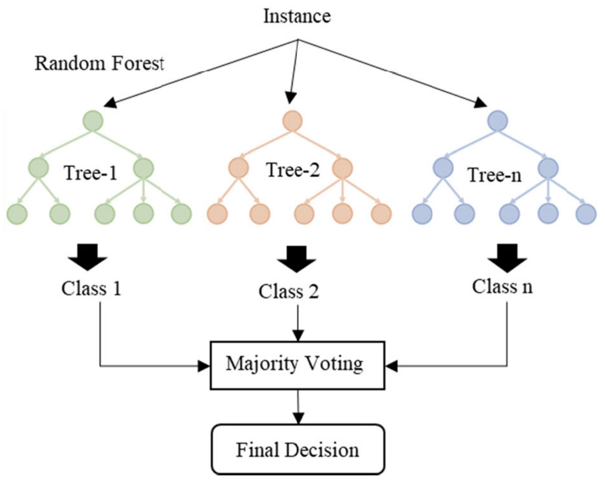


FIGURE 4. The RF architecture.

5) RF

As aforementioned, a DT faces an overfitting issue. RF rectifies this issue effectively by taking the average of multiple deep decision trees [45]. RF is an ensemble-learning algorithm that resolves classification and regression issues. This algorithm entails the construction of multiple DTs within the training timespan. RF outputs the classes' mode of a particular DT when executing a classification function and produces more superior outcomes than DT. Figure 4 shows the RF architecture.

6) SVM

An SVM identifies a hyperplane that categorizes the training instances into binary or multiclass. According to [46], the SVM algorithm accepts noted instances and related outputs, including binary or N-ary. An SVM is then constructed to categorize new instances. The training instances are mapped onto several points in the coordinate space, the instances are linearly segregated as input sets. Numerous hyperplanes that can segregate the training instance sets are available for the selection. However, the optimal selection is the maximum distance from the most proximal instance of any category. Between two hyperplanes, P categorizes the instances properly yet has less range away from the most proximal instance. By contrast, Q has a maximal range away yet shows a minute erroneous classification. Therefore, the hyperplane P is chosen. SVM can also be efficiently applied in high-dimensional spaces.

7) CNN

CNN is a deep learning approach that involves the application of different models with convolutional layers that determine the inputs, features, classification, training, and testing

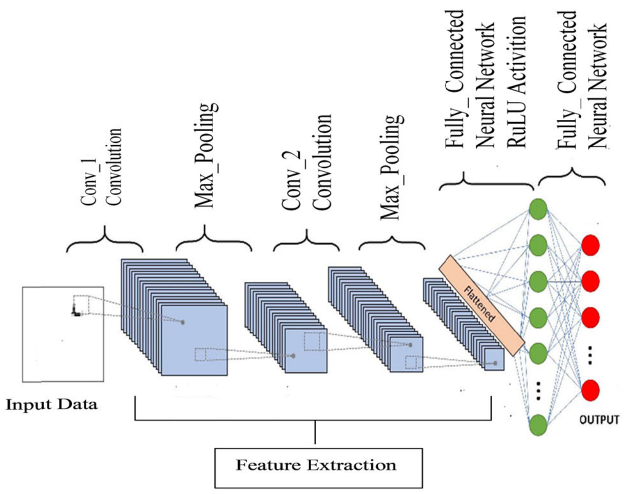


FIGURE 5. The CNN architecture.

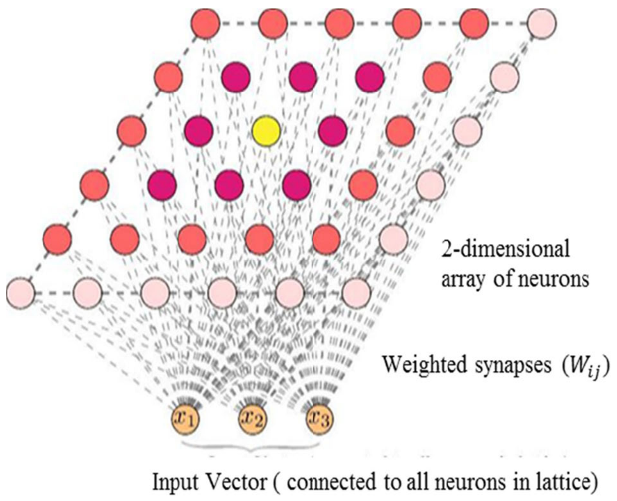

2-dimensional array of neurons
Weighted synapses (W_{ij})
x_1
x_2
x_3
Input Vector (connected to all neurons in lattice)

FIGURE 6. The SOM architecture.

procedures. The CNN model implements a softmax function to assign probabilistic values for each class. A typical CNN consists of an input layer that receives the data, convolution layers that create a features map from the data, pooling layers that select the maximum values of the features map in the convolution layer, a fully connected layer that learns and classifies outputs, and an output layer that represents the connection line as either normal or attack [47]. Some popular CNN architectures include GoogLeNet, AlexNet, ResNe, and VGGNet [48], whose main purpose is to classify images. Figure 5 presents the basic architecture of CNN for processing inputs and classifying outputs.

B. UNSUPERVISED LEARNING

Three unsupervised machine learning algorithms have been evaluated for performing in AIDSs. The basic concepts of these algorithms are described as follows.

1) K-MEANS

k-Means clustering is among the most uncomplicated unsupervised learning algorithms from the distance-based perspective. This algorithm segregates n instances into k clusters, in which each instance is a collective and the cluster has the nearest mean. As its main disadvantage, k-means clustering requires the number of clusters k to be pre-specified. Given a set of instances (p1, p2…pn), where each instance is a d-dimensional real vector, k-means clustering partitions p instances into ($k \leq p$) sets Z = {Z1, Z2,…, Zk} to minimize variance. k-Means is defined as follows [49] [50]:

$$a_z\min \sum_{i=1}^{k} \sum_{p\in Zi}^{max} |\, |p - m_i|\, |a_z \min \sum_{i=0}^{k} |Z_i| \mathrm{Var} Z_i, \quad (3)$$

where a is an argument, and μi represents the mean of points in set Zi.

2) EM

Extremely similar to k-means [51], EM extends k-means clustering in two ways. Specifically, EM computes the probabilities of cluster memberships according to one or more probability distributions and maximizes the entire probability of the data given the final clusters.

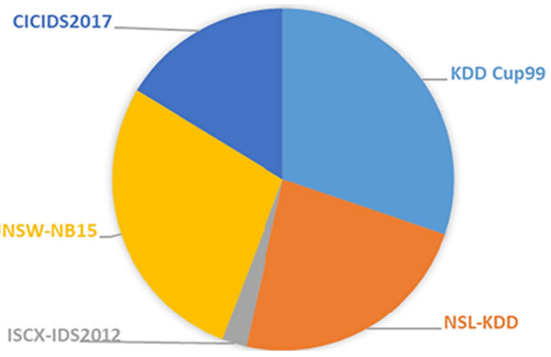


FIGURE 7. The usage of popular datasets in the literature.

3) SOM

SOM is an algorithm based on an unsupervised learning class of neural network models that can cluster data without requiring prior knowledge of the input data class categories [52]. SOM derives a topology-preserving mapping from the high-dimensional data space to map neurons (units). The mapping conserves the distance between points. The mutually proximal points are mapped to nearby map units in SOM. The SOM network can ascertain those inputs that it has come across before. Figure 6 illustrates the SOM architecture.

C. DATASETS

The most problematic stage in assessing IDS is determining the proper dataset. This section discusses the typically employed datasets in IDS evaluation, including KDD Cup 1999, NSL-KDD, UNSW-NB15, and CICIDS2017. Figure 7 shows that most researchers use the KDD Cup 1999 or NSL-KDD dataset to evaluate NIDS [39], [40]. However, these same reasons make this dataset highly complex and require advanced solutions. The most popular datasets used in the literature are described as follows:

1) KDD Cup1999

The KDD Cup 1999 dataset addresses the requirement of using suitable data for evaluating IDS. This dataset was built as a simulation dataset in 1998 and has since then been intensively used in data mining and machine learning fields. KDD Cup 1999 contains both training and test data and has 41 features that are classified into essential, traffic, and content features [53]. The standard KDD Cup 1999 dataset contains approximately five million raw data, of which attack data account for approximately 80%. These data are categorized into one ''normal'' category and four ''attack'' categories.

- Normal: Non-attack types of data.
- Attack types: Denial of service (DOS), user to root (u2R), probing attacks (Probe), and root to local (R2L).

A total of 22 attack types are included in KDD Cup 1999, with each attack being part of the categories mentioned above. The KDD Cup 1999 dataset has numeral (binary and real numbers) and text information about the requested categories and can also label data as either intrusion or non-intrusion.

2) NSL-KDD

The NSL-KDD dataset contains the most important records of the KDD Cup 1999 dataset and classifies its data features into several groups [10]. This dataset reduces the data size by deleting duplicate records and consequently improve the function of ML algorithms.

3) UNSW-NB15

The UNSW-NB15 dataset is created by using the IXIA PerfectStorm tool and a simulation program. Unlike NSL-KDD, this dataset embodies original versions of different ID cases that commonly emerge nowadays. This dataset has 175,341 normal classes, 82,332 anomaly classes, and 49 extracted features [54]. The attacks in this dataset include fuzzer, analysis, backdoor, DoS, exploit, generic, reconnaissance, shellcode, and worm attacks.

