Leveraging the Training Data Partitioning to Improve Events Characterization in Intrusion Detection Systems

Roberto Saia *, Salvatore Carta, Gianni Fenu, and Livio Pompianu

Department of Mathematics and Computer Science, University of Cagliari, Cagliari, Italy;
Email: salvatore@unica.it (S.C.), fenu@unica.it (G.F.), pompianu.livio@unica.it (L.P.)
*Correspondence: roberto.saia@unica.it (R.S.)

***Abstract*—The ever-increasing use of services based on computer networks, even in crucial areas unthinkable until a few years ago, has made the security of these networks a crucial element for anyone, also in consideration of the increasingly sophisticated techniques and strategies available to attackers. In this context, Intrusion Detection Systems (IDSs) play a primary role since they are responsible for analyzing and classifying each network activity as legitimate or illegitimate, allowing us to take the necessary countermeasures at the appropriate time. However, these systems are not infallible due to several reasons, the most important of which are the constant evolution of the attacks (e.g., zero-day attacks) and the problem that many of the attacks have behavior similar to those of legitimate activities, and therefore they are very hard to identify. This work relies on the hypothesis that the subdivision of the training data used for the IDS classification model definition into a certain number of partitions, in terms of events and features, can improve the characterization of the network events, improving the system performance. The non-overlapping data partitions train independent classification models, classifying the event according to a majority-voting rule. A series of experiments conducted on a benchmark real-world dataset support the initial hypothesis, showing a performance improvement with respect to a canonical training approach.**

***Keywords*—intrusion detection, network security, training data, algorithm**

I. Introduction

The exponential growth of network-based technologies has given rise to a stimulating environment that today most people cannot give up, a scenario that involves countless important applications such as those related to communication systems, finance, education, the food industry, and health, as well as those related to recent technologies, such as the military and civilian drones. It should be observed how in recent years the number of devices connected through networks has increased dramatically due to the massive spread of devices related to the Internet of Things (IoT), which authoritative sector studies estimate will be around sixty billion by 2025 [1].

This enormous network of devices and services expands the audience of potential targets of the attackers, also by taking into account that the COVID-19 pandemic further increases these targets due to the need for many companies to allow their employees to work from home. In such a context, one of the most known and most dangerous attacks is Ransomware [2], directed more and more frequently against public and private objectives, with enormous financial and social costs. For this reason, in these years, in addition to a great commitment of financial and human resources aimed at defining increasingly efficient network services, we have witnessed an equally great commitment as regards the development of techniques and strategies able to grant the security of these services.

However, the high degree of heterogeneity [3] that characterizes this environment makes this operation very difficult, due to both the continuous efforts of the attackers to violate the systems with more and more sophisticated techniques (a case in point is the difficulty of detecting the zero-day attacks [4]), and the problem that many attacks are often characterized by a behavior very similar to that of a legitimate network activity [5], making it difficult to detect them. To face these problems, researchers are constantly looking for more and more efficient Intrusion Detection Systems (IDSs) [6], which are designed using various techniques such as, just to name a few, those based on Machine Learning and Deep Learning [7, 8], Artificial Intelligence [9], Artificial Neural Networks [10–13], Fuzzy Logic [14], often combining more than one to define hybrid solutions [15]. Starting from the consideration that most of the approaches and strategies in the literature related to the IDS domain exploit the entire training set to define the classification model [5, 16–18], we have trivially observed that a training dataset refers to single events in terms of data rows and to the different features that characterize each event in terms of data columns.

Consequently, taking advantage of this data configuration, we have defined a kind of divide-and-conquer strategy, according to which: (i) the dataset is divided into a certain number (experimentally defined) of partitions without overlapping, each of them that refers to certain events and features; (ii) each partition is used to

Manuscript received April 24, 2023; revised May 30, 2023; accepted July 13, 2023; published December 7, 2023.

train an independent classification model; (iii) the final event classification is reached using all model classifications according to a majority-voting rule. In other words, this works proposes a Training Data Partitioning approach aimed to improve the characterization of the network events, then the IDS performance, experimentally verifying what we previously hypothesized and formalized [19], providing the following main contributions:

- Definition of the Training Data Partitioning (TDP) approach aimed to partition a training dataset according to an optimal number of events (data rows) and features (data columns);
- Formalization of the rules needed to partition a training dataset regardless of its number of rows and columns, and to perform the classification process also when it is not possible to apply the majority-voting rule, i.e., when an event receives the same number of votes in the normal and intrusion classes;
- Definition of an intrusion detection algorithm that exploits the proposed TDP approach, analyzing and classifying each network event as normal or intrusion;
- Adoption of an experimental criterion aimed to face overfitting problems, which ensures an effective separation between the data used to select the most performing baseline classifier and the optimal number of dataset partitions, and the data used during the validation process, combining this criterion with a canonical *k*-fold cross-validation one.

Although other works in the literature have already considered partitioning the training data, for instance, to parallelize the training process of a classification or regression model (e.g., distributing the process across many computing nodes [20]), or to face the limits of the hardware resources (e.g., in terms of memory in case of big datasets [21]), to the best of our knowledge, there are no significant works where this operation was aimed at a better characterization of the dataset samples to improve the classification performance.

The remainder of this paper is structured in the following way: Section II discusses the background and the related works concerning the intrusion detection research domain; Section III introduces the formal notation we used along with the formalization of the problem to face; Section IV explains the proposed approach and its implementation; Section V describes the development environment, the adopted real-world dataset, the experimental methodology, and the process of selection of the state-of-the-art baseline classifier, reporting and discussing the experimental results; Section VI concludes this work with some remarks, making mention of future research directions.

