Knowledge-Based Systems, Volume 219, 2021, 106887. https://doi.org/10.1016/j.knosys.2021.106887

Network intrusion detection with a novel hierarchy of distances between embeddings of hash IP addresses

Manuel Lopez-Martin [a, *], Belen Carro [a], Juan Ignacio Arribas [a, b], Antonio Sanchez-Esguevillas [a]

[a] Dept. TSyCeIT, ETSIT, University of Valladolid, Paseo de Belén 15, Valladolid 47011, Spain
[b] Castilla-Leon Neuroscience Institute, University of Salamanca, Salamanca 37007, Spain

* Corresponding author: manuel.lopezm@uva.es (M. Lopez-Martin)

E-mail addresses: manuel.lopezm@uva.es (M. Lopez-Martin); belcar@tel.uva.es (B. Carro); jarribas@tel.uva.es (J.I. Arribas); antoniojavier.sanchez@uva.es (A. Sanchez-Esguevillas)

The authors declare that there is no conflict of interest regarding the publication of this paper.

Abstract— Including high-dimensional categorical predictors in a machine learning model is a major challenge. This is particularly appropriate for the IP and Port addresses of network connections when they are considered as predictors (features) in machine learning models. These features are particularly important for network intrusion detection, as many attacks exploit information about IP/Port addresses. The sparsity and high dimensionality of these features make it difficult their inclusion into the models, being discarded as useful information in many cases. This work proposes to replace the original network addresses by new features based on a set of distances defined between different components of the source and destination IP and Port addresses. These distances incorporate information on the probability of co-occurrence of source and destination addresses. The distances are calculated using a dense, low-dimensional vector representation (embedding) of the different network address components. The embeddings are obtained with a neural network, which requires few computational resources, plus an additional hash function that collapses the extremely large range of IP and Port values, making the model implementation feasible. A self-supervised learning framework under a hierarchical model is used to train the encoding network .
The novel features can be used to predict future co-occurrence of source and destination network addresses, and, when applied as features in a supervised model, they significantly increase the prediction performance of most classifiers for the detection of network intrusions. We demonstrate this prediction improvement over two modern network intrusion datasets: CICIDS2017 and CICDDoS2019.

Index Terms—hash function; self-supervised learning; neural network; network address embedding; network intrusion detection

INTRODUCTION

Network traffic analysis and prediction (NTAP) is an important field from both a research/academic and economic/practical standpoint [1]. Several data networks related problems can be considered as part of NTAP, such as: network traffic resource allocation, traffic identification and network intrusion detection (NID). Machine learning (ML) models have been applied extensively to NTAP problems, being currently one of the most active areas of research in this field [2]. The ML prediction models are mainly based on the supervised learning paradigm which requires a labeled dataset with information about the network connections (features) and expected results (labels). Extracting as much information as possible from network connections is critical to the prediction process, and among the many information items of the connection, the source and destination addresses (i.e. IP and Port addresses) are particularly important. However, these addresses correspond to categorical features with an extremely large range of values (high cardinality). Dealing with high cardinality categorical features is challenging because the usual encoding method (one-hot encoding)[3] will transform them into a large number of sparse binary features, artificially increasing the dimension of the input data and creating difficulties for their inclusion in the ML models as useful features.

This work makes a new proposal to transform the source and destination IP and Port addresses into a small number of low-dimensional vectors (embeddings) from which can be extracted meaningful information about their distance. Interpreting distance in terms of probability that the source and destination addresses belong to the same network connection. We define a series of new distances between these embeddings and demonstrate that they can be used as new features replacing the original source and destination IP and Port addresses.

There are several approaches to encode a categorical variable with N possible values. The easiest one is called one-hot-

encoding which consists in creating N dummy associated variables each of them representing, with a 1 or 0, the presence or absence of each of the N possible values [3]. The coding can also be done with N-1 dichotomous (1/0) variables if one of the values is assumed as a reference. This opens the possibility of different encodings with N-1 variables by selecting different coding values (not necessarily 0 or 1), and allowing to create different contrast coding systems that are used to encode categorical values for classification or regression problems [4]. In any of the aforementioned encodings, there is a problem for high cardinality categorical variables, since the number of resulting dummy variables is large with sparse values (most of the values are 0). There is also the possibility to encode the values as simple integer/real values, which requires much less encoding space, but creating a crucial problem since these encodings involve an implicit order and a metric between values that do not correspond to the nature of the categorical variables [5]. The best approaches to solve this problem are based on different data compression strategies associated with different types of information loss: (a) Aggregating values by frequency/percentiles [5,6]. (b) Agglomerating/collapsing values with some kind of hash function [5,7,8]. (c) Categorical encoding using target statistics, where each categorical value is replaced by some conditional probability of the expected output (classification output) given the value [9–11]. (d) Dimensionality reduction of categorical variables [12,13]. (e) Random projections [14]. (f) Compressing (embedding) the information contained in the sparse one-hot-encoded variables into dense vectors of comparatively small dimensionality, where the distance between the new dense vectors is chosen to be representative of some semantically meaningful similarity/co-occurrence of the categorical values [15,16]. (g) Combinations of any of the above methods. In this work we have opted for this latter alternative, proposing a novel combination of hash functions and embeddings to encode the network addresses into dense low-dimensional vectors.

Before applying the above mentioned coding scheme to the source and destination network addresses, and considering their hierarchical structure, we separate the IP and Port addresses into seven network address elements (NAEs): IP address, Port number, concatenated IP&Port address, and the four separated components of the IP number (8-bit blocks). These seven NAEs are the ones that will be encoded in vector representations called *embeddings.* Next, we calculate the cosine distance between each pair of embeddings for the source and destination NAEs, arriving to seven distance values for each network connection. Each value corresponding to the distance between a particular pair of source and destination NAE. These seven distances will be the new features replacing the original IP and Port addresses. The distances correspond to scalar values, with which we manage to transform four high-dimensional categorical variables (source and destination IP and Port addresses) into seven real value features. Finally, a linear combination of the seven distances followed by a sigmoid function allow us to obtain a final aggregate value, that we associate to the probability that the source and destination IP and Port belong to the same connection. This final aggregate value, interpreted as a probability, is considered as an additional feature to achieve a final list of eight features as substitutes for the original source and destination IP and Port addresses. These eight final features correspond to: seven scalar values for the distances between the seven source and destination NAEs and an additional scalar value for the associated probability that the source and destination addresses share a connection.

To make the whole process computationally feasible is necessary to reduce the range of values of the NAEs, which is huge. A hash function [7,8] is used for that purpose, with an assumed possibility of collision, which can produce that two different NAEs can obtain the same hash value. We apply a hash function only to the IP, Port and IP&Port addresses. The blocks that integrate the IP address are treated as such, without being hashed, considering that their range of values is already small (0-128). For the hashed NAEs, their hash values will be used to create their embeddings. Considering the problem of hash collisions, the creation of the previously mentioned hierarchy of seven NAEs is fundamental to create a set of distances where we can guarantee that even in the case of hash collisions, the probability of a simultaneous collision for all three NAEs: IP, Port and IP&Port, is negligible. That means that, even with hash collisions, we will have at least one differentiating distance (out of the three) that will allow us to separate any IP and Port address from the rest. We show this both analytically (Section 3.2) and empirically with two large modern intrusion detection datasets (CICIDS2017 and CICDDoS2019) (Section 4)

The proposed distance calculation is performed only between peer-to-peer NAEs, for example: source and destination IP address, source and destination Port number, etc. The distance calculation can also be extended to different types of NAE (considering that their embeddings have all the same dimensions) allowing to verify the co-occurrence of different combinations of NAE types, for example: source IP address with destination IP Port. Nevertheless, the reason for establishing distances exclusively between peer-to-peer NAEs is due to the need to reduce the number of combinatorial permutations between pairs of NAE types. In this way, comparisons are reduced, and training is simplified. The potential problem of only doing peer-to-peer comparisons is solved by making item comparisons across the entire IP address hierarchy, allowing possible interactions between IP address items to be included at successive distances. This is an additional reason to include not only the distances between the IP and port addresses, but also the distances between the IP, the Port, and each of the addressing blocks that make up the IP address (8-bit blocks).

The tuning of the encoding model that generates the NAE embeddings, calculates their distance and predicts the probability of

network addresses co-occurrence is based on the end-to-end training of a single neural network (NN). This NN is trained by minimizing the classification loss (cross-entropy) between 1) a binary flag (0/1) that indicates if the source and destination addresses belong to the same connection and, 2) the output of the NN i.e. address co-occurrence probability. By optimizing this probability, we also optimize the other distances as intermediate outcomes. Hence, to train the encoding model we need to access real source and destination address pairs that belong to the same connection and, also to generate fake network address pairs that do not belong to any real connection. The training process is framed in a self-supervised scheme with a random sampling procedure that is used to generate fake connections. This procedure for generating fake pairs is called negative-sampling. We have adapted the method proposed in [15], which was used to encode the words of a vocabulary for natural language processing (NLP), following the skip-gram approach, where the words are encoded with the aim to maximize the prediction ability to detect context words given a certain origin word in a text. In our case, we assimilate the origin and context words to each pair of source and destination NAEs.