4) CICIDS2017

The CICIDS2017 dataset closely simulates real-world network data (PCAPs) and uses CICFlowmeter-V3.0 to extract 78 features and 79 labels. This dataset includes the abstract characteristic attitudes of 25 users according to the HTTP, HTTPS, FTP, SSH, and email protocols as shown in Table 2.

Data are captured across different periods. Based on the 2016 McAfee Report, the attacks in this dataset are classified into brute force FTP, brute force SSH, DoS, heartbleed, web, infiltration, botnet, and DDoS attacks, which are not found in any of the previously mentioned datasets [55]. CICIDS2017 achieves an abstract characteristics profiling of human interactions by using the B-Profile system and applies the Alpha profile to simulate various multi-stage attack scenarios. The main features of this dataset are distinguished from those of others in terms of their realistic and reliable benchmarking. The benchmarking applies 11 criteria, including complete traffic and available protocols, to ensure the reliability of the evaluation [56].

TABLE 2. Details of the CICIDS2017 dataset.

Name of Files	Class Found
Monday-Hours.pcap_ISCX.csv	Benign (Normal human activities)
Tuesday-Hours.pcap_ISCX.csv	Benign, FTP-Patator,SSH Patator
Wednesday-.pcap_ISCX.csv	Benign, DoS GoldenEye, DoSHulk, DoS lowhttptest, DoS slow loris, Heartbleed
Thursday-WebAttacks.pcap_ ISCX.csv	Benign, Brute Force, SQL Injection, XSS.
Thursday-Infilteration.pcap_.csv	Benign, Infiltration
Friday-pcap_ISCX.csv	Benign, Bot
Friday-PortScan.pcap_ISCX.csv	Benign, PortScan
Friday- DDos.pcap_ISCX. csv	Benign, DDoS

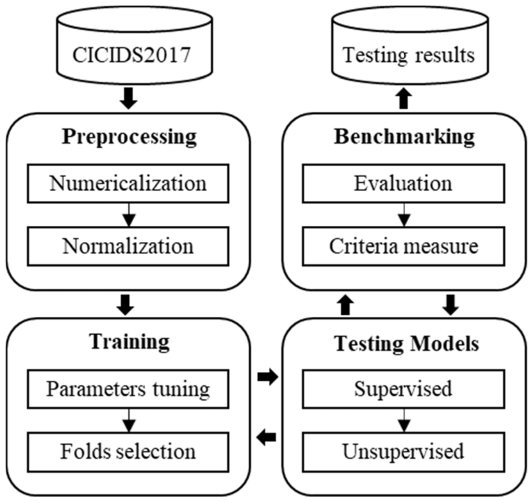


FIGURE 8. The benchmarking methodology of ML-AIDS.

IV. BENCHMARKING OF ML-AIDS IN CICIDS2017

AIDS has received much research interest since the beginning of this decade. Building an AIDS by using AI or ML techniques to develop a confidant network that resists modern types of attacks remains an important concern for researchers. Many studies improve AIDS and evaluate their results by using different metrics. Most ML algorithms are parameterized, in which their behavior cannot be estimated from the processed data. Furthermore, random parameters significantly affect the performance of AIDS models [57]. Subsequently, the behavior of parameters must be tuned to achieve an adequate evaluation. Figure 8 shows a proposed benchmarking methodology and related procedures for testing and evaluating ML-AIDS models.

A. PRE-PROCESSING

This study uses MachineLearning.CSV data, which is part of the CICIDS-2017 dataset from the ISCX Consortium. MachineLearning.CSV consists of eight (8) traffic

TABLE 3. Benchmarking the ML-AIDS algorithm.

#	Step
1.	*Divide the dataset based on 5-fold cross-validation of training and testing without removing any instance or feature to ensure the test robustness of the ML-AIDS models;*
2.	*Turn parameters of an ML-AIDS model manually, then train and test the model;*
3.	*Evaluate the results of the model by using the proposed evaluation metrics;*
4.	*Repeat steps 2 and 3 until the hyperparameters of the model are obtained based on the best result.*
5.	*Conclude the hyperparameters of the ML-AIDS model.*
6.	*Repeat step 2 to 5 for all ML-AIDS models;*
7.	*Present the benchmarking results of the ML-AIDS models based on the evaluation metrics;*
8.	*Identify the pros and cons of each model and choose the best model.*

monitoring sessions, each is in the form of a comma-separated value of CSV file format. This file contains normal traffic defined as ''BENIGN'' traffic and anomaly traffic called ''Attacks'' traffic. The attack traffics are described in more detail in the second column of Table 2. Other than normal traffic and benign traffic, there are 14 types of attacks in this dataset.

The numericalization step includes the process of replacing noise values such as null or infinity symbols with zeros or mean values. Next, the normalization step is needed because some of the CICIDS2017 attributes have very large values and non-distributed data according to their histograms. To bring all attribute values to the same scale, we apply the normalizing procedure within the [–3, 3] interval by using the formulas in (4-7):

$$\mathrm{X} = -3 < \mathrm{minimum}(z\_\mathrm{max}, \mathrm{maximum}(z, z\_\mathrm{min})) < 3 \quad (4)$$

where the standardization, z_i is represented as

$$z_i = \frac{x-\mu}{\sigma} \quad (5)$$

Mean, μ is represented as,

$$\mu = \frac{1}{N}\sum\nolimits_{i=1}^{N}(x_i) \quad (6)$$

and standard deviation, σ represents as,

$$\sigma = \sqrt{\frac{1}{N}\sum\nolimits_{i=1}^{N}(x_i-\mu)^2} \quad (7)$$

The outcome of the preprocessing is a set of 38 features that have values within the range of [−3,3]. This range is selected in order to improve the distribution of the data and achieve better training outcomes.

B. TRAINING

Many strategies can be used to generate hyperparameters, with trial and error being the most common strategy [58]. Another strategy is k-folds cross-validation, which divides the dataset into training and testing parts. One approach for evaluating the performance of ML-AIDS models and setting the benchmarking results is described in Table 3.

In this benchmarking methodology, k-folds cross-validation parts are carefully set to specific limits of training and testing percentages (i.e., 40%–60%, 50%–50%, or 60%–40%) that are not visible in the training stage to test the reliability, generalizability, and effectiveness of ML-AIDS models. The training starts by turning the parameters of each algorithm as explained in the following section, and the output is evaluated. The same process is repeated for the other parameters.

C. TESTING

The testing phase includes applying 10 popular supervised and unsupervised ML algorithms for identifying effective and efficient ML–AIDS of networks and computers. The supervised algorithms are ANN, DT, k-NN, NB, RF, SVM, and CNN. The unsupervised algorithms are EM, k-means, and SOM. Several models of these algorithms are introduced, and the turning and training parameters of each algorithm are examined to achieve optimal classification results.

D. BENCHMARKING

The performance of ML algorithms cannot be visually interpreted, and quantitative metrics (e.g., precision, recall, F1, and confusion matrix) need to be used in the evaluation. The best evaluation metrics depend on the selected ML algorithms, the processed data, and the application domain. This study evaluates AIDSs by using the CICIDS2017 dataset and by taking accuracy, precision, sensitivity (recall), F-Score, training time, and prediction time as evaluation metrics.

1) ACCURACY

Accuracy refers to the ratio of correct predictions for both the true positives (TP) and true negatives (TN) of attacks compared with the total number of tested cases.

$$\mathrm{Accuracy} = \frac{TP+TN}{TP+TN+FN+FP} \quad (8)$$

2) PRECISION

Precision (TP rate) measures the proportion of positives that are correctly identified as in (9):

$$\mathrm{Precision} = \frac{TP}{TP+FP} \quad (9)$$

3) SENSITIVITY

Sensitivity (Recall) measures the number of correct classifications penalized by the number of missed entries identified as in (10):

$$\mathrm{Sensitivity} = \frac{TP}{TP+FN} \quad (10)$$

4) F1-SCORE

F1-Score finds a balance between Precision and Recall as in (11):

$$\mathrm{F1\_Score} = 2*\frac{Precision+Recall}{Precision*Recall} \quad (11)$$

TABLE 4. Evaluation parameters.

Class	Predicted Positive	Predicted Negative
Class of Attack (Positive)	Predicted Attack as attacks (TP)	Predicted Attacks as Normal (FN)
Class of Normal (Negative)	Predicted Normal as Attacks (FP)	Predicted normal (TN) as Normal

5) TRAINING TIME

Training time (T1) describes how much time an approach uses to train the whole dataset and to build the NIDS model with the best fit as in (12):

$$\mathrm{T1} = end^{training}_{time} - start^{training}_{time} \tag{12}$$

6) TESTING TIME

Testing time (T2) describes how much time an approach uses to predict the whole dataset as either normal or attack as in (13):

$$\mathrm{T2} = end^{testing}_{time} - start^{testing}_{time} \tag{13}$$

TP refers to the number of accurately classified attacks, whereas FP refers to the number of normal connections that have been incorrectly classified as attack connections. Meanwhile, TN refers to the number of accurately classified normal connections, whereas FN refers to the number of attack connections that have been inaccurately classified as normal. The basic performance metrics used in the evaluation are summarized in Table 4.