II. Background and Related Work

The concept of intrusion understood as an attempt to gain access to the resource of a network illegally was coined many years ago in conjunction with the spread of network services, public and private [22]. Since then, literature has dealt with the aspects related to this potential threat, from the theoretical ones [23] to the practical ones concerning the contexts that have gradually emerged over the years, such as those related to the Internet of Things (IoT), Cloud Computing, Smart Cities, or health-care environments.

The IDSs analyze network traffic to identify illegal access attempts to the network or the improper use of the involved resources, as some attacks do not have the main objective of illegitimately exploiting resources but, for instance, putting them out of service, as it happens with the Denial of Service (DoS) or Distributed Denial of Service (DDoS) attacks often reported by the media [24].

It can base its operation on different modalities, the most common of them are: (i) anomaly-based, according to which it classifies the network activities based on a rules/heuristic-based strategy, then by analyzing their behavior instead querying a database of known patterns [25]; (ii) signature-based, where the new network activity pattern is compared to the known patterns stored in a database, and it is classified based on the basis of this comparison process [26]; (iii) specification-based, according to which the system inspects the involved protocols to detect anomalous sequences that may refer to an attack in progress [27]; (iv) hybrid-based, which does not represent a pure modality but a combination of the previous ones [28].

Discussing the pros and cons of the above methods: an anomaly-based IDS can face attacks such as the *zero-days* ones and, more generally, attacks characterized by an anomalous behavior but it presents a limitation given by its long response-time, which represent a crucial problem in a highly dynamic environment such as the one in which it operates; a signature-based IDS well operates in the context of known attacks and variations of them but its main limitation is given by the inability to inspect the involved protocols and the high computational load required by the classification process; a specification-based IDS can inspect the protocols related to the network activities to detect anomalous behaviors but it is not able to differentiate the legitimate and illegitimate activities that have the same behavior and, in addition, the protocol inspection/tracing capability generates a high computational load; a hybrid-based IDS is obviously characterized by the same pros and cons of the methods that it adopts.

As regards the method of detecting network activities in terms of the number and location of the IDSs, the literature indicates four main categories: (i) Host-based Intrusion Detection Systems (HIDSs) [29], which exploits several hosts to detect the network activity; (ii) Network-based Intrusion Detection Systems (NIDSs) [30], which adopts only a host to detect the network activity; (iii) Network-Node-based Intrusion Detection Systems (NNIDSs) [31], which exploit a single host strategically placed in the network; (iv) Distributed-based Intrusion Detection Systems (DIDSs) [32], which combines the aforementioned categories to detect the network activity.

A further subdivision of the IDSs based on the type of response to the attacks groups them into two broad categories, active and passive [33]: active, when in response to an attack, in addition to recording the activity (log) and forwarding the alerts, the IDS reconfigures the network to counteract the activities of the attackers; passive, when in response to an attack the IDS only records the related network activity, forwarding the needed alerts, without putting in place any active countermeasure.

The discussed classifications of the IDSs based on their operational mode are summarized in Fig. 1.

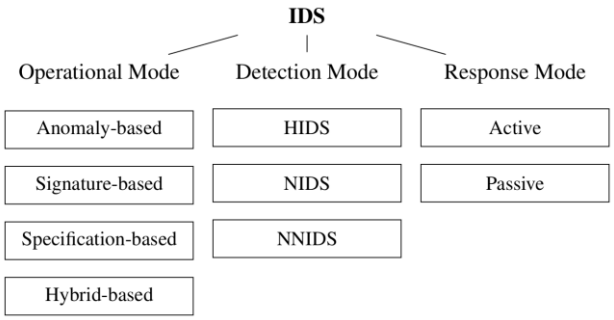

Figure 1. Intrusion detection systems modes.

Similar to other research domains (e.g., Fraud Detection [34]), where the main objective is the identification of numerically rare events, the performance evaluation metrics used in this domain must take into account the high degree of data imbalance that usually characterizes the data, as to get reliable evaluations not biased by the majority class of samples.

In the Intrusion Detection domain, the minority class is the intrusion one and the IDSs usually operate according to a binary criterion [35], classifying each network event as *normal* or intrusion. It means that the metrics to be considered are those suitable for the evaluation of binary classifiers such as, for example, those based on the *confusion matrix*, i.e., a matrix 2×2 that reports the number of True Negatives (TN), False Negatives (FN), True Positives (TP), and False Positives (FP), as shown in Fig. 2.

Some examples of confusion-matrix-based metrics widely used in this research field are the Accuracy, the True Positive Rate, and the True Negative Rate but to deal with the imbalance problem such metrics are often flanked by other ones [36] that are not influenced by this data characteristic, such as those based on the Receiver Operating Characteristic (ROC) curve, as the Area Under the Receiver Operating Characteristic Curve (AUC).

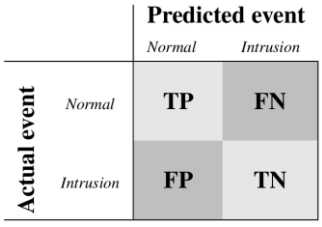

Figure 2. Confusion matrix.