To demonstrate the improvement obtained by including the new IP and Port transformed features in a prediction model, we have applied them to two datasets: CICIDS2017 and CICDDoS2019 [17,18]. These two datasets are two large and modern NID datasets. There are several ML algorithms that already offer excellent detection capabilities for these datasets, without incorporating any information about network addresses. This makes the improvements in the detection results obtained by including the proposed new features more interesting (Section 4). Additionally, when the new features are included, they appear in the group of most relevant features using several scoring algorithms (Section 4). The proposed features can be also used in an unsupervised framework to predict the probability that a new test pair of source and destination addresses belongs to a real connection. This can be useful to anticipate future traffic flows or to detect a change in the traffic pattern when the predicted probabilities do not match the observed ones.

The contributions of this work are the following: (1) Propose an efficient way to include the network addresses as useful features in ML models. (2) A novel approach to transform (encode) the source and destination network addresses (high dimensional categorical variables) into a reduced number of scalar values that represent the likelihood of sharing a network connection at different levels of granularity in the network address hierarchy. (3) An efficient computational model that requires few resources by using a hash function which permits to work with embedding tables of reduced size. (4) Demonstrate that using the new features provides an improvement in the prediction capabilities of ML models. 5) The nature of the encoding model, which is based entirely on a neural network, allows incremental training, which is needed in all data network applications with a constantly changing network traffic. (5) The preparation of the encoding model (model training) is very fast, as well as the subsequent features encoding.

The organization of the paper is as follows: Section 2. examines related works. Section 3 presents the details of the work. Section 4 provides an analysis of results and, Section 5 provides discussion and conclusions drawn.

2. RELATED WORKS

There are several works presenting different approaches in the literature related to the encoding of IP addresses:

a) Generate features based on traffic statistics from the different IP addresses, aggregated during a certain time interval e.g. number of distinct destination addresses, repetitions, entropy of the distribution of IP destinations: In [19] are presented several approaches for IP address encoding into number of bytes and packets and flows associated with the IP address, as well as features related with the IP addresses connected with a particular one, such as: arithmetic mean of these IP addresses (treating IP addresses as integers), the entropy of the distribution of the IP addresses, the standard deviation of this distribution, and the unique number of such IP addresses. A similar approach is taken in [20] using different ad-hoc features formed by standardized counts and standard deviation of number of hosts and ports connected with a certain host.
b) Translate the IP addresses to geographical locations, allowing to establish distances between locations as a proxy of distances between the IP addresses: This is the direction adopted in [21].
c) Extend algorithms for categorical distances to IP addresses: In [22] the authors propose a quantization of the IP addresses based on clustering with a predefined distance metric derived from the absolute difference between the IP-blocks considered as integers. Histograms are latter built based on the quantized addresses. The work in [23] creates different metric spaces for the different components of the network address defining a distance between them based on dynamic time warping. The network address components are extracted based on an ad-hoc knowledge of semantically assumed causal relationships between them.
d) Translate the IP address to an ad-hoc encoding based on IP addressing knowledge: [24] proposes an algorithm to discover communities of IP addresses (clustering) between addresses inside a managed network domain an external addresses applying a graph based algorithm. The work of [23] can also be included in this category.
e) Use word embeddings algorithms for the encoding of IP addresses: There is a single work with this approach [25]. This

work also creates embeddings for IP addresses, but with important differences with the method proposed here, such as: the structure of the neural network used, the features obtained, the training of the embeddings and the objective function adopted, and, in applying hash functions to further reduce the sizes and prediction times of the neural network used; in addition, this work uses the new features to improve visualization methods and clustering, not to enhance prediction in a supervised learning problem.

All the above referred approaches (with the exception of the last one which is similar to the current work) are based on the generation of ad-hoc features representing the IP and Port addresses, with specific assumptions about their properties, and their relationship and correlation with other features of the network connection. Instead, the method proposed in this work is based on feature (representation) learning from data, based on an self-supervised model that generates the transformed features as an intermediate output while trying to optimize the prediction of the probability of co-occurrence of source and destination addresses (considering all the different elements that compose a network address and their hierarchy).

3. WORK DESCRIPTION

This section introduces (1) the datasets used for all experiments carried out for this work, (2) the rational for the strategy adopted to split the network address, and (3) the encoding model used to obtain the embeddings of the network addresses and the computation of their distances.

3.1 Datasets

We have used the CICIDS2017 and CICDDoS2019 datasets [17,18], which are two modern intrusion detection datasets. They include network flow data and associated labeled intrusions, with clearly identified source and destination addresses for each network connection. Each dataset provides a different challenge for an intrusion detection classifier, each of them has a different network traffic structure and was constructed using two very different network configurations, in addition, the objectives of both datasets are also different: one is intended for the detection of generic intrusions (CICIDS2017) and the other is specifically dedicated to DDoS attacks of different types (CICDDoS2019).

3.1.1 CICIDS2017

The CICIDS2017 dataset presented in [17] corresponds to labeled network flows with identified intrusions and normal traffic. It is a network intrusion detection dataset with a large number of recorded flows and an up-to-date list of intrusions. The traffic consists of several days of recording from which we have used a single day (Friday, July 7, 2017) with 702209 connections and 396330 round trip connection flows (flows sharing the same source and destination addresses in any direction).

CICIDS2017 provides two different detection scenarios: four intrusion labels (Normal, Bot, DDoS and PortScan), or, two intrusion labels (Normal, Attack), where the Normal label is associated to normal traffic and the Attack label to the presence of any attack. The frequency distribution of attacks for the two scenarios are: 2 labels – Normal(58.9%) and Attack(41.1%); and 4 labels – Normal (58.9%), Bot (0.3%), DDoS (18.2%) and PortScan (22.6%).

In addition to the intrusion label, CICIDS2017 provides 85 features (predictor variables), of which, 7 are categorical variables and 78 are continuous. The categorical features are source IP, source Port, destination IP, destination Port, protocol (3 values), time stamp and flow id (forms by the concatenation of source and destination IP&Port addresses). The continuous features provide descriptive statistics on the size, number, rate, duration and inter-arrival-times of the header and payload of the flow's packets. After processing the dataset, eliminating the variables with a constant zero value or very low variance, we arrive at a final set of 70 continuous and 7 categorical features. In addition, the continuous features, which have a wide range of values, have been scaled with a log (1+x) transformation followed by a min-max scaling to a range of values between [0-1]. The scaling of the continuous features is required by several ML models (e.g. neural networks)

Train/test sets used to train the NID models

CICIDS2017 has been randomly divided into two separated training and test sets, containing 562539 and 139670 connections respectively (20% split for testing), these two sets are used to train the ML models presented in Section 4 with the test set used to obtain the intrusion detection metrics. In order to perform the split between training and test sets we have ensured that all connection belonging to the same flow enter completely into one of the training or test sets. The split has been done respecting the distribution of the labels in the original dataset (stratified split).

Train/test sets used to train the Encoding model

An additional dataset is prepared to train the Encoding model (Section 3.3), which requires a specific train/test sets. This

additional dataset is related but different from the dataset used to train the ML models for intrusion detection. This dataset will be used to train the Encoding model that will predict whether a pair of source and destination network addresses are part of the same connection i.e. predict the probability that they belong to a real or fake network connection. To perform these predictions, we use exclusively the source and destination IP and Port addresses, and none of the other features of CICIDS2017. To build the new dataset, we extract the source and destination IP and Port addresses from the original CICIDS2017. This provides initial training and test sets that correspond to actual source and destination network addresses, that is, source and destination addresses that belong to actual connections. To train the Encoding model (Section 3.3), we also need fake connections with randomly selected source and destination addresses in the set of network addresses of the dataset (negative-sampling) [26]. To build the fake connections, we fix the source or destination address of a real connection and randomly change the other address. Following this procedure, we arrive at a final training set of 2250156 source and destination addresses, where 50% of the samples correspond to real connections and the other 50% correspond to fake connections. Similarly, the test set has 558680 samples with the same composition of real and fake connections as the training set. The labels for this dataset consist of binary labels (0/1) associated with real and fake connections, respectively.

3.1.2 CICDDoS2019

The dataset CICDDoS2019 [18] is intended to test ML models in the classification of distributed denial of services (DDoS) attacks. The objective of the dataset is to provide a complete and updated collection of logged traffic associated with different types of DDoS attacks. The network configuration that generates the dataset, the intrusions recorded, and the traffic patterns are completely different to the CICIDS2017 dataset. It is also a labelled dataset with several instances associated to different days of recording from which we have used one day (March 11th, 2019) for the UDP type of attacks with 674463 connections and 672320 round trip connection flows (flows sharing the same source and destination addresses in any direction). The mode of the number of connections per round trip connection flow is 1-2, with a few having as much as 15 connections.

The subset of CICDDoS2019 chosen for this work provides two different detection scenarios: four intrusion labels (Normal, Syn, UDP and UDPLag), or, two intrusion labels (Normal, Attack), where the Normal label is associated to normal traffic and the Attack label to the presence of any attack. The frequency distribution of attacks for the two scenarios are: 2 labels – Normal(0.6%) and Attack(99.4%); and 4 labels – Normal (0.6%), Syn (82.5%), UDP (16.6%) and UDPLag (0.3%). It is a much more unbalanced dataset than CICIDS2017.