V. RESULTS AND DISCUSSION

This section discusses the benchmarking classification algorithms for a multi-class label AIDS CICIDS2017 dataset. These 10 ML algorithms are categorized into 7 supervised (k-NN, SVM, DT (both types c4.5, ID3), RF, ANN, NB, and CNN) and 3 unsupervised (K-means clustering, EM clustering, and SOM) algorithms. Some of these algorithms are built by several models, whose settings revolve around the tuning of parameters for each algorithm to determine those parameters with the best fit and the optimal initial values for training and testing. The ML-AIDS algorithms are implemented by using Python3 in Anaconda 3 on a computer with OPTIPLEX 3010 Dell, Intel Core i3, 3.60 GHz processor, 4 GB primary memory, and 2 GB GPU functioning on Ubuntu 16.04. The selected ML algorithms are subjected to seven supervised and three unsupervised learning tests. The evaluation metrics include accuracy, precision, recall, F1-Score, T1, and T2. The AIDS CICIDS2017 dataset used in these tests has four types of classes whose names, labels, and supports are presented in Table 5. Support represents the number of tested instances (both normal and attacks; support = N1–(N1/N2), where N1 denotes the number of instances in the dataset and N2 denotes the size of the dataset.

TABLE 5. Classes in the CICIDS2017 testing dataset.

#	Class name	Class label	Support
1	BENIGN	C1	53518
2	Brute Force	C2	482
3	XSS	C3	210
4	SQL Injection	C4	9

TABLE 6. Performance evaluation results for the ANN algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
Solver = lbglfs = loss = categorical crossentropy	C1	1.00	1.00	1.00	1.00	95.47	78.94
	C2	0.99	0.58	0.99	0.73		
	C3	0.02	1.00	0.01	0.02		
	C4	0.00	0.00	0.00	0.00		
Solver = adam, loss = categorical crossentropy	C1	1.00	1.00	1.00	1.00	53.78	48.03
	C2	0.98	0.57	0.98	0.72		
	C3	0.04	0.80	0.04	0.07		
	C4	0.33	0.20	0.11	0.14		
Solver = sgd loss = categorical crossentropy	C1	1.00	1.00	1.00	1.00	56.98	57.72
	C2	0.86	0.53	0.86	0.66		
	C3	0.00	0.00	0.00	0.00		
	C4	0.00	0.00	0.00	0.00		

A. RESULTS OF SUPERVISED LEARNING ALGORITHMS

The results of the seven ML-AIDS are described in detail as follows:

1) ANN

The ANN classifier testing involves three models that are represented based on the parameters related to the construction of the training models. These parameters are assigned default values (activation = ‘relu,’ alpha = 0.0001, batch_size = ‘auto,’ number of hidden layers = ‘4’, Optimizer = ‘’). Table 6 presents the results for the three ANN models.

Some differences are observed in the training parameters. Specifically, one ANN model can detect C1, C2, C3, and C4 attacks, whereas the other two models fail to detect C4 attacks. The best ANN model has the settings Solver = ‘Adam,’ loss = ‘categorical_crossentropy,’ and Epoch = 100. This model achieves a 99.31% accuracy, 99.50% precision, 99.31% recall, 99.22% F1-Score, 53.78s training time, and 48.03s testing time.

2) DT

The DT classifier testing involves six models that are represented based on the parameters related to the construction of the training models. The maximum depth (max depth) and feature type (gini and entropy) are two adjustable parameters.

TABLE 7. Performance evaluation results for the DT algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
criterion = 'gini', max depth=4	C1	1.00	1.00	1.00	1.00	1.13	0.73
	C2	0.86	0.67	0.86	0.76		
	C3	0.03	0.88	0.03	0.06		
	C4	0.00	0.00	0.00	0.00		
criterion = 'gini', max depth = 6	C1	1.00	1.00	1.00	1.00	3.33	1.68
	C2	0.78	0.72	0.78	0.75		
	C3	0.32	0.45	0.32	0.38		
	C4	0.00	0.00	0.00	0.00		
criterion = gini, max depth = None, class weight = balanced	C1	1.00	1.00	1.00	1.00	1.85	1.33
	C2	0.73	0.73	0.72	0.72		
	C3	0.44	0.41	0.43	0.42		
	C4	0.44	0.57	0.44	0.50		
criterion = entropy, max depth=4	C1	1.00	1.00	1.00	1.00	0.90	0.75
	C2	0.90	0.64	0.90	0.75		
	C3	0.00	0.00	0.00	0.00		
	C4	0.00	0.00	0.00	0.00		
criterion = entropy, max depth = 6	C1	1.00	1.00	1.00	1.00	1.85	0.79
	C2	0.87	0.70	0.88	0.78		
	C3	0.17	0.47	0.17	0.25		
	C4	0.11	0.40	0.22	0.29		
criterion = entropy, max depth = None, class weight = balanced	C1	1.00	1.00	1.00	1.00	1.23	1.12
	C2	0.74	0.73	0.74	0.73		
	C3	0.37	0.38	0.37	0.38		
	C4	0.67	0.67	0.67	0.67		

TABLE 8. Performance evaluation results for the k-NN algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
k=1	C1	1.00	1.00	1.00	1.00	11.13	7.92
	C2	0.73	0.72	0.73	0.73		
	C3	0.44	0.41	0.43	0.42		
	C4	0.29	0.25	0.22	0.24		
k=2	C1	1.00	1.00	1.00	1.00	41.16	10.41
	C2	0.89	0.70	0.89	0.78		
	C3	0.18	0.45	0.19	0.27		
	C4	0.22	1.00	0.22	0.36		
k=3	C1	1.00	1.00	1.00	1.00	47.44	11.92
	C2	0.79	0.73	0.79	0.76		
	C3	0.40	0.46	0.40	0.43		
	C4	0.33	0.75	0.33	0.46		
k=4	C1	1.00	1.00	1.00	1.00	46.48	13.13
	C2	0.87	0.70	0.87	0.77		
	C3	0.20	0.41	0.21	0.28		
	C4	0.00	0.00	0.00	0.00		
k=5	C1	1.00	1.00	1.00	1.00	51.2	11.34
	C2	0.80	0.70	0.80	0.75		
	C3	0.35	0.43	0.34	0.38		
	C4	0.00	0.00	0.00	0.00		

The tree level value is determined by max depth, which starts from 1 to 6, with "None" being the last value. The tree level value affects the model fitting. For instance, a higher max depth corresponds to higher accuracy, whereas a lower max depth leads to underfitting and implies a poor AIDS performance. The selected feature is either the "gini" of the Gini impurity criterion or the "entropy" of the information gain criterion. An attack is detected when these features have low values. Table 7 presents the results for the six DT models.

Some differences are observed in the training parameters. Three DT models have a high level of nodes and can detect C1, C2, C3, and C4 attacks, two models fail to detect C4 attacks, and one model fails to detect skewed (C3 and C4) attacks. In sum, this dataset is extremely imbalanced, and we set the class weight to "balanced." The best DT model has the settings criterion = entropy, max depth = none, and class weight = balanced. This model achieves 99.49% accuracy, 99.49% precision, 99.49% recall, 99.49% F1-Score, 1.23s training time, and 1.12s testing time.

3) K-NN

The k-NN classifier testing involves five models denoted by 1, 2, 3, 4, and 5 k neighbors. Table 8 presents the results of these models. Some differences are observed in their k values. Specifically, 3-NN models can detect all BENIGN (C1), Brute Force (C2), XSS (C3), and SQL Injection (C4) attacks, whereas 4-NN and 5-NN models fail to detect C4 attacks. The 1-NN model obtains the best results for all classes with 99.49% accuracy, 99.5% precision, 99.49% recall, 99.49% F1-score, 11.13s training time, and 7.92s testing time.

4) NB

The NB classifier testing involves only one default model that is able to detect C1, C2, C3, and C4 attacks and achieves 98.86% accuracy, 99.01% precision, 98.86% recall, 98.85% F1-Score, 1.07s training time, and 0.15s testing time. Table 9 shows that the NB classifier has a positive detection ability for modern types of attacks and has a considerably short detection time.

TABLE 9. **Performance evaluation results for the NB algorithm.**

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
default	C1	1.00	1.00	1.00	1.00	1.07	0.15
	C2	0.09	0.20	0.09	0.13		
	C3	0.97	0.33	0.97	0.49		
	C4	1.00	0.30	1.00	0.46		

5) RF

The RF classifier has many subtrees that are constructed to detect various types of attacks. The number of subtrees and maximum tree level in RF affects both detection rate and time complexity. Accordingly, the subtree is treated as the parameter in this evaluation. The RF classifier testing involves five models that are represented based on the parameters related to the construction of the training models. The maximum depth (max depth) and the number of estimators (n estimators) are both adjustable. Table 10 presents the results of the five RF models. Some differences are observed in the training parameters.

Specifically, two RF models are able to detect C1, C2, C3, and C4 attacks, two models fail to detect C4 attacks, and one model fails to detect C2, C3, and C4 attacks. In sum, the dataset is extremely imbalanced, and we set the class weight to "balanced" in order for the training model to pay attention to the skewed attacks than to the other majority classes. The best RF model has the settings n estimators = 100, max depth = none, and class weight = balanced. This model achieves 99.54% accuracy, 99.56% precision, 99.54% recall, 99.55% F1-Score, 9.38s training time, and 6.76s testing time. However, RF is more time consuming compared with the other models.

6) SVM

The SVM classifier testing involves four models that are represented based on the parameters related to the construction of the training models. The SVM model serves as a kernel function in these datasets. Table 11 presents the results of four SVM models. Some differences are detected in the kernel function and training parameters. Specifically, one SVM model is able to detect C1, C2, C3, and C4 attacks, whereas the other three models fail to detect the rare C4 attacks. In sum, the dataset is extremely imbalanced. Accordingly, the class weight is set to "balanced" (to pay attention to the skewed attacks than to the other majority classes), and the number of iterations is set to –1. The best SVM model has the settings kernel = RBF, max iter = –1, and class weight = balanced and achieves 96.72% accuracy, 99.27% precision, 96.72% recall, 97.89% F1-Score, 343.56s training time, and 33.17s testing time. However, this model has a longer detection time compared with the other models.