III. Notation and Problem Definition

Specifying that we used the notation $|E|$ to indicate the cardinality of the set E, we denote as $E = \{e_1, e_2, \dots, e_N\}$ a set of network events composed of a subset $E^+ = \{e_1^+, e_2^+, \dots, e_X^+\}$ of *normal* events ($E^+ \subseteq E$), a subset $E^- = \{e_1^-, e_2^-, \dots, e_Y^-\}$ of *intrusion* events ($E^- \subseteq E$), and a subset of $\hat{E} = \{\widehat{e_1}, \widehat{e_2}, \dots, \widehat{e_M}\}$ unclassified events ($\hat{E} \subseteq E$).

According to the above notation, we have $E = \left(E^+ \cup E^- \cup \hat{E}\right)$, where each event $e \in E$ is characterized by a series of features $F = \{f_1, f_2, \dots, f_W\}$, and it can only belong to one of the two classes in $C = \{normal, intrusion\}$.

In addition, we denote as $T = \{e_1, e_2, \dots, e_K\}$ (given by $E^+ \cup E^-$) the training set, which can be partitioned into $P = \{p_1, p_2, \dots, p_Z\}$ partitions, according to the operation as $P_{(ER,FC)}$, with ER the number of *Event Rows*, and FC the number of *Feature Columns*, then $|P| = Z = (ER \times FC)$.

Given the set E, this means that: each partition can be composed of $\frac{N}{ER}$ events and $\frac{W}{FC}$ features, since $|E| = N$ and $|F| = W$; the bounds of ER is $1 \leq ER \leq |T|$ and that of FC is $1 \leq FC \leq |F|$, then the pair of values $ER = FC = 1$ indicates the canonical data configuration without partitioning.

It should be noted that each partition defined according to the ER value must contain samples that belong to both classes in C, to allow us the training of the evaluation model. Since we consider the intrusion detection problem in binary terms, according to the provided notation we can formalize it as shown in Eq. (1), denoting the intrusion detection approach as Ξ, and the evaluation function (it returns 1 if the classification is correct, 0 otherwise) of an event $\hat{e}$ as $eval(\hat{e}, \Xi)$, so our problem can be expressed in terms of maximization of the η value ($\left|\hat{E}\right|$ represents the η upper bound).

$$\max_{0 \leq \eta \leq \left|\hat{E}\right|} \eta = \sum_{m=1}^{|\hat{E}|} eval(\hat{e}_m, \Xi) \qquad (1)$$

This means that the maximum value returned by Eq. (1) can only be achieved by an ideal intrusion detection approach able to correctly classify all the events, i.e., the goal to aim for.

IV. Proposed Approach

The problem previously defined in Eq. (1) needs to be transposed into the proposed TDP approach, then revised by subdividing the classification process according to the number of training set partitions.

It means that the evaluation process is now composed of Z sub-processes (i.e., $|P| = Z$), then the Ξ intrusion detection approach is executed Z times, and the final classification of the event depends on the result of each execution: for example, assuming we have the values $K = 4$, $W = 4$, $ER = 2$, $FC = 2$, which means that we subdivide the training set T into $|P| = Z = (2 \times 2) = 4$ partitions, each of them composed of $\frac{K}{ER} = \frac{4}{2} = 2$ events and $\frac{W}{FC} = \frac{4}{2} = 2$ features. In this way, the process of

training of each evaluation model m uses, respectively, the events and features in each p_1, p_2, p_3, p_4 partitions of Eq. (2), generating four m_1, m_2, m_3, and m_4 evaluation models.

$$P_{(2,2)} = \boxed{\begin{matrix} p_1 & p_2 \\ p_3 & p_4 \end{matrix}} \Rightarrow \boxed{\begin{matrix} f_{1,1} & f_{2,1} f_{3,1} & f_{4,1} \\ f_{1,2} & f_{2,2} f_{3,2} & f_{4,2} \\ f_{1,3} & f_{2,3} f_{3,3} & f_{4,3} \\ f_{1,4} & f_{2,4} f_{3,4} & f_{4,4} \end{matrix}} \Rightarrow \boxed{\begin{matrix} m_1 & m_2 \\ m_3 & m_4 \end{matrix}} \quad (2)$$

A. Data Partitioning Criteria

Given that the number of partitions determined by the FC and ER values may not exactly partition the features (data columns) and events (data rows), i.e., when $(|F| \; mod \; FC) \neq 0$ or $(|T| \; mod \; ER) \neq 0$, we need to define a criterion able to overcome this problem.

Starting from the notation provided in Section III, we denote $\mu_1 = (|F| \; mod \; FC)$ and $\mu_2 = (|T| \; mod \; ER)$, formalizing the needed criterion, respectively for the set F and T, as shown in Eq. (3). Intending to not bias the original information, such a criterion adopts two different strategies: concerning the set F, it duplicates the last feature column μ_1 times, whereas concerning the set F, it duplicates the last μ_2 event rows, as to avoid adding events that belong to the same class in C.

$$\begin{aligned} & F \quad \Rightarrow \quad F = \{f_1, f_2, \dots, f_W, f_{W+1}, f_{W+2}, \dots, f_{W+\mu_1}\} \\ & \text{with } f_{W+1} = f_{W+2} = \dots = f_{W+\mu_1} = f_W \end{aligned} \quad (3)$$

$$\begin{aligned} & T \quad \Rightarrow \quad T = \{e_1, e_2, \dots, e_K, e_{K+1}, e_{K+2}, \dots, e_{K+\mu_2}\} \\ & \text{with } e_{K+1} = e_K, e_{K+2} = e_{K-1}, \dots = e_{K+\mu_2} = e_{K-\mu_2} \end{aligned}$$

Another criterion is instead aimed to solve the problems that happen when it is not possible to apply the majority-voting rule since the two possible destination classes (i.e., *normal* and *intrusion*) receive the same number of votes.