The number of continuous and categorical features is similar to CICIDS2017, but after processing the dataset eliminating the variables with a constant zero value or very low variance, we arrive at a final set of 67 continuous and 7 categorical features. The scaling of the continuous features is also similar to CICIDS2017 with a log (1+x) transformation followed by a min-max scaling to a range of values between [0-1].

Train/test sets used to train the NID models

The CICDDoS2019 dataset has been randomly divided into two separated training and test sets, containing 539579 and 134884 connections respectively (20% split for testing), these two sets are used to train the ML models presented in Section 4.3 with the test set used to obtain the intrusion detection metrics. The train/test split procedure is similar to the one used for the CICIDS2017 dataset (Section 3.1.1).

Train/test sets used to train the Encoding model

To train the Encoding model (Section 3.3) with the CICDDoS2019 dataset, we have prepared an additional dataset created from the source and destination IP addresses of CICDDoS2019. The process to build this additional dataset has been identical to the process used for CICIDS2017 (Section 3.1.1). The source and destination IP addresses of the CICDDoS2019 samples are used to acquire real connections, and a negative sampling process as presented in 3.1.1 is used to create the fake connections. Finally reaching a dataset of 2158316 and 539536 connections for the training and test sets respectively, with a similar distribution between fake and true connections.

3.2 Network address split

As presented in Section 1 (Introduction), we have split the IP and Port addresses into seven network address elements (NAEs): (a) IP address (IP), (b) Port number (Port), (c) a concatenation of the IP and Port numbers (IP&Port) and, (d) the four separated components of the IP number (8-bit blocks) (IP_1, IP_2, IP_3, IP_4). These elements follow a hierarchical structure which can be observed in Fig. 2. This initial split of the IP and Port addresses into separate components, at different level of granularity, is an original proposal of this work and is fundamental to the proposed work. The main reasons for this approach are:

a) The aim of the work is to replace the high-cardinality source and destination IP and Port addresses by a few scalar

features that incorporate information about the co-occurrence of the source and destination addresses. Our hypothesis is that this information related to address co-occurrence is the most important for a prediction model. Considering the huge range of values for the IP and Port addresses (2^{32}, 2^{16} and 2^{48} for the IP, Port and IP&Port respectively), and the even bigger number for all possible combinations of source and destination addresses, it is clear that the intended feature transformation is a challenge. The adopted solution has been to reduce the range of values to a feasible range using a hash function. That means that the output range of values of the hash function must be relatively small, which increases importantly the chances of collisions even when considering that the real number of distinct IP and Port values on a network session will be much smaller that the theoretical range. For example, for the CICIDS2017 dataset the number of distinct (source and destination) IP values is 8330 which is approximately three times the proposed maximum output value for the hash function, which is 3000 (Section 4). That means a guaranteed probability of collision and that using a hash function for the IP, Port and IP&Port will provide noisy (with collision) values for each of these NAEs, considered separately. Nevertheless, this probability of collision get extremely reduced when considering a total probability of collision as the probability that all three NAEs (IP, Port and IP&Port) will have a simultaneous collision.

An upper bound expression for the total probability of collision for the three NAEs (IP, Port and IP&Port) is provided in Eq. 1. Considering that the hash function will provide a uniform distribution of output values for any input, and that N_{IP}, N_{Port} , $N_{IP\&Port}$ are the number of distinct values for the IP, Port and IP&Port NAEs, and k is the maximum output value of the hash function. The probability of a single total collision is the product of the probabilities that the second source and destinations pair will land on the same hash bucket as the first pair. This probability depends on the hash size (k) following the expression $(1/k)^2$. This product must be extended to all three NAEs, that is the reason of the additional power 3 in Eq. 1, forming the expression $[(1/k)^2]^3$. This probability still needs to be multiplied by all possible combinations of source and destination addresses for each NAE (first term in Eq. 1) to account for all the possible ways that the second pair of addresses can be constructed. The inclusion of this last term allows creating a hard upper bound by considering all possible address combinations in the dataset.

We have performed Monte Carlo simulations to empirically evaluate the collision probability [27]. The simulation results corroborate that the probability that two source and destination network address elements (NAE) have a simultaneous collision in both sides (source and destination) follows the expression $(1/k)^2$. Considering the three NAEs simultaneously the probability is $(1/k)^6$. In all simulations performed with a k equal to 3000, not a single simultaneous collision has appeared for the three NAEs with a number of samples in the simulation of 1e+7. Even considering only two NAEs (IP and Port), not a single simultaneous collision is obtained.

Considering the hashing of the IP&Port address and its dependency on their constituent IP and Port values, it is important to note that the hash function acts differently for the Port, IP and IP&Port addresses. The Port is hashed using its integer number and for the IP and IP&Port we use their text representation, leaving the resulting hash value for the IP&Port not directly related to their constituent IP and Port values.

We show in Fig. 1 that the total probability of collision is negligible for the parameter values considered in this work: maximum hash value of 3000, and maximum number of different IP, Port and IP&Port addresses of 218406, considering that the CICIDS2017 dataset has 8330, 62434 and 218406 different values for the (source and destination) IP, Port and IP&Port NAEs, respectively. The probability graphs presented in Fig. 1 are obtained by applying Eq. 1.

$$\text{Probability of collision} = (N_{IP}{}^2 + N_{Port}{}^2 + N_{IP\&Port}{}^2) \cdot \left[\left(\frac{1}{k}\right)^2\right]^3 = (N_{IP}{}^2 + N_{Port}{}^2 + N_{IP\&Port}{}^2) \cdot \left(\frac{1}{k}\right)^6 <$$
$$< 3N_{max}{}^2 \cdot \left(\frac{1}{k}\right)^6 = 3N_{max}{}^2/k^6 \quad (1)$$

Assuming $N_{max} = max\,(N_{IP}, N_{Port}, N_{IP\&Port})$, and each N_{IP}, N_{Port}, $N_{IP\&Port} \gg 0$. Where N_{IP}, N_{Port} , $N_{IP\&Port}$ are the number of distinct values for the IP, Port and IP&Port NAEs, and k is the maximum output value of the hash function.

The approach adopted here has some similarities to bloom filters [28] that implement a set-membership detector based on a series of hash functions that are applied separately to all elements in the set; allowing detection, with complete certainty, that there is no similar value in the set if there is no collision with any previously produced hash outputs.

b) The IP-blocks (8-bits blocks) have a range of values of [0.128] which is very small and can be used directly to obtain a sensible word embedding. These NAEs are not hashed, which makes it even more difficult to have a total collision considering the seven NAEs. Taking all these into account, when all seven NAEs are considered simultaneously and their distances are obtained, we can be sure that the seven distances plus the additional probability of source and destination address co-occurrence, all of them provided by the Encoding model (Section 3.3), form a series of unique non-clashing features for any combination of source and destination network addresses.

c) As already mentioned in Section 1, having the distances between different address components at different level of granularity can be considered as a substitute for the need to explore distances between different types of NAE, which could considerably increase the complexity of the Encoding model.

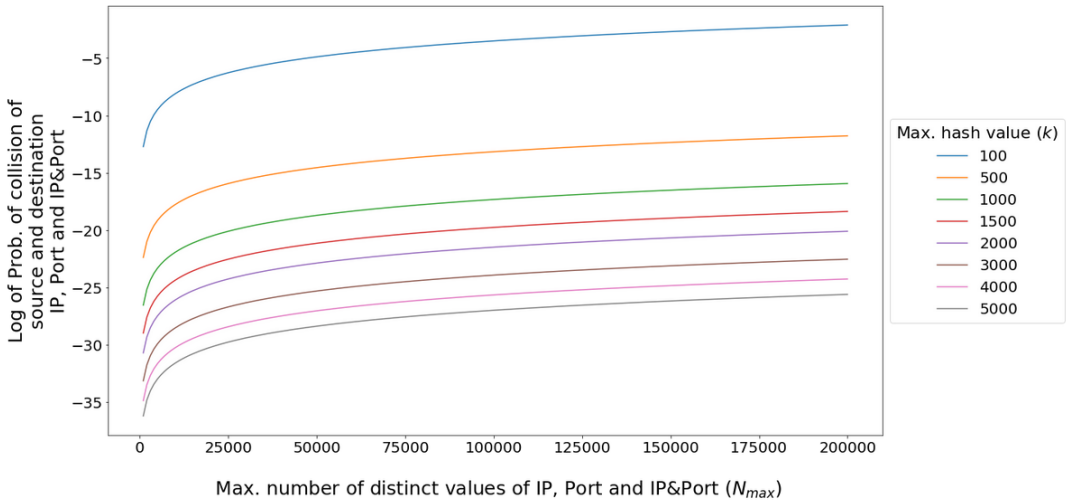

Fig 1. Chart showing collision probability (y-axis) vs. maximum number of different IP, Port and IP&Port addresses (x-axis), and for different scenarios based on the maximum output value of the hash function (chart legend). The y-axis provides the logarithm of the probability of collision.

3.3 Encoding model description

The aim of this work is to transform the source and destination network addresses (IP and Port) of a network connection, which are high cardinality categorical features, into a small set of scalar features that incorporate information on the simultaneous occurrence (co-occurrence) of source and destination NAEs on a network connection (i.e. their 'distance').