TABLE 10. **Performance evaluation results for the RF algorithm.**

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
n estimators=1, max depth=1,	C1	1.00	0.99	1.00	0.99	2.73	0.12
	C2	0.00	0.00	0.00	0.00		
	C3	0.00	0.00	0.00	0.00		
	C4	0.00	0.00	0.00	0.00		
n estimators =1, max depth=6	C1	1.00	1.00	1.00	1.00	0.69	0.16
	C2	0.91	0.54	0.91	0.67		
	C3	0.04	0.70	0.03	0.06		
	C4	0.00	0.00	0.00	0.00		
n estimators =100, max depth=6	C1	1.00	1.00	1.00	1.00	8.44	7.48
	C2	0.89	0.69	0.89	0.78		
	C3	0.03	1.00	0.03	0.06		
	C4	0.00	0.00	0.00	0.00		
n estimators =100, max depth = none	C1	1.00	1.00	1.00	1.00	7.82	4.99
	C2	0.74	0.76	0.74	0.75		
	C3	0.48	0.44	0.46	0.45		
	C4	0.22	1.00	0.22	0.36		
n estimators =100, max depth = none, class weight = balanced	C1	1.00	1.00	1.00	1.00	9.38	6.76
	C2	0.71	0.78	0.75	0.76		
	C3	0.56	0.47	0.50	0.49		
	C4	0.22	1.00	0.22	0.36		

7) CNN

The CNN classifier testing involves two models that are represented based on the parameters related to the construction of the training models as shown in Table 12. The main parameter is the epoch iteration of the training vectors. Table 11 presents the results of two CNN models. No differences are detected in the training parameters. Both models fail to detect C4 attacks and require a considerably long training time. The best CNN model has 100 epochs and achieves 99.50% accuracy, 99.46% precision, 99.50% recall, 99.47% F1-Score, 261.8s training time, and 1.73s testing time.

B. RESULTS OF UNSUPERVISED LEARNING ALGORITHMS

The results of the three unsupervised ML-AIDS are described as follows:

1) K-MEANS

The k-means classifier testing involves three models that are represented based on the parameters related to the

TABLE 11. Performance evaluation results for the SVM algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
kernel = RBF, max iter=-1	C1	1.00	1.00	1.00	1.00	64.25	8.75
	C2	0.86	0.50	0.86	0.63		
	C3	0.04	1.00	0.04	0.07		
	C4	0.00	0.00	0.00	0.00		
kernel = poly, max iter=-1	C1	1.00	1.00	1.00	1.00	18.91	7.69
	C2	0.87	0.49	0.87	0.62		
	C3	0.04	0.89	0.04	0.07		
	C4	0.00	0.00	0.00	0.00		
kernel = linear, max iter=-1	C1	1.00	1.00	1.00	1.00	25.98	9.46
	C2	0.91	0.49	0.91	0.64		
	C3	0.04	1.00	0.04	0.07		
	C4	0.00	0.00	0.00	0.00		
kernel = RBF, max iter=-1, class weight = balanced	C1	0.97	1.00	0.97	0.99	343.56	33.17
	C2	0.49	0.58	0.49	0.53		
	C3	0.80	0.16	0.80	0.26		
	C4	0.78	0.01	0.78	0.02		

TABLE 12. Performance evaluation results for the CNN algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
epoch 25	C1	1.00	1.00	1.00	1.00	261.8	8.13
	C2	0.70	0.99	0.82	0.99		
	C3	0.63	0.06	0.10	0.06		
	C4	0.00	0.00	0.00	0.00		
epoch 100	C1	1.00	1.00	1.00	1.00	287.4	9.00
	C2	0.99	0.70	1.00	0.82		
	C3	0.05	0.44	0.02	0.04		
	C4	0.00	0.00	0.00	0.00		

construction of the training models. Table 14 presents the results of three k-means models. Slight differences are observed in the training parameters. All k-means models are able to detect C1, C2, C3, and C4 attacks but have poor classification performance. The best k-means model has the settings n clusters = 4 and max_iter = 300. This model achieves 23.41% accuracy, 67.37% precision, 23.41% recall, 37.36% F1-Score, 3.12s training time, and 2.99s testing time.

TABLE 13. Performance evaluation results for the k-means algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
default	C1	0.39	0.98	0.61	0.75	11.19	9.69
	C2	0.31	0.00	0.05	0.01		
	C3	0.02	0.00	0.01	0.00		
	C4	0.33	0.00	0.44	0.00		

TABLE 14. Performance evaluation results for the EM algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
n_clusters=4, max iter = 10	C1	0.29	1.00	0.32	0.48	3.15	2.58
	C2	0.17	0.00	0.04	0.00		
	C3	0.02	0.01	0.57	0.01		
	C4	0.44	0.00	0.33	0.00		
n clusters=4, max iter = 30	C1	0.28	0.98	0.22	0.35	2.99	2.95
	C2	0.62	0.01	0.41	0.02		
	C3	0.19	0.00	0.20	0.01		
	C4	0.22	0.00	0.11	0.00		
n clusters=4, max iter = 300	C1	0.36	0.99	0.23	0.38	3.21	3.44
	C2	0.03	0.00	0.18	0.01		
	C3	0.19	0.01	0.38	0.01		
	C4	0.33	0.00	0.22	0.00		

2) EM

The EM classifier testing involves only one default model. The EM algorithm is used for traffic data analysis and uses the same features utilized in k-means clustering. Table 13 shows that the EM classifier has a poor attack detection ability.

The default model can detect C1, C2, C3, and C4 attacks but achieves poor classification performance. This model also achieves 60.06% accuracy, 86.88% precision, 60.06% recall, 74.11% F1-Score, 11.19s training time, and 9.69s testing time.

3) SOM

The SOM classifier testing involves only one default model that can only detect C1 attacks and has false alarms that are higher than those of the other algorithms. This model also achieves 59.06% accuracy, 85.88% precision, 60.00% recall, 74.11% F1-Score, 120.27s training time, and 0.05s testing time. Table 15 shows that the SOM classifier has poor attack detection ability.

TABLE 15. Performance evaluation results for the SOM algorithm.

Model Setting	Class label	Accuracy	Precision	Recall	F1-Score	T1 (s)	T2 (s)
default	C1	0.97	0.96	0.95	0.47	120.27	0.05
	C2	0.00	0.00	0.00	0.00		
	C3	0.00	0.00	0.00	0.00		
	C4	0.00	0.00	0.00	0.00		

C. OVERALL EVALUATION

This work investigates the attack detection ability of ML-AIDS algorithms by testing them on the CICIDS2017 dataset. This paper also contributes to the benchmarking of 10 classification algorithms, which are classified into supervised (k-NN, SVM, DT, RF, ANN, NB, and CNN) and unsupervised learning (k-means clustering, EM, and SOM) algorithms. Some ML-AIDS algorithms are represented by several models. These algorithms are tested on 48 models. However, only 31 models are reported in this paper after excluding those that obtain very poor results.

Table 16 presents the overall performance of the 10 tested ML-ATDS. In general, the k-NN, DT, and NB algorithms have a greater capability to detect web attacks compared with the other algorithms. Among all ML-AIDS algorithms, the supervised learning algorithms outperform the unsupervised ones. DT and k-NN emerge as the best supervised learning algorithms when both training and testing time are considered and ignored, respectively. Meanwhile, EM emerges as the best unsupervised learning algorithm with and without considering training and testing time. These results cover all the tested models of a particular algorithm and the standard deviation (SD) of their accuracy. It also covers the results of five related studies of [59], [60]–[62] and [63]. As discussed before, the related work neglects to measure the efficiency of the performance that is represented by the training and testing time. Furthermore, several benchmark algorithms of this work surpass the results of the related work. Here, the related work that has the bias of data balancing and feature selection have been excluded.

Figure 9 shows the average accuracy, training time, and testing time of the 10 algorithms. This figure supports our finding that DT, k-NN, and NB are the best basic ML-AIDS algorithms for detecting web attacks.

Correspondingly, several biased instances may be selected due to the highly unbalanced classes of the CICIDS2017 dataset. Model building time refers to the total amount of time for an ML algorithm to develop a trained model out of the training data. This duration needs to be set as short as possible to minimize the amount of time for a trained model to detect intrusion. Algorithms need to be repeatedly trained online in order to discover new types of

TABLE 16. Overall performance of the ML-AIDS algorithms.