It introduces a discriminating classification obtained by defining a further evaluation model trained on the entire E set (canonical approach), generating in this way $c_1, c_2, \dots, c_Z, c_{Z+1}$ classifications.

By way of example, assuming a scenario with ER = FC = 2, which generates the m_1, m_2, m_3, m_4 classification models that lead toward the c_1, c_2, c_3, c_4 classifications for an event, where we assume that two of them are *normal* and the other two are an *intrusion*, the final classification is reached by adding the classification c_5 obtained through a further evaluation model trained on the entire set T.

The formalization of all possible classification cases is shown in Eq. (4), where c_1 and c_2 indicate, respectively, *normal* and *intrusion* elements in set C.

We experimented that such two criteria do not significantly alter the involved processes, also because they are applied to both the training and test set, and for simplification reasons, from now on, we will consider their application as an internal preprocessing step applied during the data partition process.

$$\begin{aligned} & c_1, \; if \sum_{i=1}^{Z} \omega(c_i, c_1) > \sum_{i=1}^{Z} \omega(c_i, c_2) \\ & c_2, \; if \sum_{i=1}^{Z} \omega(c_i, c_1) < \sum_{i=1}^{Z} \omega(c_i, c_2) \\ & c_1, \; if \sum_{i=1}^{Z} \omega(c_i, c_1) = \sum_{i=1}^{Z} \omega(c_i, c_2) \wedge c_{Z+1} = c_1 \\ & c_2, \; if \sum_{i=1}^{Z} \omega(c_i, c_1) = \sum_{i=1}^{Z} \omega(c_i, c_2) \wedge c_{Z+1} = c_2 \end{aligned}$$

with

$$\omega(a, b) = \begin{cases} 0, if \; a \neq b \\ 1, if \; a = b \end{cases} \quad (4)$$

B. Data Classification Algorithm

The criteria previously formalized allow us to apply the proposed TDP approach to any dataset, according to Algorithm 1, which performs the classification of the new network events. It takes as input a baseline classification algorithm β, the set of classified events T (i.e., the training set), the set $\hat{E}$ of unclassified events, the ER and FC values that determine the data partitioning, returning as output the classification of all the events in the set $\hat{E}$.

It should be noted that Algorithm 1 is formalized in terms of pseudocode using functions that explicitly refer to canonical (e.g., trainModel and getMajorityVotingClassifications) and related to our approach (e.g., getPartitions and getEventClass), which have been previously formalized.

Algorithm 1. TDP classification

Require: β=Baseline classification algorithm, T=Training set, Ê=Events to evaluate, ER=Event rows, FC=Feature columns

Ensure: Ē=Classification of all the events in the set Ê

```
1.  procedure getTDP(β, T, Ê, ER, FC)
2.    if(ER x FC) is even) then
3.      m'' ← trainModel(β, T)
4.    end if
5.    p ← getPartitions(T, ER, FC)
6.    for all p in P do
7.      m ← trainModel(β, p)
8.      M.add(m)
9.    end for
10.   for all ê in Ê do
11.     P''← getPartitions(ê, ER, FC)
12.     for all m in M do
13.       c ← getEventClass(m'', ê)
14.       C.add(c)
15.     end for
16.     if(ER x FC) is even then
17.       c'' ← getEventClass(m'', ê)
18.       C.add(c'')
19.     end if
20.     Ē.add(getMajorityVotingClassification(ê, C)
21.   end for
```

In more detail: at steps from 2 to 4 an evaluation model is trained by using the entire set E, if the numbers of partitions (i.e., Z = |ER × FC|) is even; at step 5 the training test T is processed in order to define the partitions, according to the ER and FC values; for each partition, at steps from 6 to 9, an evaluation model of the algorithm β is trained; the classification of the events in $\hat{E}$ is performed from step 10 to 21, where at step 11 the features of the unevaluated event $\hat{e}$ are divided into partitions, according to the ER and FC values, at steps from 12 to 15 each evaluation models in M provide an event classification, and an additional canonical classification based on the model trained at step 3 is added to the set C when if the number of involved partitions is even (steps from 16 to 19); at step 20 the event $\hat{e}$ is classified according the majority-voting rule, and the classification is added to the set κ, which is returned by the algorithm at step 22, with the classification of all the events in the input set (i.e., $\hat{E}$).

Computational Complexity. The TDP approach does not present an excessive computational complexity [37], also considered that the involved tasks can be distributed over several processes and/or machines, for example using frameworks such as MapReduce [38].

C. Approach Architecture

All the elements previously described that compose the proposed TDP approach are summarized in the high-level description of Fig. 3, according to the Algorithm 1 processes. It starts from a preprocessing step, where the training set T and the unclassified events set $\hat{E}$ are managed through the criteria formalized in Section IV-A.

Subsequently, the data partitioning is carried out based on the ER and FC values, and the related classification models are trained and used in the last step, where all the classifications decide the event classification using the majority-voting rule.