To perform this transformation we have adapted two initial tools/ideas: hash functions [8], to reduce the initial range of NAE values to an smaller range of integer values, and, the skip-gram approach proposed in [15], which originally embedded words of a vocabulary into semantically meaningful vectors whose distance is associated to the likelihood of proximity of the words in a corpus of texts. We will not present here in detail these tools/ideas as there are available for that purpose a large number of excellent references. We will focus on detailing how we have adapted and integrated these ideas into the proposed new model.

Fig. 5 shows a high-level graphic representation of the proposed model to perform the feature transformation. It serves as a summary of the process followed to create the features:

a) The seven source and destination NAEs, which are derived from the source and destination network addresses (Section 3.2), are used to construct a distance metric between them. The distance used is the cosine distance between the vector embeddings of each pair of source and destination NAEs.
b) Vector embeddings are created using a simple network architecture that translates the one-hot-encoded representation of the integer value associated to each NAE into a dense small-dimensional vector. Each embedding layer operates as an embedding table. Each type of NAE will have a different embedding table.
c) For the IP_1, IP_2, IP_3 *and* IP_4 NAEs which have a small range of values ([0,128]), their own integer value will be used as input to their embedding tables. For IP, Port, and IP&Port NAEs (with a huge range of values), their values are pre-processed separately with a hash function that collapses the input values into a much smaller set of integer values. The resulting hash values will be used as inputs to the IP, Port, and IP&Port embedding tables.
d) A linear combination of the seven cosine distances between the source and destination NAE pairs, followed by a

sigmoid function, will provide a final scalar value (output of the encoding model) that is interpreted as the total probability of co-occurrence of the source and destination network addresses. This probability is also included as an output feature in the set of transformed features.

e) Model training is done end-to-end using gradient descent with a cross-entropy loss between the model outputs and the binary ground-truth labels, which indicate whether the source and destination network addresses correspond to a real or fake connection (Section 3.1.1 and 3.1.2).

In Fig. 5 we use two IP and Port addresses as an example: source IP&Port address (192.168.10.12-443) and destination IP&Port address (128.199.12.23-123). The IP, Port and IP&Port addresses have a theoretical range of values of 2^{32}, 2^{16} and 2^{48} which are unfeasible to work with. We reduce this range of values using a hash function that provides a mapping from the large set of initial values to a reduced set of integer values. We consider different inputs to the hash function: for the Port value we use its integer number and for the IP and IP&Port values we use their text representation. In either case, the hash function transforms each distinct initial value to a hopefully distinct integer value. The possibility that two distinct initial values get transformed into the same integer value is called a collision (Fig. 4). When the NAE hashes are considered separately, there is a certain possibility of collision, but this possibility becomes negligible when considering all three NAE hashes simultaneously (Section 3.2) [7,28].

The hash function implementation is provided by the R language plus an additional modulus division to adapt to the desired range of integer outputs. The algorithm used is xxHash [29] which is a non-cryptographic hash function that prioritizes speed, in addition to passing the SMHasher test that evaluates the collision, dispersion and randomness of hash functions [30]. We have experimented with the use of 1000, 3000 and 5000 for the output range of integers of the hash function, using finally 3000 as the selected value (Section 4).

The four blocks that compose an IP address (8-bit blocks), which have a small range of integer values ([0,256]), are not hashed. The four IP blocks are represented as IP1, IP2, IP3 and IP4. For simplicity purposes, in Fig. 5 we show only the first and last IP blocks (IP1 and IP4). We have not included the type of protocol in the coding since its range of values is quite small and can be incorporated by itself into a non-problematic feature (e.g. one-hot-encoded).

After the hash function is applied to the IP, Port and IP&Port values, we arrive to an integer representation for all NAEs, including the IP blocks. These integer values have a different range of values depending on the type of NAE. The range of values for each IP block is [0-255], and [0-2999] for the rest of the NAEs. These integer values are the input to the neural network (NN) that composes the proposed Encoding model. The NN has an initial set of embedding layers (one different per type of NAE) that transform the input integer values into a one-hot-encoding representation and provide a simple connection between this one-hot-encoded input and an output layer of dimension 3 (Fig. 3). Other dimensions of the embedding output layer have been investigated (Section 4) but finally 3 has been chosen (Fig. 3).

The weights of the embedding layer (Fig. 3) form the values of the embedding tables (one per type of NAE). These embedding tables, once the NN is trained, can be used separately to obtain the embedding values for each NAE (in case that is needed). In our case, unlike other approaches, we do not focus on these embedding values as our output, but on the cosine distance between each pair of source and destination embeddings for the different defined NAEs.

The cosine distance is a normalized dot product between each embeddings pair. The cosine distance allows to implement a similarity computation between the different source and destination NAEs. These distances are the input to a final layer that implements a linear combination of their input values followed by a sigmoid function, which gives an output value ($\hat{Y}$) (Fig. 5) that can be interpreted as the probability that the source and destination network addresses are part of a common network connection. This output value together with the cosine distances between the 7 NAEs integrate the list of 8 features that we use in our ML models for the experiments presented in Section 4. These list of 8 features are the real output of our model and represent the transformation into useful features of the initial source and destination network addresses. The cosine distance was chosen after testing other alternative distances, e.g. scalar (dot) product, with worse results.

The computation of the cosine distance and the sigmoid function are the only non-linearities of the model, which makes it extremely simple, with short training and prediction times and low memory requirements given the small sizes of the different embedding tables (one per NAE). The small size of these tables can be attributed to having a very small range of input values thanks to the hash function, and their output values correspond to very short-dimensional (three-dimensional) vector embeddings.

The NN presented in Fig. 5 is trained end-to-end with the training sets described in Section 3.1.1 and 3.1.2 to train the Encoding model. In this case, the original training set corresponds to real connections with real source and destination IP and

Port addresses. In order to train the model, we need also false (fake) connections where the source or destination addresses have been changed randomly by other addresses not connected with the original one (negative-sampling) (Section 3.1). The expected ground-truth output (Y) of the NN will be a binary value {0,1} with a 0 for the false connections (results of negative-sampling) and a 1 for the real ones. The NN will be trained trying to reduce the following loss function (logloss/binary-cross-entropy):

$$Loss = LogLoss(Y, \hat{Y}) = -\frac{1}{N}\sum_{i=0}^{N} Y_i \log(\hat{Y}_i) + (1 - Y_i)\log(1 - \hat{Y}_i) \quad (2)$$

where N is the number of connections (real and false) used in our training set.

To train the model we have used a batch size of 100, with 100 epochs and an early-stopping with a patience of 10 (maximum number of epochs without improvement in results), using a validation set of 20% of the training set. The optimizer used has been Adam [31] with default parameters. The total number of trainable weights has been 29056.

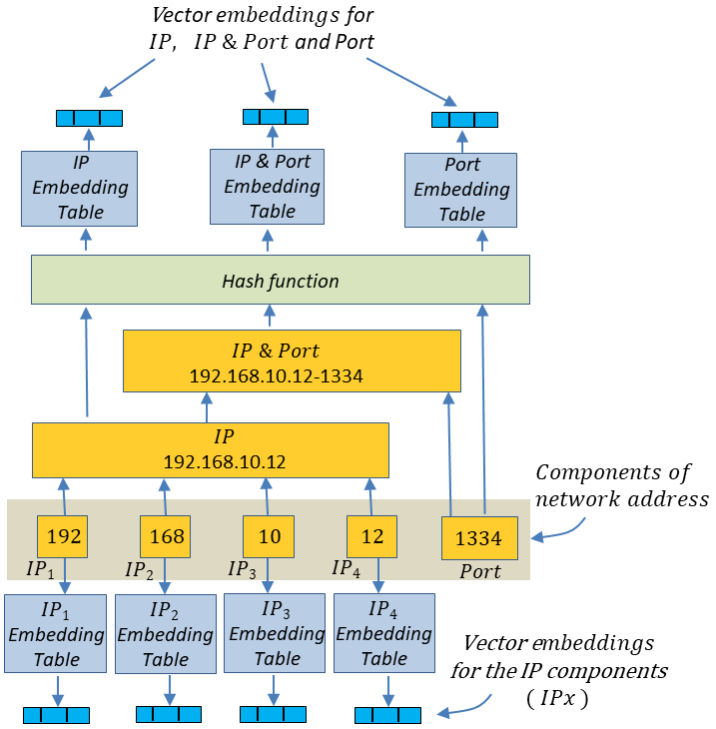

Fig 2. Diagram showing the hierarchy of network address entities for IP and Port addresses. Some entities require a hash function to reduce their range of values. All of them are transformed to a low dimensional dense vector representation (embedding) using different embedding tables. The embedding tables are built by training the neural network shown in Fig. 5

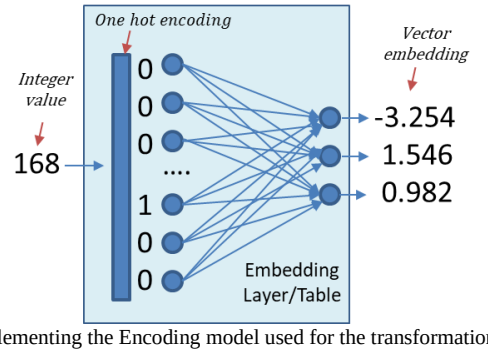

Fig 3. Detail of the embedding layer of the NN implementing the Encoding model used for the transformation of features. This layer provides a mapping of an integer value to a one-hot-encoded representation which is directly connected to an output vector. A linear activation is used for the layer. The weights of the layer form the embedding table.