Algorithm	Accuracy	Precision	Recall	F1-Score	Accuracy SD	T1(s)	T2(s)
ANN	0.9928	0.9937	0.9928	0.9917	0.1233	53.78	48.03
DT	0.9949	0.9943	0.9949	0.9942	0.1363	1.23	1.12
k-NN	0.9952	0.9949	0.9952	0.9949	0.1473	11.13	7.92
NB	0.9886	0.9901	0.9886	0.9885	0.2324	1.07	0.15
RF	0.9930	0.9909	0.9930	0.9912	0.1110	9.38	6.76
SVM	0.7521	0.9916	0.7521	0.7660	0.3084	343.56	33.17
CNN	0.9947	0.9943	0.9946	0.9944	0.4936	261.80	1.73
k-means	0.2559	0.9747	0.2559	0.3996	1.0127	3.12	2.99
EM	0.6006	0.8688	0.6006	0.7411	1.0968	11.19	9.69
SOM	0.5906	0.8588	0.6000	0.7411	1.1096	120.27	0.05
[59]	*	0.96	0.96	0.96	*	*	*
[60]	0.84	*	*	*	*	*	*
[61]	*	0.95	0.98	0.96	*	*	*
[62]	0.98	99.52	98.68	92.76	*	*	*
[63]	*	0.34	0.50	0.74	*	*	*

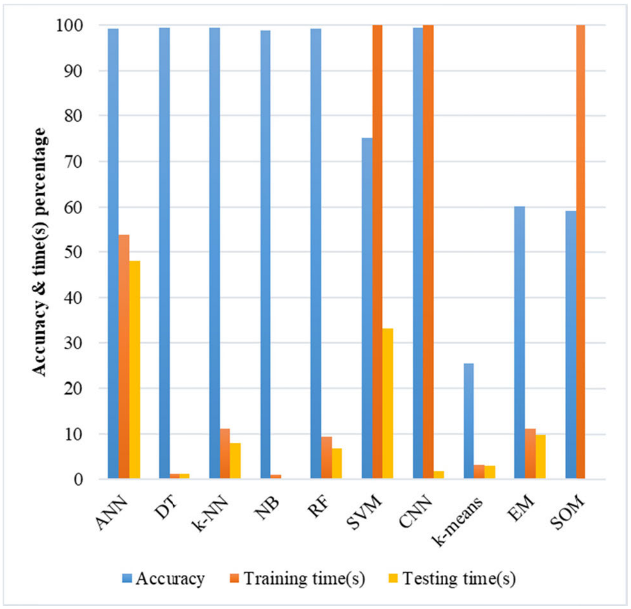


FIGURE 9. The overall accuracy and runtime performance.

attacks. Figure 9 shows that the DT, RF, NB, k-means, and EM algorithms have a much shorter model building duration compared with the other algorithms, whereas the CNN, SVM, SOM, and ANN algorithms consume much time to build a trained model.

Appendix B shows the performance evaluation results of the tested ML-NIDS algorithms for each type of attack. From

TABLE 17. A review analysis of the ML-AIDS models.

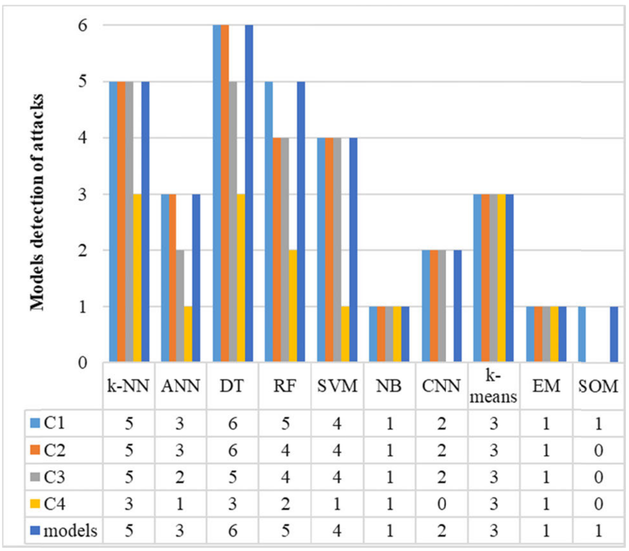
Ref.	Algorithm or Model	Dataset	Detected Attacks	Effectiveness					Efficiency			Classification type	
				Precision	Recall	F-score	Accuracy	Detection	Training Time (ms)	Memory Usage	Testing (ms)	Binary	Multi-Class
[59]	SVM+ANN	NSL-KDD	5- Classes	*	*	*	0.99	*	*	*	*	*	√
[60]	Gain-ratio feature selection and decision trees	kddcup99.	5- Classes	0.99	0.97	0.78	0.82	*	√	√	*	*	√
[23]	Random forest as feature selection	kddcup99.	Normal	0.98	*	*	*	*	*	*	*	*	√
			DoS	0.92	*	*	*	*					
			U2R	0.17	*	*	*	*					
			R2L	0.66	*	*	*	*					
			Probe	0.60	*	*	*	*					
[61]	Bat algorithm with k-means	kddcup99.	5- Classes	*	*	*	*	*	*	*	*		√
[62]	Set of feature grouping based on pairwise MI and LS-SVM as an intrusion classifier	kddcup99.	2- Classes	*	*	*	0.96	0.95	√	√	*	√	*
[63]	pure cluster" and propose a hierarchical semi-supervised k-means algorithm (HSK-means)	kddcup99.	Normal	0.96	0.85	0.92	*	0.86	*	*	*	*	√
			DoS	0.97	0.97	0.97	*	0.99					
			U2R	0.75	0.73	0.75	*	0.98					
			R2L	0.65	0.92	0.73	*	0.90					
			Probe	0.85	0.95	0.92	*	0.73					
[64]	Decision Tree classifier	kddcup99.	22-Classes	*	*	*	0.20	*	*	*	*	*	√
[65]	Binary bat algorithm as feature selection and Naïve Bayes classifier	kddcup99.	Normal	*	*	*	*	0.98	√	√	40	*	√
			DoS	*	*	*	*	0.98					
			U2R	*	*	*	*	0.98					
			R2L	*	*	*	*	0.74					
			Probe	*	*	*	*	0.94					
[66]	K-Centroid clustering and Genetic algorithm	Kdd99Cup	Normal	*	*	*	0.85	*	*	*	*	*	√
			DoS	*	*	*	0.92	*					
			U2R	*	*	*	0.96	*					
			R2L	*	*	*	0.77	*					
			Probe	*	*	*	0.32	*					
[67]	Quasi-optimal Algorithm	NSL-KDD	5- Classes	*	*	*	*	*	*	*	*	*	√
[68]	Canonical correlation + feature association impact scale	NSL-KDD	5- Classes	0.98	0.98	0.98	*	*	√	√	*	*	√
[69]	PCA and optimized SVM	NSL-KDD	5- Classes	0.94	0.98	0.96	*	*	√	√	*	*	√
[70]	hybrid PSO feature, selection and SVM	KDDcup99	5- Classes	*	*	*	0.90	*	√	√	*	*	√
[71]	Principle component analysis	ISCX 2012	2- Classes	*	*	*	0.97	*	√	√	*	√	*
[18]	Principal component analysis + SVM	NSL-KDD	Normal	*	*	*	0.95	*	√	√	*	*	√
			DoS	*	*	*	0.99	*					
			U2R	*	*	*	0.99	*					
			R2L	*	*	*	0.70	*					
			Probe	*	*	*	0.99	*					
[72]	Deep learning approach	NSL-KDD	5-Classes	0.85	0.96	0.76	*	*	*	*	*		√
[73]	Sparse auto-encoder and logistic classifier	NSL-KDD	2- Classes	0.85	0.93	*	0.88	*	√	√	*	√	
[35]	ACO algorithm	NSL-KDD	Normal	0.70	0.97	0.98	*	*	√	√	*	*	√
			DoS	0.82	0.99	0.90	*	*					
			U2R	0.7	0.94	0.12	*	*					
			R2L	0.13	0.99	0.23	*	*					
			Probe	0.86	0.74	0.80	*	*					
[74]	Fuzziness and ANN	KDD-99	2- Classes	*	*	*	0.84	*	*	*	*	√	*
[75]	Finite Dirichlet Mixture Model	UNSW-NB15	2- Classes	*	*	*	0.94	*	*	*	*	√	*
[76]	Information gain and RepTree algorithms	UNSW-NB15	2- Classes	*	*	*	0.90	*	√	√	*	√	*
[19]	ABC and AFS algorithms	UNSW-NB15	2- Classes	*	*	*	0.98	*	√	√	√	√	*