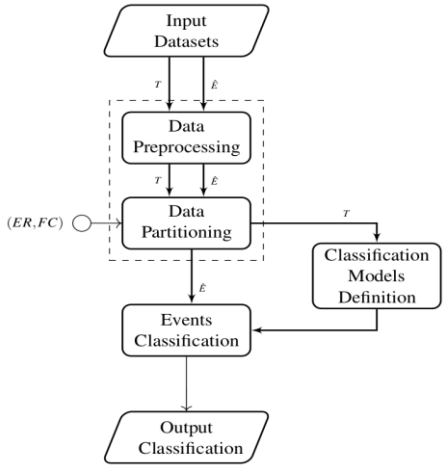

Figure 3. TDP high-level architecture.

V. Experiments

We carried out all the experiments using an 11th Generation *Intel* Core i7-1165G7, 2.80GHz × 8 CPUs machine, with 16 GB of RAM, *Linux* operating system with kernel 5.10.0-14-amd64, and the *Python* language with the *Scikit-learn* (http://scikit-learn.org) library. In addition, to ensure the reproduction of the experiments, we fixed the pseudo-random number generator seed of the *Scikit-learn* to 1.

A. Dataset

The real-world dataset used for the validation process, the NSL-KDD (https://www.unb.ca/cic/datasets/nsl.html), can be considered a benchmark dataset in the intrusion detection field [39]. Considering that many works in the literature adopted it, the use of this dataset allows anyone to compare them.

The network events in the dataset are related to the UDP, TCP, and ICMP protocol activity, including many types of attacks. The NSL-KDD dataset release contains 148,517 events (77,054 normal events and 71,463 intrusion events). Each event contains 43 features (4 categorical, 6 binary, 23 discrete, and 10 continuous), involving four different classes of network attacks (120 Privilege Escalation, 53,387 Denial of Service, 14,077 Remote Scanning, and 3,879 Remote Access). Tables I and II report the main characteristics of the dataset.

TABLE I. NSL-KDD Dataset Overview

Total Events $\lvert E^+\rvert + \lvert E^-\rvert$	Normal $\lvert E^+\rvert$	Intrusion $\lvert E^-\rvert$	Features $\lvert F\rvert$	Classes $\lvert C\rvert$
148,517	77,054	71,463	43	2

TABLE II. NSL-KDD Network Attacks Classes

Class	Attacks	Type	Description
1	120	Privilege Escalation	Aimed to obtain a privileged access as unprivileged user (e.g., Buffer overflow, Rootkit, Perl, Loadmodule, Xterm, Sqlattack, and Ps).
2	53,387	Denial of Service	Aimed to block a service/system generating an huge number of normal network activities (e.g., Mail-bomb, Land, Back, Pod, Smurf, Neptune, Teardrop, Udpstorm, Processtable, Worm, and Apache2).
3	14,077	Remote Scanning	Aimed to get details about a service/system through a series of invasive and non-invasive approaches (e.g., Nmap, IPsweep, Satan, Portsweep, Saint, and Mscan).
4	3,879	Remote Access	Aimed to obtain a remote system access by exploiting different techniques (e.g., Ftp write, Guess password, Imap, Phf, Warez-master, Multihop, Xsnoop, Xlock, Snmpguess, Httptunnel, Snmpgetattack, Named, and Sendmail).

B. Methodology

As preprocessing steps: (i) we transformed the dataset's categorical features into a numerical one; (ii) we introduced a class feature that classifies each event according to a binary criterion, using the value 0 for the normal events, and the value 1 for the intrusion ones, removing the original attack-type label for each event.

We carried out all the experiments according to a preliminary division of the dataset into two parts:

1. An in-sample part (80%), used to detect the most performing classification baseline algorithm in the context of the canonical and TDP approaches, and to detect the optimal number of TDP partitions (i.e., ER and FC values);
2. An out-of-sample part (20%), used to perform the validation process, where we compare the performance of the TDP approach to that of the canonical one.

Such a methodology avoids overfitting since it provides an effective separation between the data related to the different data processes, where we also apply a canonical *k*-fold cross-validation criterion with $k = 5$.

For performance assessment, we use three metrics widely used in this domain: the True Negative Rate (TNR), the True Positive Rate (TPR), and the Area Under the Receiver Operating Characteristic Curve (AUC).

C. Baseline Algorithms

We selected the most performing state-of-the-art classification algorithm by testing the intrusion detection performance of several algorithms widely used for this task in the literature, using only the *in-sample* part of the dataset.

TABLE III. BASELINE ALGORITHMS PARAMETERS

Algorithm	Parameters
Adaptive Boosting	algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=50, random_state=1
Decision Tree	class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, presort=False, random_state=1, splitter='best'
Gradient Boosting	criterion='friedman_mse', init=None, learning_rate=0.1, loss='deviance', max_depth=3, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=100, presort='auto', random_state=1, subsample=1.0, verbose=0, warm_start=False
Multilayer Perceptron	activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9, beta_2=0.999, early_stopping=False, epsilon=1e-08, hidden_layer_sizes=(100,), learning_rate='constant', learning_rate_init=0.001, max_iter=200, momentum=0.9, nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1, verbose=False, warm_start=False
Random Forests	bootstrap=True, class_weight=None, criterion='gini', max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1, oob_score=False, random_state=1, verbose=0, warm_start=False

The algorithms we take into consideration are Adaptive Boosting [40], Decision Tree [41], Gradient Boosting [42], Multilayer Perceptron [43], and Random Forests [44]. The algorithms parameters are reported in Table III, whereas the measured performance is shown in Table IV, where the results indicate Random Forests as the most performing algorithm (the best performance is highlighted in bold).