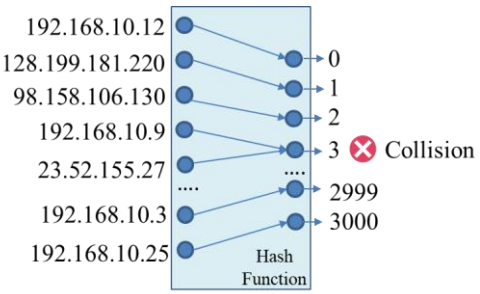

Fig 4. Detail of the operation of the hash function. An input string is mapped to a limited range of integer values. The range of input values is greater than the range of output values, causing possible collisions (two or more inputs are assigned to the same output). A well-designed hash function will provide a uniform distribution of output values for any input.

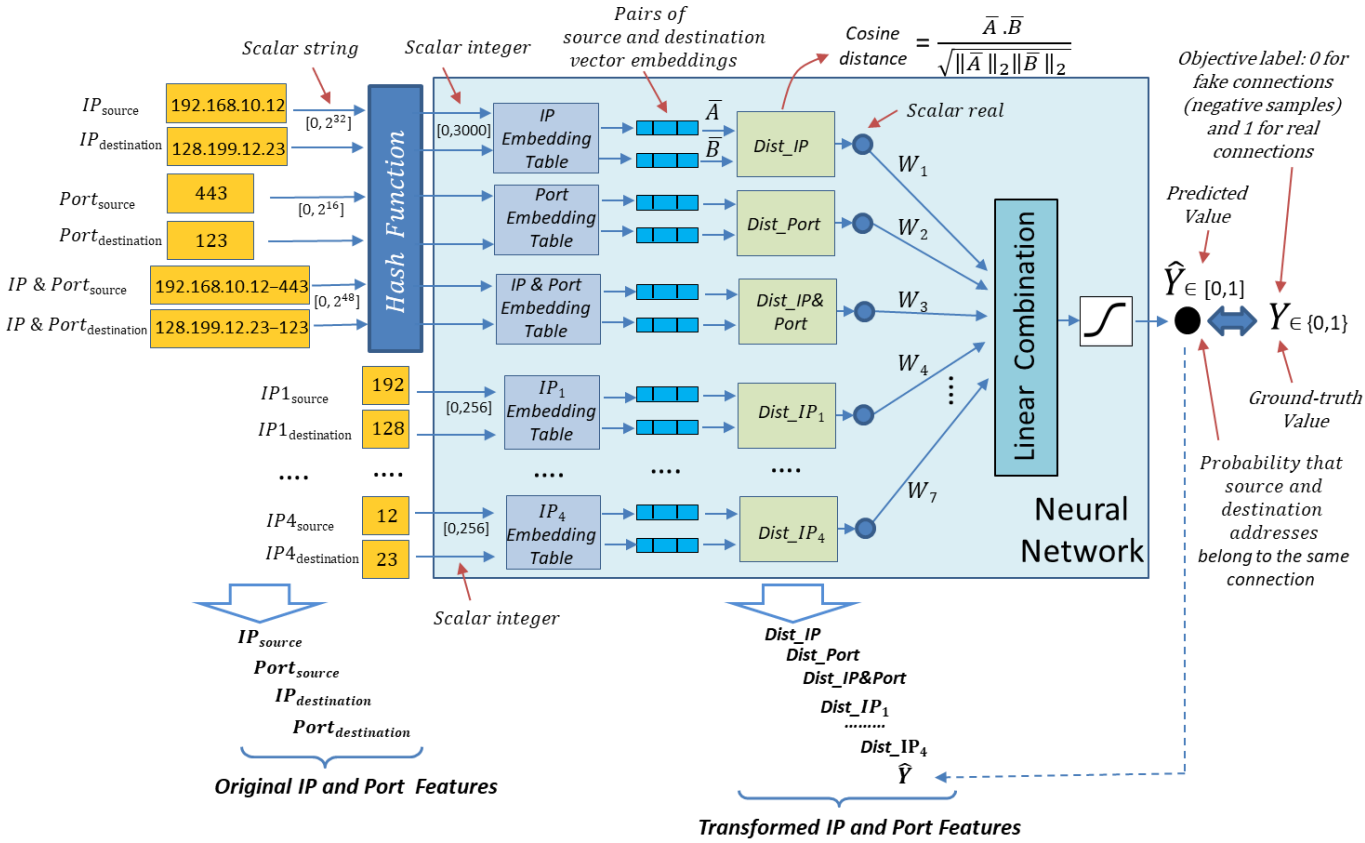

Fig 5. Diagram depicting the architecture of the NN used to derive the new IP and Port transformed features and their associated embedding tables. The output of the NN is the probability of co-occurrence of the source and destination IP and Port addresses. This output is also part of the transformed features. The area in light-blue covers the entire neural network. The configuration of the embedding tables is described in Fig.3.

4. RESULTS

This section presents several results obtained by using the new features generated by our proposed model (Section 3.3). These results attempt to provide empirical evidence on the properties of the new features, focusing in two directions:

- When the new features replace the original source and destination network addresses of the NID datasets (CICIDS2017 and CICDDoS2019), they provide an improvement in the prediction metrics of several classic ML models: Adaboost, Linear Support Vector Machine (SVM), Logistic Regression, Random Forest, Gradient Boosting Machine (GBM) and Multilayer Perceptron (MLP). We present the prediction results for NID with the two detection scenarios in Sections 3.1.1 and 3.1.2 (2 labels and 4 labels). We will show that the improvement is significative for both scenarios. These results are particularly challenging for these datasets, as the detection results are already very high using exclusively the original numeric features.
- The new features can be used independently to predict whether a pair of new test source and destination addresses will be part of the same connection. To demonstrate these results, we will use the test sets described in Section 3.1.1 and 3.1.2 as part of the datasets used to train the Encoding model. This test sets consist of pairs of real source and destination

addresses, as well as fake pairs obtained by negative-sampling (Section 3.1). It has been particularly important to ensure that none of the test pairs have been used during training.

Section 4.1 presents in detail the results obtained with the CICIDS2017 dataset, which is a more generic dataset than CICDDoS2019 and, therefore, it seems more appropriate to extend the experimental details on CICIDS2017. The results obtained with CICDDoS2019 are presented in Section 4.2. The results presented for CICDDoS2019 are more condensed and focus on showing that the best model for CICIDS2017 is also adequate to improve the results obtained with CICDDoS2019, which provides evidence on the generalization of the results for different datasets.

We implemented the neural network model in python using Tensorflow/Keras [32]. For all other models we used the scikit-learn python package [33]. To perform the preparation of the dataset, we used the R environment.

4.1 Results for the CICIDS2017 dataset

All the results presented in this section have been produced with the dataset CICIDS2017 (Section 3.1.1). This section presents the results obtained by using the new distance features to improve the prediction results of several classic ML models (Section 4.1.1), and classify a pair of source and destination network addresses as real or fake (section 4.1.2).

4.1.1 NID prediction

We will apply the new features generated by our model to the two detection scenarios described in Section 3.1.1: a) detection of Normal and Attack traffic and b) detection of Normal, Bot, DDoS and Portmap attacks. The first scenario poses a binary classification problem with a balanced dataset while the second scenario a highly unbalanced multi-class detection problem. The results obtained for the first scenario are presented in Fig. 7, and the second in Fig. 6.

In both Fig. 6 and 7, the columns of the tables represent the different set of features used to make the prediction of the labels. The first column named: “No IP and Port features” corresponds to using only the numeric features of CICIDS2017 i.e. the IP and Port information is not used. The second column named: “IP and Port frequency features” corresponds to replacing the source and destination IP and Port values with their relative frequency in the dataset. This is a common encoding method for the network address. The successive columns named: “New transformed features” correspond to different configurations that include the IP and Port information using different sets of distances between NAEs. The different configurations are represented by the following string composed of a set of NAEs (inside parenthesis) and two numbers separated by commas. The set of NAEs, inside parenthesis, are the NAEs which are used to obtain the distances. As an example: (IP, Port, IP&Port) means to use only three distances between these three NAEs; and (IP, Port, IP&Port, IP1,IP2,IP3,IP4) uses seven distances between all possible NAEs. For reasons of space we have not included the feature: “Prob. of co-occurrence”, in the list of features, nevertheless this additional feature is indeed included in all configurations for the new transformed features (from the third column to the right).The number following the parenthesis is the maximum index value used in the hash function and, the next number indicates the dimension of the embedding vectors. These three configuration parameters correspond to hyper-parameters of the model in Section 3.3.