TABLE 17. ***(Continued.)*** **A review analysis of the ML-AIDS models.**

[27]	Constrained-optimization-based extreme learning machines	UNSW-NB15	2- Classes	*	*	*	0.83	*	√	√	*	√	*
[28]	GAA	UNSW-NB15	10-Classes	*	*	*	0.99	*	*	*	*		√
[29]	genetic algorithm as a search strategy and logistic regression	KDD-99	5-Classes	*	*	*	*	0.99	√	√	*	*	√
[33]	Machine learning techniques	CIDDS-001	2- Classes	*	*	*	0.99	1.00	*	*	*	√	*
[77]	Skip-gram model classifier	UNSW-NB15	2- Classes	0.99	0.82	*	*	0.91	*	*	*	√	
[78]	Entropy estimation, co-clustering, information gain ratio and extra-trees algorithm	UNSW-NB15	Normal	*	*	*	*	0.94	*	*	*	√	*
			DDoS	*	*	*	*		*	*	*		
[30]	Hyper clique improved binary gravitational search algorithm as FS and SVM	UNSW-NB15	2- Classes	*	*	*	*	0.96	√	√	*	√	*
[79]	Ensemble is applied to feature selection	cidds-001	2- Classes	*	*	*	*	0.93	√	√	*	√	*
[80]	Misuse and information gain ratio	UNSW-NB15	Normal	0.99	0.96	*	0.90	0.85	*	*	*	*	√
			Generic	0.99	0.97	*							
			Exploits	0.69	0.93	*							
			DoS	0.52	0.11	*							
			Probe	0.93	0.81	*							
[81]	Decision trees and genetic algorithms classifier	UNSW-NB15	Normal	0.94	0.97	*	0.47	*	*	*	*	*	√
			Generic	0.98	0.81	*							
			Exploits	0.76	0.76	*							
			Fuzzers	0.74	0.64	*							
			Recon.	0.64	0.46	*							
			DoS	0.20	0.14	*							
			Analysis	0.68	0.20	*							
			Backdoor	0.17	0.67	*							
			Shellcode	0.16	0.36	*							
			Worms	4.00	0.18	*							
[82]	LSTM+ Softmax classifier	UNSW-NB15	5-Classes	0.91	0.92	0.92	0.90	*	√	√	*	*	√
[83]	gradient boosted machine	UNSW-NB15	2- Classes	*	*	*	*	0.99	*	*	*	√	
[84]	iForest outlier, genetic algorithm and random forest classifier	UNSW-NB15	Normal	0.89	0.96	0.93	*	*	√	√	*	*	√
			Generic	0.99	0.96	0.98							
			Exploits	0.75	0.66	0.70							
			Fuzzers	0.94	0.38	0.54							
			Recon.	0.88	0.82	0.85							
			DoS	0.35	0.46	0.39							
			Analysis	0.04	0.06	0.05							
			Backdoor	0.15	0.40	0.21							
			Shellcode	0.35	0.78	0.48							
			Worms	0.77	0.79	0.78							
[85]	ANN classifier	UNSW-NB15	Normal	0.98	0.99	0.99	*	*	*	*	*	√	
			abnormal										
[86]	Gaussian mixture models	DDoS	BENIGN	*	*	*	0.99	*	*	*	*	√	
			DDoS	*	*	*							
[87]	Random forest as feature selection and k-nearest neighbors	CICIDS2017	All types of Attacks	0.99	0.99	0.99	*	0.99	√	√	√	*	√
[88]	Generative adversarial networks and random forest classifier	Web Attacks	BENIGN	0.99	0.99	0.99	*	*	*	*	*	*	√
			Brute Force										
			Sql Injection										
			XSS										
		PortScan	BENIGN	0.99	0.99	0.99	*	*				√	*
			PortScan										
		BOT	BENIGN	0.86	0.53	0.65	*	*				√	*
			Bot										
		Infiltration	BENIGN	1.00	0.60	0.75	*	*				√	*
			Infiltration										
		Patator	BENIGN	*	*	*	*	*				*	√
			FTP-Patator	1.00	0.99	0.99	*	*					
			SSH-Patator	1.00	0.99	0.99	*	*					
		DoS	BENIGN	*	*	*	*	*				*	√
			DoS GoldenEye	0.99	0.99	0.99	*	*					
			DoS Hulk	0.99	0.99	0.99	*	*					
			DoS Slowhttptest	0.99	0.99	0.99	*	*					
			Heartbleed	1.00	1.00	1.00	*	*					
		DDoS	Benign	0.99	0.99	0.99	*	*				√	*
			DDoS										

TABLE 17. ***(Continued.)*** **A review analysis of the ML-AIDS models.**

[89]	Negative selection algorithm and classifiers	CICIDS 2017	All types of Attacks	0.97	0.97	0.97	*	0.98	√	√	*	*	√
[90]	Random forest classifier	CICIDS 2017	DDoS	*	*	*	*	0.96	*	*	*	√	√
[91]	LSTM, CNN and FNN	CICIDS 2017	*	*	*	*	*	0.98	√	*	√	*	*
[92]	Blockchain-based scheme and RNNs	Web Attacks	BENIGN	*	*	*	*	*	√	√	√	*	√
			Brute Force	*	*	*	*	0.85					
			Sql Injection	*	*	*	*	0.82					
			XSS	*	*	*	*	0.91					
		PortScan	BENIGN	*	*	*	*	*				√	*
			PortScan	*	*	*	*	0.98					
		BOT	BENIGN	*	*	*	*	*				√	*
			BoT	*	*	*	*	0.97					
		Infiltration	BENIGN	*	*	*	*	*				√	*
			Infiltration	*	*	*	*	1.00					
		Patator	BENIGN	*	*	*	*	*				*	√
			FTP-Patator	*	*	*	*	0.99					
			SSH-Patator	*	*	*	*	99.9					
		DoS	BENIGN	*	*	*	*	97.5				*	√
			DoS GoldenEye	*	*	*	*	77.1					
			DoS Hulk	*	*	*	*	97.7					
			DoS Slowhttptest	*	*	*	*	94.1					
			Heartbleed	*	*	*	*	1.00					
		DDoS	BENIGN	*	*	*	*	1.00				√	*
			DDoS	*	*	*	*						



	k-NN	ANN	DT	RF	SVM	NB	CNN	k-means	EM	SOM
C1	5	3	6	5	4	1	2	3	1	1
C2	5	3	6	4	4	1	2	3	1	0
C3	5	2	5	4	4	1	2	3	1	0
C4	3	1	3	2	1	1	0	3	1	0
models	5	3	6	5	4	1	2	3	1	1

FIGURE 10. **The overall performance of the algorithms based on different types of attacks.**

Appendix B and the corresponding tables of the algorithms' performance outcomes, the majority of the algorithms are able to detect C1, C2, and C3 attacks. By contrast, only three models of the k-NN, one model of ANN, three models of DT, two models of RF, one model of SVM, three models of k-means, and one model of EM are able to detect C4 attacks. Meanwhile, all CNN and SOM models fail to detect C4 attacks, and SOM models can only detect C1 attacks. In sum, these models achieve high detection rates for C1 and C2 attacks, moderate detection rates for C3 attacks, and low detection rates for C4 attacks. These results can be ascribed to the unbalanced nature of the multiclass CICIDS2017 dataset. Specifically, the majority of the instances in this dataset are of C1 class, followed by C2, and very few instances of C3 and C4 classes. Therefore, detection accuracy alone may not reveal the actual attack detection ability of ML-AIDS models. Figure 10 shows the detection ability of the ML-AIDS algorithms and the related models for four types of attacks.

This paper contributes to the construction of multi-criteria evaluation metrics for multi-class anomaly detection in the CICIDS2017 dataset. A new evaluation and benchmarking methodology for selecting the optimal ML-AIDS diagnostic model based on multi-criteria evaluation metrics is also proposed. This methodology is evaluated by using 31 ML-AIDS models. As shown in Appendix A, several ML-AIDS models obtain better results for the CICIDS2017 dataset compared with certain models, such as those proposed in [82], [83], and [84]. These studies also measure the overall accuracy, precision, and/or recall rates of these models but not for each class of attacks, thereby resulting in unsuitable and unreliable matrices for evaluating the performance of models in addressing imbalanced multi-class attacks in samples with 168,000 normal classes and 2,000 attack classes. Moreover, some models solve the unbalanced data problem by adding or removing some instances from the original dataset, but doing so will affect the behavior of attacks and increase the vulnerability of AIDS. Figure 10 illustrates the

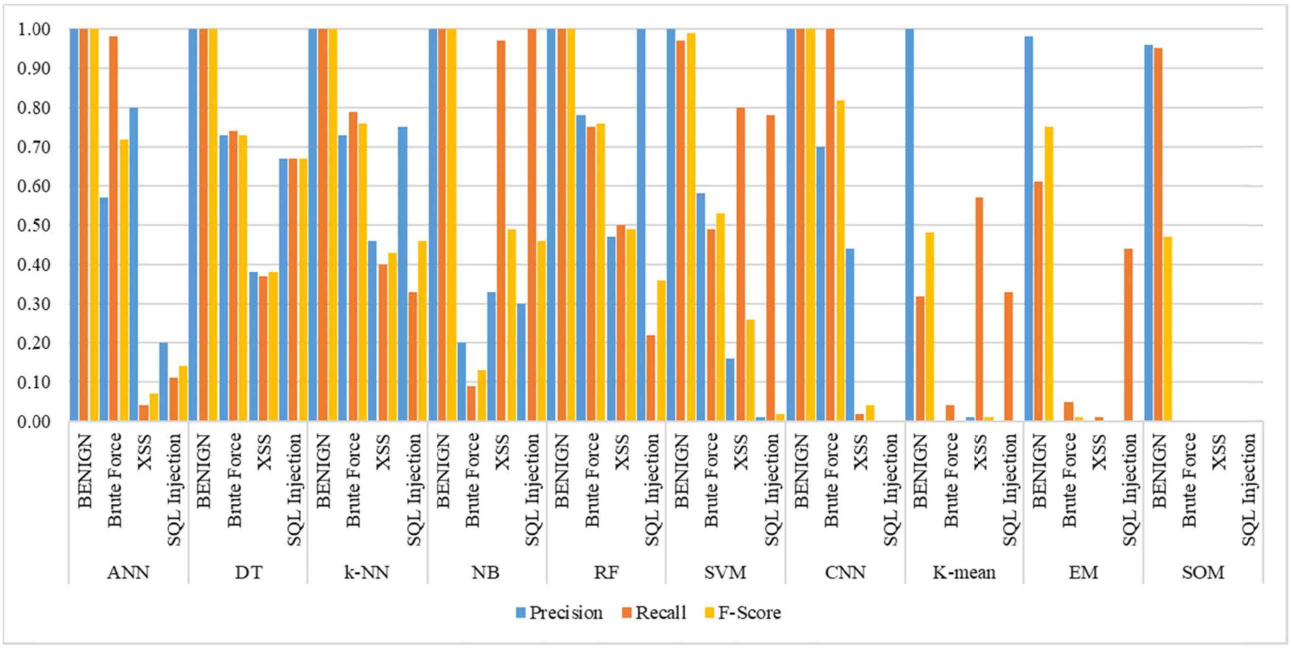


FIGURE 11. **Evaluation of the ML-AIDS.**

ability of these models in handling each type of attack. The performance of ML-AIDS may be further enhanced by applying the appropriate data learning, data standardization, binary feature encoding, and normalization techniques.