TABLE IV. IN-SAMPLE ALGORITHMS PERFORMANCE

Algorithm	TNR	TPR	AUC
Adaptive Boosting	0.9518	0.9565	0.9539
Decision Tree	0.9747	0.9738	0.9737
Gradient Boosting	0.9747	0.9738	0.9737
Multilayer Perceptron	0.9141	0.9003	0.9031
Random Forests	**0.9849**	**0.9900**	**0.9877**

D. Optimal Data Partitioning

The experiments performed in this step are aimed to detect the optimal number of data partitions for the proposed TDP approach, then the ER and FC values. The process took place using the Random Forests algorithm previously selected and the in-sample part of the dataset, where during the experiments we applied the *k*-fold cross-validation criterion based on five folds (i.e., $k = 5$).

The optimal parameters were selected by considering all the metrics discussed in Section V-B (i.e., TNR, TPR, and AUC).

The results indicate ER = 1 and FC = 2 as optimal values in the context of the RF algorithm we selected in Section V-C. This is visible in Table V, where to simplify, we report only the most significant range of values (the best performance is highlighted in bold).

TABLE V. IN-SAMPLE DATA PARTITIONING VALUES TUNING

ER	FC	TNR	TPR	AUC
1	1	0.9849	0.9900	0.9877
1	2	**0.9993**	**0.9998**	**0.9995**
1	3	0.9992	0.9961	0.9978
2	1	0.9984	0.9961	0.9989
2	2	0.9987	0.9997	0.9991
2	3	0.9938	0.9968	0.9950
3	1	0.9983	0.9991	0.9987
3	2	0.9984	0.9995	0.9989
3	3	0.9947	0.9962	0.9953

It should be noted that the pair of values ER = 1 and FC = 1 represents the canonical data configuration without data partitioning.

E. Results

After identifying the most performing baseline algorithm (Section V-C) and the optimal parameters for partitioning data under the proposed approach (Section V-D), performing both the processes in the in-sample part of the dataset, the next step is to compare our TDP approach with the canonical one. We perform this operation in the out-of-sample part of the dataset, as to obtain reliable results not influenced by overfitting.

In addition, we made all the experiments according to the *k*-fold cross-validation criterion with $k = 5$. The results are reported in Table VI, where the best performance is highlighted in bold.

TABLE VI. OUT-OF-SAMPLE PERFORMANCE

Approach	Algorithm	TNR	TPR	AUC
Canonical	Random Forests	0.9868	0.9870	0.9867
TDP	Random Forests	**0.9913**	**0.9933**	**0.9924**

F. Discussion

Based on the performed experiments and the related results we can make the following considerations:

- The process aimed at defining the data partitioning values (i.e., ER and FC), carried out using the in-sample part of the dataset, has identified ER = 1 and FC = 2 as optimal data partitioning values, indicating that no partitioning will be made in terms of events;
- The performance comparison between the canonical approach (i.e., based on a classification model trained on the entire training set) and the proposed TDP one, which we made using the out-of-sample part of the dataset, shows that TDP outperforms the canonical approach;
- In more detail, the results of the performance comparison reported in Table VI indicate that TDP outperforms the canonical approach in terms of all the considered evaluation metrics (i.e., TNR, TPR, and AUC), supporting the hypothesis behind this work;
- Despite the improvements in performance may appear modest, the results are promising since in the intrusion detection domain there are many scenarios [45, 46] where the IDSs operate continuously (24 hours a day and 365 days a year), then also minor improvements lead to the detection of a significant number of intrusion events, with all the consequent advantages;
- Just by way of example, even considering a limited number of events such as those present in the used NSL-KDD dataset, the performance gain we have achieved in terms of TPR and TNR (respectively, +0.0063 and +0.0045) made it possible to correctly identify a further 485 legitimate activities (normal events) and 321 attacks (intrusion events);
- An additional performance analysis, aimed at verifying that the increase in performance in terms of TPR and TNR did not depend on the increasing of false positives and/or false negatives cases (although the measured improvement in terms of AUC can rule this out), showed that the proposed approach also gets an improvement in this sense, with a False Positive Rate value that going down from 0.0132 to 0.0087, and a False Negative Rate value that going down from 0.0130 to 0.0067;
- The experimental results demonstrate that the training data partitioning on which the proposed approach relies can improve the performance of an IDS, and this is also further supported by the adoption of a double validation criterion (*in-sample*}/out-of-sample and *k*-fold cross-validation) during the experiments, which provides us reliable results not influenced by overfitting;
- In other words, the performed experiments have shown the effectiveness of such an approach, where non-overlapping data partitions train independent classification models combined according to a majority-voting rule, and in this regard, it is necessary to consider that we operate in a challenging domain characterized by a high level of data heterogeneity, a similarity between some legitimate and illegitimate classes of events, as well as a limited room for improvement given by the good performance achieved by the state-of-the-art approaches;
- In conclusion, since the proposed approach can improve the performance of a baseline algorithm, it can be adopted in many state-of-the-art approaches that exploit this algorithm to improve their performance.

VI. CONCLUSION AND FUTURE WORK

Given the increasing dependence of people on public and private network services, the security of such networks has long become a crucial requirement, albeit very difficult to ensure due to the high dynamism of the involved factors. In this scenario, the IDSs play a primary role since they analyze and classify network events to identify any unauthorized use. These systems must face the continuous evolution of the techniques used by the attackers, with the complication that many attacks have a behavior very similar to that of normal network activities.