The classification metrics used have been accuracy, F1-score, precision and recall. To define these metrics, we have considered that the presence of an intrusion is a ‘positive’ result and the opposite state a ‘negative’ result; with these definitions, a true positive (TP) is a positive detection corresponding to a real intrusion, a true negative (TN) is a negative detection corresponding to a normal connection, a false positive (FP) is a positive detection corresponding to a normal connection, and a false negative (FN) is a negative detection corresponding to a real intrusion. Then, the metrics can be defined as:

$$Accuracy = \frac{\#TP + \#TN}{\#TP + \#TN + \#FP + \#FN} \quad (3)$$

$$Precision = \frac{\#TP}{\#TP + \#FP} \quad (4)$$

$$Recall = \frac{\#TP}{\#TP + \#FN} \quad (5)$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (6)$$

where a ‘#’ sign before a symbol indicates the total number of that symbol e.g. $\#TP$ is the total number of true positives in our

prediction results.

The F1-score is more suitable for an unbalanced dataset, since accuracy in that case is a misleading metric. For the multi-class scenario (Fig. 6) it is necessary to aggregate the results for the different classes (label values), having different aggregate options: macro, micro, samples and weighted; opting for the macro average as defined in [33].

The results in Fig. 6 and 7 are coded based on a color palette between green and red, with a greener color that indicates a better result and a redder color for a worse result. The color code is applied along the rows of the tables, which allows to appreciate where the highest concentration of best results is found, which in this case occurs for the feature set option labeled with the letter L, which corresponds to using the seven distances generated by the Encoding model plus the output probability generated by the model, with a maximum hash index of 3000 and a 3-dimensional embedding vector.

The lower part of the tables of Fig. 6 and 7, provide some descriptive statistics (median, mean, standard deviation and mean absolute deviation - MAD) for all the F1-scores of the different ML models used. The statistics have been applied column-wise. The improvement rates of each option with respect to the first option ("No IP and port characteristics") are also provided in the last rows of the tables, considering separately the improvement rates for the mean and median values.

Below the tables in Fig. 6 and 7 is a box-plot chart with a box for each feature set presented at the tables. Each box gives the minimum, maximum, median and interquartile interval for all results of each of the columns in the tables (in this case, we have included all results and not just the results for the F1-score). The best results correspond to the box-plot with the highest median and the smallest interquartile interval. The box showing the best results corresponds to the best features set option (option L) (the rightmost column).

Looking at the results in Fig. 6 and 7, we can conclude that the selection of hyper-parameters that provide the best NID prediction results are: 1) incorporating all the distance features obtained with all the NAEs, 2) a maximum hash index of 3000 and 3) an embedding dimension of 3. This feature set option is labeled with the letter L in Fig. 6 and 7. It is interesting that employing higher values for the maximum hash index or the embedding dimension does not imply better results. One reason for this may be that an increase in both parameters implies a greater number of weights to train and the possibility of incurring in over-fitting. Also, it is a demonstration that, with a fairly small value for both the maximum hash index and the embedding dimension, we achieved our intended goals according to the results in Section 3.2. It is also interesting that, for the best option, we not only have better results for the mean and median values, but that the standard deviation and the MAD are also smaller, which implies better and more robust results when using the proposed additional features. For the 4 labels scenario (Fig. 6), the option with the best set of features provide an average improvement of results over 16% for the F1-score versus the results obtained with no IP and Port features (bottom of Fig. 6), and over 13% versus the results for using the IP and Port frequency features. For the 2 labels scenario (Fig. 7), the best option gives an average improvement of over 8% versus the results obtained with no IP and Port features (bottom of Fig. 7), and over 25% versus the results for using the IP and Port frequency features. In this case, the IP and Port frequency labels reduce the prediction metrics compared to not including any IP and Port features. This is never the case for the new transformed features. It is also interesting for the generalization of results, that option L is the best features set option for both the 2 and 4 labels scenarios.

Prediction Metrics (4 labels)			Feature Sets											
			A No IP and Port features	B IP and Port frequency features	C New transformed features (IP,IP+Port),1000,3	D New transformed features (IP,Port,IP+Port),1000,3	E New transformed features (IP,Port,IP+Port),3000,3	F New transformed features (IP,Port,IP+Port),5000,3	G New transformed features (IP,Port,IP+Port),3000,5	H New transformed features (IP,Port,IP+Port),3000,2	I New transformed features (Port,IP1,IP2,IP3,IP4),3000,3	J New transformed features (IP1,IP2,IP3,IP4),3000,3	K New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4),5000,5	L New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4),3000,3
	Adaboost	Accuracy	0.6752	0.7462	0.9965	0.9965	0.9066	0.9965	0.9067	0.9866	0.9209	0.9209	0.7832	0.7357
		F1	0.4275	0.4421	0.7484	0.7484	0.7268	0.7484	0.7268	0.7414	0.6880	0.6880	0.6952	0.6777
		Precision	0.6346	0.6530	0.7477	0.7477	0.7467	0.7477	0.7467	0.7377	0.6713	0.6713	0.7490	0.7480
		Recall	0.3953	0.4863	0.7491	0.7491	0.7195	0.7491	0.7195	0.7454	0.7164	0.7164	0.7393	0.6385
	Linear SVM	Accuracy	0.9890	0.8172	0.9890	0.9847	0.9821	0.9937	0.9927	0.9871	0.9975	0.9977	0.9838	0.9975
		F1	0.7419	0.7839	0.7420	0.7379	0.7355	0.7462	0.7453	0.7402	0.8720	0.8759	0.7369	0.8720
		Precision	0.7452	0.8927	0.7452	0.7434	0.7424	0.7471	0.7467	0.7445	0.9989	0.9990	0.7433	0.9989
		Recall	0.7389	0.7469	0.7390	0.7329	0.7294	0.7454	0.7440	0.7363	0.8306	0.8342	0.7314	0.8306
	Logistic Regression	Accuracy	0.9965	0.9970	0.9971	0.9971	0.9966	0.9973	0.9971	0.9966	0.9982	0.9982	0.9972	0.9982
		F1	0.7487	0.9280	0.8351	0.8377	0.7704	0.8724	0.8433	0.7488	0.8914	0.8977	0.7672	0.8918
		Precision	0.7481	0.9929	0.9984	0.9955	0.9720	0.9707	0.9875	0.7483	0.9880	0.9782	0.9675	0.9866
		Recall	0.7493	0.8907	0.8015	0.8034	0.7605	0.8349	0.8081	0.7493	0.8507	0.8594	0.7592	0.8514
	Random Forest	Accuracy	0.9975	0.9993	0.9981	0.9979	0.9979	0.9973	0.9977	0.9981	0.9984	0.9983	0.9983	0.9990
		F1	0.8857	0.9653	0.8810	0.8681	0.8628	0.7976	0.8342	0.8725	0.9071	0.9008	0.8965	0.9531
		Precision	0.9705	0.9922	0.9869	0.9833	0.9701	0.9987	0.9896	0.9876	0.9823	0.9754	0.9873	0.9773
		Recall	0.8479	0.9438	0.8407	0.8293	0.8260	0.7765	0.8013	0.8328	0.8688	0.8635	0.8562	0.9337
	GBM	Accuracy	0.9978	0.6723	0.9987	0.9976	0.9986	0.9986	0.9984	0.9978	0.9992	0.9992	0.9991	0.9978
		F1	0.8798	0.5986	0.9335	0.8885	0.9481	0.9230	0.9091	0.8953	0.9653	0.9642	0.9592	0.9636
		Precision	0.9900	0.6745	0.9712	0.8885	0.9228	0.9855	0.9911	0.9028	0.9979	0.9962	0.9757	0.9959
		Recall	0.8390	0.6409	0.9069	0.8884	0.9821	0.8869	0.8688	0.8886	0.9403	0.9397	0.9450	0.9392
	MLP	Accuracy	0.9980	0.9976	0.9981	0.9981	0.9980	0.9968	0.9978	0.9982	0.9980	0.9981	0.9986	0.9988
		F1	0.8857	0.9640	0.8784	0.8819	0.8827	0.7490	0.8659	0.8858	0.9642	0.9555	0.9293	0.9649
		Precision	0.9955	0.9969	0.9991	0.9973	0.9991	0.7485	0.9968	0.9907	0.9970	0.9261	0.9447	0.9982
		Recall	0.8440	0.9391	0.8368	0.8401	0.8406	0.7494	0.8258	0.8448	0.9393	0.9965	0.9162	0.9396

Median (F1)	0.8142	0.8559	0.8568	0.8529	0.8166	0.7733	0.8387	0.8107	0.8992	0.8993	0.8319	0.9225
MAD (F1)	0.0715	0.1087	0.0505	0.0322	0.0736	0.0259	0.0487	0.0699	0.0461	0.0398	0.0962	0.0418
Mean (F1)	0.7616	0.7803	0.8364	0.8271	0.8210	0.8061	0.8208	0.8140	0.8813	0.8804	0.8307	0.8872
Std. Desv. (F1)	0.1772	0.2178	0.0773	0.0674	0.0900	0.0753	0.0708	0.0777	0.1021	0.1004	0.1111	0.1099
Median (% Increase v.s. No IP&Port features)	0.00%	5.12%	5.23%	4.75%	0.29%	-5.03%	3.01%	-0.44%	10.44%	10.44%	2.17%	13.29%
Mean (% Increase v.s. No IP&Port features)	0.00%	2.46%	9.83%	8.60%	7.81%	5.85%	7.77%	6.88%	15.72%	15.60%	9.08%	16.49%