VI. CONCLUSION

This paper reviews the previous work on AIDS that employ different datasets. The performance of related ML-AIDS models in detecting attacks on a binary dataset is also tested. These models show limitations in detecting novel types of attacks with multi-classification. Furthermore, most of the related studies use accuracy as their main evaluation metric, thereby preventing them from achieving a fair comparison and evaluation of various ML-AIDSs. To address this problem, this study proposes a benchmarking approach that involves several steps and uses real data to ensure an effective evaluation of AIDS performance based on ML algorithms. The evaluation covers different aspects, various forms of raw network datasets, and recommended performance metrics. Benchmarking tests are also performed to assess the development of effective ML-AIDS by using supervised and unsupervised ML algorithms (i.e., ANN, DT, k-NN, NB, RF, SVM, CNN, EM, K-means, and SOM). The tests are conducted by launching web attacks of CICIDS2017 datasets. Experiment results show the absence of any single ML algorithm that can able to detect all types of web attacks. The K-NN-AIDS, DT-AIDS, and NB-AIDS models achieve excellent performance, whereas the SOM-AIDS and EM-AIDS models achieve poor performance due to their high FP and FN alarms. The proposed benchmarking approach can help researchers in constructing an improved AIDS and comparing their findings with those of this study. Future studies should focus on measuring the impact of feature selection and consider new methodological steps of developing deep learning CNN-AIDS model.

ACKNOWLEDGMENT

The authors would like to thank the INSFORNET, Center for Advanced Computing Technology (C-ACT), Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka.

APPENDIX A

See Table 17.

APPENDIX B

See Figure 11.

REFERENCES

[1] K. Khan, A. Mehmood, S. Khan, M. A. Khan, Z. Iqbal, and W. K. Mashwani, "A survey on intrusion detection and prevention in wireless ad-hoc networks," *J. Syst. Archit.*, vol. 105, May 2020, Art. no. 101701.

[2] 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.

[3] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51691–51713, 2019.

[4] N. Hubballi and V. Suryanarayanan, "False alarm minimization techniques in signature-based intrusion detection systems: A survey," *Comput. Commun.*, vol. 49, pp. 1–17, Aug. 2014.

[5] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.

[6] S. Soheily-Khah, P. Marteau, N. Béchet, "Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the ISCX dataset," in *Proc. ICDIS*, 2017, pp. 219–226.

[7] M. H. Abdulraheem and N. B. Ibraheem, "A detailed analysis of new intrusion detection dataset," *J. Theor. Appl. Inf. Technol.*, vol. 97, no. 17, pp. 4519–4537, 2019.

[8] A. Özgür and H. Erdem, "A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ*, vol. 4, p. e1954v1, Jan. 2016.

[9] C. Azad and V. K. Jha, "Data mining in intrusion detection: A comparative study of methods, types and data sets," *Int. J. Inf. Technol. Comput. Sci.*, vol. 5, no. 8, pp. 75–90, Jul. 2013.

[10] 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.

[11] J. Mchugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, 2000.

[12] B. A. Tama, L. Nkenyereye, S. M. R. Islam, and K.-S. Kwak, "An enhanced anomaly detection in Web traffic using a stack of classifier ensemble," *IEEE Access*, vol. 8, pp. 24120–24134, 2020.

[13] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[14] P. Aggarwal and S. K. Sharma, "Analysis of KDD dataset attributes—Class wise for intrusion detection," *Procedia Comput. Sci.*, vol. 57, pp. 842–851, 2015.

[15] M. K. Siddiqui and S. Naahid, "Analysis of KDD CUP 99 dataset using clustering based data mining," *Int. J. Database Theory Appl.*, vol. 6, no. 5, pp. 23–34, Oct. 2013.

[16] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst. (SPACES)*, Jan. 2015, pp. 92–96.

[17] S. Sahu, "Network intrusion detection system using J48 decision tree," in *Proc. ICACCI*, 2015, pp. 2023–2026.

[18] B. Subba, S. Biswas, and S. Karmakar, "Enhancing performance of anomaly based intrusion detection systems through dimensionality reduction using principal component analysis," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst.*, Nov. 2016, pp. 1–6.

[19] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," *Comput. Netw.*, vol. 136, pp. 37–50, May 2018.

[20] H. Güneş and N. Zincir-Heywood, "Analysis of three intrusion detection system benchmark datasets using machine," in *Proc. Int. Conf. Intell. Secur. Inform.*, 2005, pp. 362–363.

[21] S. Parsazad, E. Saboori, and A. Allahyar, "Fast feature reduction in intrusion detection datasets," in *Proc. MIPRO*, 2012, pp. 1023–1029.

[22] V. Rampure and A. Tiwari, "A rough set based feature selection on KDD CUP 99 data set," *Int. J. Database Theory Appl.*, vol. 8, no. 1, pp. 149–156, Feb. 2015.

[23] M. A. M. Hasan, M. Nasser, S. Ahmad, and K. I. Molla, "Feature selection for intrusion detection using random forest," *J. Inf. Secur.*, vol. 7, no. 3, pp. 129–140, 2016.

[24] S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," Tech. Rep.

[25] K. Kim and M. E. Aminanto, "Deep learning in intrusion detection perspective: Overview and further challenges," in *Proc. Int. Workshop Big Data Inf. Secur.*, Sep. 2017, pp. 1–12.

[26] 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.

[27] C.-R. Wang, R.-F. Xu, S.-J. Lee, and C.-H. Lee, "Network intrusion detection using equality constrained-optimization-based extreme learning machines," *Knowl.-Based Syst.*, vol. 147, pp. 68–80, May 2018.

[28] N. Moustafa, J. Slay, and G. Creech, "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks," *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019.

[29] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Comput. Secur.*, vol. 70, pp. 255–277, Oct. 2017.

[30] M. R. G. Raman, *An Efficient Intrusion Detection Technique Based on Support Vector Machine and Improved Binary Gravitational Search Algorithm*. Amsterdam, The Netherlands: Springer, 2019.

[31] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020.

[32] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, pp. 479–482, Dec. 2018.

[33] A. Verma and V. Ranga, "On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques," *Pertanika J. Sci. Technol.*, vol. 26, no. 3, pp. 1307–1332, 2018.

[34] V. R. Balasaraswathi, M. Sugumaran, and Y. Hamid, "Feature selection techniques for intrusion detection using non-bio-inspired and bio-inspired optimization algorithms," *J. Commun. Inf. Netw.*, vol. 2, no. 4, pp. 107–119, Dec. 2017.

[35] T. Mehmod and H. B. M. Rais, "Ant colony optimization and feature selection for intrusion detection," *Lect. Notes Electr. Eng.*, vol. 387, no. 3, pp. 305–312, 2016.

[36] R. Hippe, J. Dornheim, S. Zeitvogel, and A. Laubenheimer, "Evaluation of machine learning algorithms for anomaly detection," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Service Cyber Secur.*, 2020, p. 65.

[37] A. Ahmad and E. Harjula, "Evaluation of machine learning techniques for security in SDN," in *Proc. IEEE Globecom Workshops*, Sep. 2020, pp. 1–6.

[38] J. Meira, R. Andrade, I. Praça, J. Carneiro, V. Bolón-Canedo, A. Alonso-Betanzos, and G. Marreiros, "Performance evaluation of unsupervised techniques in cyber-attack anomaly detection," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 11, pp. 4477–4489, Nov. 2020.

[39] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Softw. Netw.*, vol. 2017, no. 1, pp. 177–200, 2017.

[40] Q. Dang, "Studying machine learning techniques for intrusion detection systems," *Stud. Mach. Learn. Tech.*, 2019.

[41] A. Gelbukh, F. C. Espinoza, and S. N. Galicia-Haro, "The best neural network architecture," in *Nature-Inspired Computation and Machine Learning* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8857. Jul. 2014.

[42] N. Bhargava and G. Sharma, "Decision tree analysis on J48 algorithm for data mining," *Proc. Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 6, pp. 1114–1119, 2013.

[43] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[44] D. D. Lewis, "Naive (Bayes) at forty: The independence assumption in information retrieval," in *Proc. ECML*, 1998, p. 16.

[45] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, 2016.

[46] J. Jha and L. Ragha, "Intrusion detection system using support vector machine," *Int. J. Appl. Inf. Syst.*, vol. 2013, no. 1, pp. 25–30, 2013.

[47] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electron.*, vol. 9, no. 6, pp. 1–21, 2020.

[48] S. Das. Analytics Vidhya. *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and More*. [Online]. Available: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[49] P. Praveen and B. Rama, "A k-means clustering algorithm on numeric data," *Int. J. Pure Appl. Math.*, vol. 117, no. 7, 2017.

[50] A. A. H. Hassan, W. M. Shah, M. F. I. Othman, and H. A. H. Hassan, "Evaluate the performance of K-means and the fuzzy C-means algorithms to formation balanced clusters in wireless sensor networks," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 2, pp. 1515–1523, 2020.

[51] T. Fv, "The expectation-maximization algorithm," *IEEE Signal Process. Mag.*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[52] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, nos. 1–3, pp. 1–6, May 1998.

[53] T. Eldos, M. Siddiqui, and A. Kanan, "On the KDD'99 dataset: Statistical analysis for feature selection," *J. Data Min. Knowl.*, vol. 3, no. 3, pp. 88–90, 2012.

[54] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Multi-classification of UNSW-NB15 dataset for," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 15, pp. 5094–5104, 2018.