To improve the characterization of the network events to mitigate this kind of problem, this work proposed a TDP approach that relies on the partitioning of training data in terms of time (event rows) and characteristics (feature columns). The event classification is made using multiple classification models trained individually on each of these partitions, whose results define the event classification according to a majority-voting rule.

The validation process, the results of which are to be considered reliable thanks to the adoption of a double criterion (in-sample/out-of-sample and k-fold cross-validation) aimed at avoiding overfitting problems, proved the validity of the proposed TDP approach, showing improvements in terms of TNR, TPR, and AUC, compared to the canonical approach.

In future work, we plan to experiment with the proposed approach in the context of different domains/datasets, as to explore the relations between the optimal number of partitions and the involved data type and domain. This is because, although the double validation criterion we have adopted (*in-sample*/*out-of-sample* e *k*-fold cross-validation) guarantees us that the results obtained do not depend on randomness (*k*-fold cross-validation) or overfitting problems (*in-sample*/*out-of-sample*), investigating the relationships between the optimal number of partitions and the context of the data provides elements for further improvements of the approach.

CONFLICT OF INTEREST

The authors state that there are no conflicts of interest in the publication of this research.

AUTHOR CONTRIBUTIONS

Conceptualization, R.S., L.P., S.C., and G.F.; data curation, R.S. and L.P.; formal analysis, R.S.; methodology, R.S., L.P., S.C., and G.F.; resources, R.S., L.P., S.C., and G.F.; supervision, S.C. and G.F.; validation, R.S., L.P., and S.C.; writing, original draft, R.S.; writing, review and editing, R.S., L.P., and S.C.; all authors had approved the final version.

FUNDING

This research was partially funded and supported by Visioscientiae Srl.

REFERENCES

[1] A. N. Mian, S. W. H. Shah, S. Manzoor, A. Said, K. Heimerl, and J. Crowcroft, "A value-added IoT service for cellular networks using federated learning," *Computer Networks*, vol. 213, 109094, 2022.

[2] A. Tandon and A. Nayyar, "A comprehensive survey on ransomware attack: A growing havoc cyberthreat," *Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing*, vol. 839, pp. 403–420, 2019.

[3] M. A. Khan and J. Kim, "Toward developing efficient Conv-AE-based intrusion detection system using heterogeneous dataset," *Electronics*, vol. 9, no. 11, 1771, 2020.

[4] K. Radhakrishnan, R. R. Menon, and H. V. Nath, "A survey of zero-day malware attacks and its detection methodology," in *Proc. TENCON 2019, 2019 IEEE Region 10 Conference (TENCON)*, IEEE, 2019, pp. 533–539.

[5] S. Carta, A. S. Podda, D. R. R. Recupero, and R. Saia, "A local feature engineering strategy to improve network anomaly detection," *Future Internet*, vol. 12, no. 10, 177, 2020.

[6] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[7] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *Applied Sciences*, vol. 9, no. 20, 4396, 2019.

[8] M. B. Pranto, M. H. A. Ratul, M. M. Rahman, I. J. Diya, and Z.-B. Zahir, "Performance of machine learning techniques in anomaly detection with basic feature selection strategy—A network intrusion detection system," *Journal of Advances in Information Technology*, vol. 13, no. 1, pp. 36–44, February 2022.

[9] V. Kanimozhi and T. P. Jacob, "Artificial intelligence based network intrusion detection with hyper-parameter optimization tuning on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing," in *Proc. 2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2019, pp. 33–36.

[10] T.-T.-H. Le, Y. Kim, H. Kim *et al*., "Network intrusion detection based on novel feature selection model and various recurrent neural networks," *Applied Sciences*, vol. 9, no. 7, 1392, 2019.

[11] M. A. Rassam and M. A. Maarof, "Artificial immune network clustering approach for anomaly intrusion detection," *Journal of Advances in Information Technology*, vol. 3, no. 3, pp. 147–154, 2012.

[12] O. Al-Jarrah and A. Arafat, "Network intrusion detection system using neural network classification of attack behavior," *Journal of Advances in Information Technology*, vol. 6, no. 1, pp. 1–8, 2015.

[13] S. P. Sithungu and E. M. Ehlers, "GAAINet. A generative adversarial artificial immune network model for intrusion detection in industrial IoT systems," *Journal of Advances in Information Technology*, vol. 13, no. 5, pp. 456–461, 2022.

[14] B. Kavitha, S. Karthikeyan, and P. S. Maybell, "Emerging intuitionistic fuzzy classifiers for intrusion detection system," *Journal of Advances in Information Technology*, vol. 2, no. 2, pp. 99–108, 2011.

[15] M. L. B. Meyer and Y. Labit, "Combining machine learning and behavior analysis techniques for network security," in *Proc. 2020 International Conference on Information Networking (ICOIN)*, IEEE, 2020, pp. 580–583.

[16] R. Saia, S. Carta, and D. R. Recupero, "A probabilistic-driven ensemble approach to perform event classification in intrusion detection system," in *Proc. the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018)*, 2018, pp. 139–146.

[17] R. Saia, S. Carta, D. R. Recupero, and G. Fenu, "A feature space transformation to intrusion detection systems," in *Proc. the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2020)*, 2020, pp. 137–144.

[18] R. Saia, S. Carta, D. R. Recupero, G. Fenu, and Ma. Stanciu, "A Discretized Extended Feature Space (DEFS) model to improve the anomaly detection performance in network intrusion detection systems," in *Proc. the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, 2019, pp. 322–329.