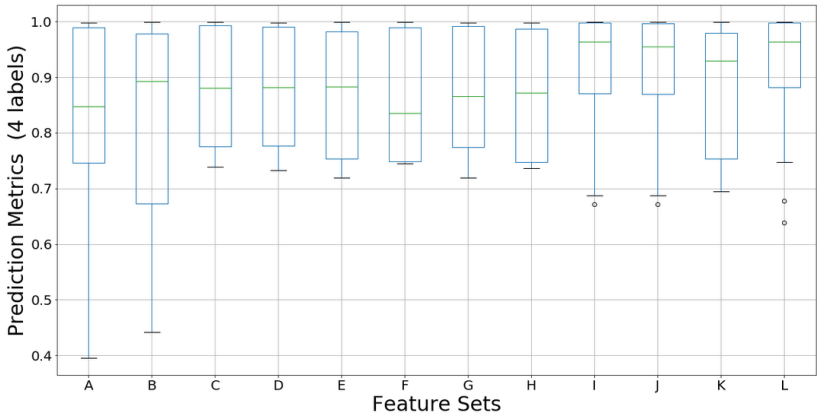

Fig 6. (Upper part) Prediction results table for different feature sets options (4 labels detection). (Lower part) Box-plots for the prediction results in the above table. We can observe the improvement obtained by using the additional IP distance features (all boxes except the leftmost one), not only by increasing the median value but also by reducing the dispersion of values.

Prediction Metrics (2 labels)			Feature Sets											
			A No IP and Port features	B IP and Port frequency features	C New transformed features (IP,IP+Port),1000,3	D New transformed features (IP,Port,IP+Port),1000,3	E New transformed features (IP,Port,IP+Port),3000,3	F New transformed features (IP,Port,IP+Port),5000,3	G New transformed features (IP,Port,IP+Port),3000,5	H New transformed features (IP,Port,IP+Port),3000,2	I New transformed features (Port,IP1,IP2,IP3,IP4),3000,3	J New transformed features (IP1,IP2,IP3,IP4),3000,3	K New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4),5000,5	L New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4),3000,3
	Adaboost	Accuracy	0.9956	0.91126	0.9980	0.9982	0.9983	0.9982	0.9982	0.9982	0.9978	0.9979	0.9989	0.9993
		F1	0.9946	0.88986	0.9976	0.9978	0.9979	0.9978	0.9978	0.9978	0.9973	0.9975	0.9986	0.9992
		Precision	0.9975	0.91330	1.0000	0.9999	0.9999	0.9999	0.9989	0.9998	1.0000	1.0000	0.9985	1.0000
		Recall	0.9918	0.86760	0.9951	0.9958	0.9959	0.9957	0.9967	0.9958	0.9946	0.9950	0.9987	0.9984
	Linear SVM	Accuracy	0.7263	0.69471	0.7424	0.7322	0.7226	0.8237	0.8326	0.7306	0.9694	0.9737	0.7763	0.9676
		F1	0.5052	0.45541	0.5477	0.5211	0.4951	0.7294	0.7464	0.5167	0.9616	0.9671	0.6288	0.9592
		Precision	0.9979	0.86603	0.9980	0.9981	0.9984	0.9972	0.9979	0.9985	0.9999	0.9999	1.0000	0.9999
		Recall	0.3382	0.30894	0.3774	0.3526	0.3292	0.5750	0.5961	0.3485	0.9260	0.9364	0.4586	0.9217
	Logistic Regression	Accuracy	0.9810	0.94985	0.9819	0.9847	0.9760	0.9935	0.9933	0.9776	0.9962	0.9964	0.9892	0.9962
		F1	0.9765	0.93706	0.9776	0.9811	0.9702	0.9922	0.9919	0.9722	0.9953	0.9957	0.9867	0.9953
		Precision	0.9982	0.97330	0.9977	0.9981	0.9982	0.9953	0.9956	0.9981	0.9994	0.9994	1.0000	0.9995
		Recall	0.9558	0.90342	0.9583	0.9647	0.9437	0.9891	0.9882	0.9475	0.9912	0.9920	0.9739	0.9912
	Random Forest	Accuracy	0.9979	0.99938	0.9982	0.9981	0.9981	0.9974	0.9977	0.9979	0.9993	0.9985	0.9984	0.9993
		F1	0.9975	0.99925	0.9978	0.9976	0.9977	0.9968	0.9972	0.9975	0.9991	0.9981	0.9980	0.9992
		Precision	0.9995	0.99993	1.0000	0.9998	1.0000	0.9997	0.9999	0.9998	0.9993	0.9998	0.9998	0.9998
		Recall	0.9955	0.99858	0.9957	0.9954	0.9954	0.9939	0.9944	0.9952	0.9989	0.9964	0.9963	0.9986
	GBM	Accuracy	0.9968	0.71745	0.9982	0.9982	0.9982	0.9982	0.9982	0.9982	0.9993	0.9993	0.9993	0.9993
		F1	0.9962	0.48089	0.9978	0.9979	0.9979	0.9979	0.9978	0.9979	0.9992	0.9992	0.9992	0.9992
		Precision	0.9996	0.99825	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999	0.9998	0.9999
		Recall	0.9927	0.31673	0.9958	0.9958	0.9958	0.9959	0.9958	0.9958	0.9984	0.9984	0.9985	0.9985
	MLP	Accuracy	0.9956	0.99220	0.9981	0.9975	0.9981	0.9969	0.9977	0.9981	0.9993	0.9990	0.9982	0.9988
		F1	0.9946	0.99058	0.9977	0.9970	0.9977	0.9962	0.9972	0.9977	0.9991	0.9988	0.9979	0.9986
		Precision	0.9957	0.98836	0.9998	0.9999	0.9998	0.9996	0.9995	0.9999	0.9998	0.9992	0.9999	0.9988
		Recall	0.9936	0.99281	0.9955	0.9941	0.9957	0.9928	0.9949	0.9955	0.9985	0.9984	0.9958	0.9984
	Median (F1)		0.9946	0.91346	0.9976	0.9973	0.9977	0.9965	0.9972	0.9976	0.9982	0.9978	0.9980	0.9989
	MAD (F1)		0.0022	0.08145	0.0002	0.0005	0.0002	0.0013	0.0006	0.0002	0.0009	0.0012	0.0009	0.0003
	Mean (F1)		0.9108	0.79218	0.9194	0.9154	0.9094	0.9517	0.9547	0.9133	0.9919	0.9927	0.9349	0.9918
	Std. Desv. (F1)		0.1988	0.25420	0.1823	0.1933	0.2033	0.1089	0.1021	0.1946	0.0150	0.0126	0.1500	0.0160
	Median (% Increase v.s. No IP&Port features)		0.00%	-8.16%	0.30%	0.27%	0.31%	0.19%	0.25%	0.30%	0.36%	0.32%	0.33%	0.43%
	Mean (% Increase v.s. No IP&Port features)		0.00%	-13.02%	0.94%	0.51%	-0.15%	4.49%	4.82%	0.28%	8.91%	9.00%	2.65%	8.89%

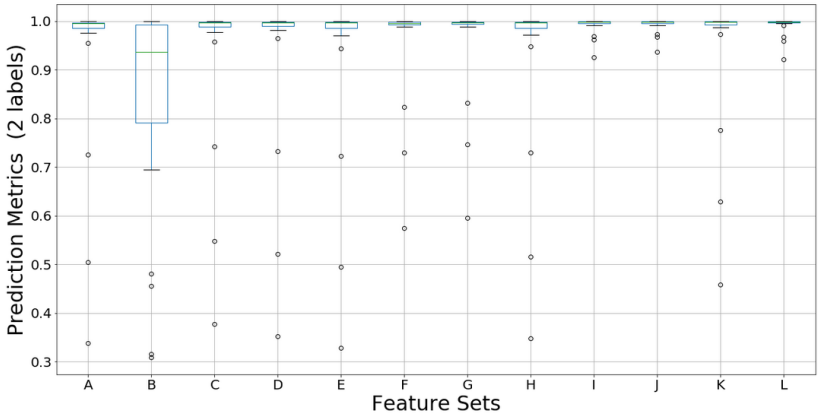

Fig 7. (Upper part) Prediction results table for different feature sets options (2 labels detection). (Lower part) Box-plots for the prediction results in the above table. We can observe the improvement obtained by using the additional IP distance features (all boxes except the leftmost one), not only by increasing the median value but also by reducing the dispersion of values.

In addition, the new transformed features rank among the most discriminant features using the scores of feature importance provided by Random Forest (based on Gini impurity)[34](Fig. 8). This is important, considering that the original numeric features obtain already high detection results, which implies that they are already very relevant.