[55] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, *A Detailed Analysis of the CICIDS2017 Data Set*. Springer, 2019.

[56] Z. Pelletier and M. Abualkibash, ''Evaluating the CIC IDS-2017 dataset using machine learning methods and creating multiple predictive models in the statistical computing language R,'' *Science*, vol. 5, no. 2, pp. 187–191, 2020.

[57] M. E. Wahed and O. Farouk, ''Tuning to optimize SVM approach for breast cancer diagnosis with blood analysis data,''

[58] A. Gozzoli. (folyDHUB). *Practical Guide to Hyperparameters Optimization for Deep Learning Models*. [Online]. Available: https://blog.floydhub.com/guide-to-hyperparameters-search-for-deep-learning-models/

[59] J. Hussain, S. Lalmuanawma, and L. Chhakchhuak, ''A two-stage hybrid classification technique for network intrusion detection system,'' *Int. J. Comput. Intell. Syst.*, vol. 9, no. 5, pp. 863–875, Sep. 2016.

[60] S. Masarat, S. Sharifian, and H. Taheri, ''Modified parallel random forest for intrusion detection systems,'' *J. Supercomput.*, vol. 72, no. 6, pp. 2235–2258, Jun. 2016.

[61] S. Sedghi and M. Mirnia, ''Integration bat algorithm with k-means for intrusion detection system,'' *Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 7, pp. 315–319, 2017.

[62] S. Mohammadi, H. Mirvaziri, and M. Ghazizadeh-Ahsaee, ''Multivariate correlation coefficient and mutual information-based feature selection in intrusion detection,'' *Inf. Secur. J. Global Perspective*, vol. 26, no. 5, pp. 229–239, Sep. 2017.

[63] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, ''MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system,'' *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1949–1959, Apr. 2018.

[64] K. Peng, V. C. M. Leung, L. Zheng, S. Wang, C. Huang, and T. Lin, ''Intrusion detection system based on decision tree over big data in fog environment,'' *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–10, Mar. 2018.

[65] P. Natesan, R. R. Rajalaxmi, G. Gowrison, and P. Balasubramanie, ''Hadoop based parallel binary bat algorithm for network intrusion detection,'' *Int. J. Parallel Program.*, vol. 45, no. 5, pp. 1194–1213, Oct. 2017.

[66] B. Chakrabarty, O. Chanda, and M. Saiful, ''Anomaly based intrusion detection system using genetic algorithm and K-centroid clustering,'' *Int. J. Comput. Appl.*, vol. 163, no. 11, pp. 13–17, Apr. 2017.

[67] A. A. Nasr, S. Al-Azhar, M. M. Ezz, and M. Z. Abdulmaged, ''A learnable anomaly detection system using attributional rules,'' *Int. J. Comput. Netw. Inf. Secur.*, vol. 8, no. 11, pp. 58–64, Nov. 2016.

[68] V. Jyothsna and V. V. R. Prasad, ''FCAAIS: Anomaly based network intrusion detection through feature correlation analysis and association impact scale,'' *ICT Exp.*, vol. 2, no. 3, pp. 103–116, Sep. 2016.

[69] S. T. Ikram, ''Improving accuracy of intrusion detection model using PCA and optimized SVM,'' *J. Comput. Inf. Technol.*, vol. 24, no. 2, pp. 133–148, Jun. 2016.

[70] V. Chahar, R. Chhikara, Y. Gigras, and L. Singh, ''Significance of hybrid feature selection technique for intrusion detection systems,'' *Indian J. Sci. Technol.*, vol. 9, no. 48, Jan. 2017.

[71] M. Kakavand, N. Mustapha, A. Mustapha, and M. T. Abdullah, ''Effective dimensionality reduction of payload-based anomaly detection in TMAD model for HTTP payload,'' *KSII Trans. Internet Inf. Syst.*, vol. 10, no. 8, pp. 3884–3910, 2016.

[72] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, ''A deep learning approach for network intrusion detection system,'' in *Proc. EAI Int. Conf. Bio-Inspired Inf. Commun. Technol.*, 2015, pp. 21–26.

[73] S. Gurung, M. K. Ghose, and A. Subedi, ''Deep learning approach on network intrusion detection system using NSL-KDD dataset,'' *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 3, pp. 8–14, 2019.

[74] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, ''Fuzziness based semi-supervised learning approach for intrusion detection system,'' *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.

[75] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, ''Big data analytics for intrusion detection system: Statistical decision-making using finite Dirichlet mixture models,'' pp. 3–31, 2017.

[76] M. Belouch, S. El, and M. Idhammad, ''A two-stage classifier approach using RepTree algorithm for network intrusion detection,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, pp. 389–394, 2017.

[77] R. S. M. Carrasco and M.-A. Sicilia, ''Unsupervised intrusion detection through skip-gram models of network behavior,'' *Comput. Secur.*, vol. 78, pp. 187–197, Sep. 2018.

[78] M. Idhammad, K. Afdel, and M. Belouch, ''Semi-supervised machine learning approach for DDoS detection,'' *Appl. Intell.*, vol. 48, no. 10, pp. 3193–3208, 2018.

[79] W. He, H. Li, and J. Li, ''Ensemble feature selection for improving intrusion detection classification accuracy,'' in *Proc. Int. Conf. Artif. Intell. Comput. Sci.*, Jul. 2019, pp. 28–33.

[80] 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.

[81] D. Papamartzivanos, F. G. Mármol, and G. Kambourakis, ''Dendron: Genetic trees driven rule induction for network intrusion detection systems,'' *Futur. Gener. Comput. Syst.*, vol. 79, pp. 558–574, Feb. 2018.

[82] P. M. Kolte, ''Performance analysis of intrusion detection system utilizing deep learning techniques,'' *J. Gujarat Res. Soc.*, vol. 21, no. 10, pp. 1358–1366, 2019.

[83] B. A. Tama and K. H. Rhee, ''An in-depth experimental study of anomaly detection using gradient boosted machine,'' *Neural Comput. Appl.*, vol. 31, no. 4, pp. 955–965, 2019.

[84] J. Ren, J. Guo, W. Qian, H. Yuan, X. Hao, and H. Jingjing, ''Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms,'' *Secur. Commun. Netw.*, vol. 2019, pp. 1–11, Jun. 2019.

[85] M. Al-Zewairi, S. Almajali, and A. Awajan, ''Experimental evaluation of a multi-layer feed-forward artificial neural network classifier for network intrusion detection system,'' in *Proc. Int. Conf. New Trends Comput. Sci. (ICTCS)*, Oct. 2017, pp. 167–172.

[86] Ö. Cepheli, S. Büyükçorak, and G. K. Kurt, ''Hybrid intrusion detection system for DDoS attacks,'' *J. Electr. Comput. Eng.*, vol. 2016, pp. 1–8, Apr. 2016.

[87] G. Wang, J. Chen, and L. T. Yang, *Effectiveness of Machine Learning Based Intrusion Detection Systems*, vol. 1. Springer, 2018.

[88] J. H. Lee and K. H. Park, ''GAN-based imbalanced data intrusion detection system,'' *Pers. Ubiquitous Comput.*, 2019.

[89] S. Hosseini and H. Seilani, ''Anomaly process detection using negative selection algorithm and classification techniques,'' *Evolving Syst.*, Dec. 2019.

[90] N. Bindra and M. Sood, ''Detecting DDoS attacks using machine learning techniques and contemporary intrusion detection dataset,'' *Autom. Control Comput. Sci.*, vol. 53, no. 5, pp. 419–428, Sep. 2019.

[91] J. Lee, J. Kim, I. Kim, and K. Han, ''Cyber threat detection based on artificial neural networks using event profiles,'' *IEEE Access*, vol. 7, pp. 165607–165626, 2019.

[92] M. A. Ferrag and L. Maglaras, ''DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids,'' *IEEE Trans. Eng. Manag.*, vol. 67, no. 4, pp. 1285–1297, Nov. 2019.

ZIADOON KAMIL MASEER received the master's degree in network security from Pune University, India. He is currently a Senior Lecturer with the College of Science, Tikrit University, Iraq. His research interests include intrusion detection, forensic networks, and cybersecurity.

ROBIAH YUSOF received the Ph.D. degree in network security from Universiti Teknikal Malaysia Melaka. She is currently a Senior Lecturer with UTeM. Her research interests include intrusion detection, malware, network security, and forensic networks.

NAZRULAZHAR BAHAMAN received the Ph.D. degree in network security from UKM, Malaysia. He is currently a Senior Lecturer with UiTM, Malaysia. His research interests include computer networks and security.

SALAMA A. MOSTAFA received the B.Sc. degree in computer science from the University of Mosul, Iraq, in 2003, and the M.Sc. degree in information technology and the Ph.D. degree in information and communication technology from the College of Graduate Studies, Universiti Tenaga Nasional (UNITEN), Malaysia, in 2011 and 2016, respectively. He is currently a Lecturer with the Department of Software Engineering, Universiti Tun Hussein Onn Malaysia (UTHM). His specialization and research interests include autonomous agent, human–computer collaboration, machine learning, optimization, and software quality assurance.

CIK FERESA MOHD FOOZY received the degree in information technology and multimedia and the master's degree in computer science (information security) from Universiti Teknologi Malaysia (UTM), in 2006 and 2009, respectively, and the Ph.D. degree in information security from Universiti Teknikal Malaysia Melaka (UTeM), in 2017. She started her career as a Lecturer at the Department of Information Security and Web Technology, UTHM, in November 2011. She is currently an Active Researcher and has been written and presented a number of papers in conferences and journals.