[19] R. Saia, A. S. Podda, G. Fenu, and R. Balia, "Decomposing training data to improve network intrusion detection performance," in *Proc. the 13th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2021)*, 2021, pp. 241–248.

[20] S. Khalifa, P. Martin, and R. Young, "Label-aware distributed ensemble learning: A simplified distributed classifier training model for big data," *Big Data Research*, vol. 15, pp. 1–11, 2019.

[21] H.-F. Yu, C.-J. Hsieh, K.-W. Chang, and C.-J. Lin, "Large linear classification when data cannot fit in memory," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 4, pp. 1–23, 2012.

[22] J. P. Anderson. (1980). Computer security threat monitoring and surveillance. [Online]. Available: https://seclab.cs.ucdavis.edu/projects/history/papers/ande80.pdf

[23] S. Axelsson. (2000). Intrusion detection systems: A survey and taxonomy. Technical Report. [Online]. Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7a15948bdcb530e2c1deedd8d22dd9b54788a634

[24] J. F. Balarezo, S. Wang, K. G. Chavez, A. Al-Hourani, and S. Kandeepan, "A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks," *Engineering Science and Technology, an International Journal*, vol. 31, 101065, 2021.

[25] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE Access*, vol. 9, pp. 22351–22370, 2021.

[26] M. Masdari and H. Khezri, "A survey and taxonomy of the fuzzy signature-based intrusion detection systems," *Applied Soft Computing*, vol. 92, 106301, 2020.

[27] L. O. Nweke, "A survey of specification-based intrusion detection techniques for cyber-physical systems," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 5, 2021.

[28] A. N. Cahyo, A. K. Sari, and M. Riasetiawan, "Comparison of hybrid intrusion detection system," in *Proc. 2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE)*, IEEE, 2020, pp. 92–97.

[29] S. Jose, D. Malathi, B. Reddy, and D. Jayaseeli, "A survey on anomaly based host intrusion detection system," *Journal of Physics: Conference Series*, vol. 1000, 012049, 2018.

[30] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and AdaBoost algorithms," *Journal of King Saud University-Computer and Information Sciences*, vol. 31, no. 4, pp. 541–553, 2019.

[31] S. Potluri and C. Diedrich, "High performance intrusion detection and prevention systems: A survey," in *Proc. ECCWS2016, the 15th European Conference on Cyber Warfare and Security*, 2016.

[32] K. K. R. Amrita, "A hybrid intrusion detection system: Integrating hybrid feature selection approach with heterogeneous ensemble of intelligent classifiers," *International Journal of Network Security*, vol. 20, no. 1, pp. 41–55, 2018.

[33] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion detection systems: A cross-domain overview," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3639–3681, 2019.

[34] R. Saia, S. Carta *et al*., "A frequency-domain-based pattern mining for credit card fraud detection," in *Proc. the 2nd International*

Conference on Internet of Things, Big Data and Security (IoTBDS 2017), 2017, pp. 386–391.
[35] H. M. Chuang, H. Y. Huang, F. Liu, and C. H. Tsai, "Classification of intrusion detection system based on machine learning," in *Proc. the Second International Conference on Cognitive Cities, IC3 2019*, Springer, 2019, pp. 492–498.
[36] N. Munaiah, A. Meneely, R. Wilson, and B. Short. (2016). Are intrusion detection studies evaluated consistently? A systematic literature review. Technical Report of Rochester Institute of Technology. [Online]. Available: http://scholarworks.rit.edu/article/1810
[37] S. Bae, "Big-O notation," in *JavaScript Data Structures and Algorithms*, Springer, 2019, pp. 1–11.
[38] P. R. Kanna and P. Santhi, "Hybrid intrusion detection using MapReduce based black widow optimized convolutional long short-term memory neural networks," *Expert Systems with Applications*, vol. 194, 116545, 2022.
[39] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," *Expert Systems with Applications*, vol. 124, pp. 196–208, 2019.
[40] Y. Zhou, T. A. Mazzuchi, and S. Sarkani, "M-AdaBoost-A based ensemble system for network intrusion detection," *Expert Systems with Applications*, vol. 2020, 113864, 2020.
[41] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proc. International Conference on Information and Communication Technology for Intelligent Systems*, Springer, pp. 207–218, 2017.
[42] P. Verma, S. Anwar, S. Khan, and S. B Mane, "Network intrusion detection using clustering and gradient boosting," in *Proc. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2018, pp. 1–7.
[43] K. Chisholm, C. Yakopcic, M. S. Alam, and T. M. Taha, "Multilayer perceptron algorithms for network intrusion detection on portable low power hardware," in *Proc. 2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2020, pp. 901–906.
[44] P. Negandhi, Y. Trivedi, and R. Mangrulkar, "Intrusion detection system using random forest on the NSL-KDD dataset," *Emerging Research in Computing, Information, Communication and Applications*, Springer, 2019, pp. 519–531.
[45] A. Ayodeji, Y.-K. Liu, N. Chao, and L.-Q. Yang, "A new perspective towards the development of robust data-driven intrusion detection for industrial control systems," *Nuclear Engineering and Technology*, vol. 52, no. 12, pp. 2687–2698, 2020.
[46] M. R. Begli, F. Derakhshan, and H. Karimipour, "A layered intrusion detection system for critical infrastructure using machine learning," in *Proc. 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE)*, IEEE, 2019, pp. 120–124.