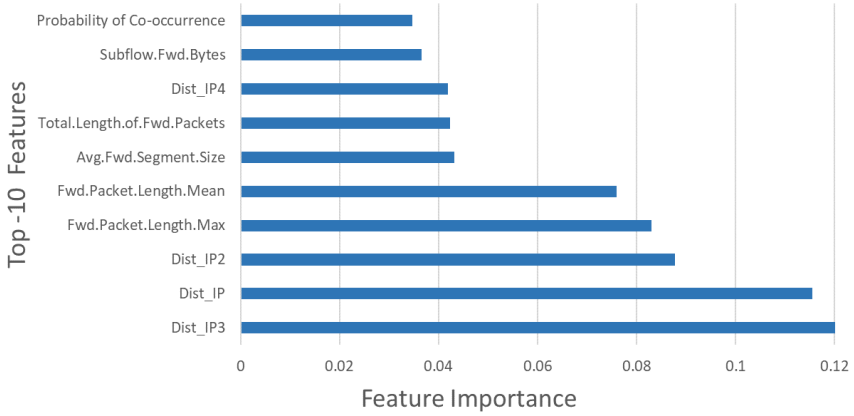

Fig 8. Diagram showing the 10 most relevant features, using impurity-based feature importance provided by Random Forest. The cosine distance for the embeddings of IP, IP2, IP3, IP4, and the output probability of the Encoding model (Prob. of Co-occurrence) are among the 10 most relevant features. A higher value implies a more important feature.

To ensure that the results obtained when using the new features are significantly better (from a statistical point of view) than those obtained by not using them, we have applied the Wilcoxon one-sided paired rank-sum test for the comparison of performance metrics between options A ("No IP and Port features") and L (option with best results that includes new transformed features). Fig. 9 presents the results for applying this test between the two options (A and L) and the two detection scenarios (4 and 2 labels). Test results are presented in the left and right parts of Fig. 9, where the p-value indicates if the results allow (or not) to reject the null hypothesis that is associated with a non-significant difference in the results. The test used is one-side to specifically check if one of the groups has higher ranked mean that the other. From the results in Fig. 9, using a significance level of 1%, we can infer that the prediction metrics obtained by adding the new features are significantly better than the alternative option.

Wilcoxon one-sided paired rank-sum test for difference of results between models:			
4 Labels No IP and Port features vs. New transformed features (option L)		2 Labels No IP and Port features vs. New transformed features (option L)	
p-value	Significant results?	p-value	Significant results?
8.166E-06	Yes	9.692E-06	Yes

Fig 9. Results of the Wilcoxon paired rank-sum test to check significance of the better results obtained when adding the proposed transformed features.

4.1.2 IP addresses co-occurrence prediction

The same features obtained with the Encoding model (Section 3.3) can also be used to predict whether a pair of source and destination network addresses will be part of a new test connection. The prediction results are obtained with a test set formed by source and destination addresses never seen during training, some of them associated to real traffic and others to randomly selected addresses (fake connections). All source and destination addresses (for real and fake connections) are extracted from the set of network addresses in CICIDS2017.

The prediction is made with the model proposed in Section 3.3, where the output probability produced by the model corresponds to the probability that the source and destination addresses belong to the same connection (co-occurrence). Fig. 10 presents the results obtained by using different feature set options (described in Section 4.1.1). We have chosen the most representative set of options. These options form the columns in Fig. 10 with the rows providing different prediction metrics. The first column in Fig. 10 corresponds to a dummy classifier that makes random predictions for each pair of network addresses in the test set; this is because in this case you do not have any IP and Port information to base your prediction on. The reason for not including option B ("IP and Port frequency features") in Fig. 10 is because this option provides IP and Port frequency features, but not the frequency for the combination of source and destination addresses, which would have produced extremely small and similar frequency values.

When analyzing the results of Fig. 10 is interesting to see the excellent prediction metrics obtained when using the new IP distance features. The best results are again given for the same selection of hyper-parameters for the model that gave already best results for improving network intrusion detection. The good results provide an additional indication that the proposed distances

contain (in a reduced set of values) valuable information to discriminate between real and false connections within a huge combinatorial space.

The results in Fig. 10 have been obtained with the dataset CICIDS2017 (Section 3.1.1).

	Feature Sets						
	A No IP and Port features	D New transformed features (IP,Port,IP+Port),1000,3	E New transformed features (IP,Port,IP+Port),3000,3	G New transformed features (IP,Port,IP+Port),3000,5	H New transformed features (IP,Port,IP+Port),3000,2	K New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4),5000,5	L New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4),3000,3
Accuracy	0.5000	0.8250	0.8585	0.6930	0.8368	0.6805	0.8569
F1	0.4994	0.8417	0.8716	0.6496	0.8556	0.6280	0.8703
Precision	0.5000	0.7683	0.7978	0.7565	0.7672	0.7516	0.7958
Recall	0.4989	0.9306	0.9605	0.5691	0.9670	0.5393	0.9601
AUC	0.5002	0.8999	0.9125	0.8493	0.9077	0.8358	0.9184

Fig 10. Table of prediction results for the detection of real vs. false pairs of source and destination addresses

4.2 Results for the CICDDoS2019 dataset

All the results presented in this section have been produced with the dataset CICDDoS2019 (Section 3.1.2). Similar to the results attained in Sections 4.1.1 and 4.1.2 (for the CICIDS2017 dataset), this section presents the results obtained by using the new distance features to improve the prediction results of several classic ML models (Section 4.2.1), and classify a pair of source and destination network addresses as real or false (Section 4.2.2).

4.2.1 NID prediction

We will apply the new features generated by our model to the two detection scenarios described in Section 3.1.2: a) detection of Normal and Attack traffic and b) detection of Normal, Syn, UDP and UDPLag attacks. The first scenario poses a binary classification problem with a very unbalanced dataset, and the second scenario a similarly highly unbalanced multi-class detection problem. The results obtained for the first scenario are presented in Fig. 12, and the second in Fig. 11. The meaning and structure of the diagrams in Fig. 11 and 12 are the same used for Fig. 6 and 7 in Section 4.1.1.

Fig. 11 presents the prediction metrics of six ML models to classify the 4 labels provided by CICDDoS2019. The first column presents the results using no information about network addresses (Option A), the second column presents the results using the IP and Port frequency features, and the third column using the transformed features explained in Section 3.3. The hyper-parameter values used to obtain the features in the third column correspond to the combination described as option L in Section 4.1.1. In this case we will only show the prediction metrics with the feature set that provides the best results for CICIDS2017. Our intention with CICDDoS2019 is to show the good generalization of the results obtained for CICIDS2017.

Looking at Fig. 11, we can see that the prediction results of option A are already quite high, without using the new features, and they improve even more when they are used. The boxplots to the right of Fig. 11 also give a visual indication of the improvement in the distribution of results statistics when the new features are used. We obtain an improvement in the results of 5% on average (Fig. 11 bottom-left).

CICDDoS2019 is an extremely unbalanced dataset for the 2-labels scenario, with extremely good classification results without using the new features and, almost perfect predictions when using them, this is the reason why, even when there is an improvement when using the new features, this improvement is not evident (Fig. 12). The box-plots for these results (right part of Fig. 12) present only some outliers with the interquartile intervals collapsing into a single value. These graphic results make it difficult to appreciate the improvement obtained when adding the new features for the 2-labels scenario. We need to resort to a statistical hypothesis test, such as the Wilcoxon test, to verify the significance of the improvement (Fig. 13). We have applied the Wilcoxon one-sided paired rank-sum test for the comparison of performance metrics. Fig. 13 presents the Wilcoxon test results applied to the two options: options A ("No IP and Port features") and L (option with the new transformed features) and the two detection scenarios (4 and 2 labels) for the CICDDoS2019 dataset. The test used is one-side to specifically check if one of the groups has higher ranked mean that the other. The hypothesis tests results are presented in the left and right parts of Fig. 13. Since the p-value obtained is smaller than the usual significance level of 1%, we can conclude that the prediction metrics obtained by adding the new features (option L) are significantly better than the alternative option (option A) for the two scenarios (4 and 2 labels) with the CICDDoS2019 dataset.

Prediction Metrics (4 labels)		Feature Sets: A No IP and Port features	B IP and Port frequency features	L New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4), 3000,3
Adaboost	Accuracy	0.8649	0.9969	0.9969
	F1	0.5789	0.7478	0.7478
	Precision	0.7450	0.7457	0.7457
	Recall	0.7724	0.7498	0.7498
Linear SVM	Accuracy	0.9972	0.9966	0.9973
	F1	0.8456	0.8269	0.8601
	Precision	0.8940	0.8669	0.8983
	Recall	0.8242	0.7973	0.8369
Logistic Regression	Accuracy	0.9971	0.9962	0.9972
	F1	0.8452	0.8149	0.8518
	Precision	0.8889	0.8560	0.8927
	Recall	0.8229	0.7830	0.8281
Random Forest	Accuracy	0.9973	0.9975	0.9974
	F1	0.8297	0.8601	0.8428
	Precision	0.9081	0.9279	0.9326
	Recall	0.8034	0.8295	0.8126
GBM	Accuracy	0.9919	0.9965	0.9965
	F1	0.5879	0.7496	0.8513
	Precision	0.7516	0.8307	0.8920
	Recall	0.5610	0.7606	0.8267
MLP	Accuracy	0.9974	0.9970	0.9974
	F1	0.8605	0.7833	0.8618
	Precision	0.8955	0.8985	0.9003
	Recall	0.8395	0.7693	0.8392
Median (F1)		0.8375	0.7991	0.8516
MAD (F1)		0.0156	0.0387	0.0086
Mean (F1)		0.7580	0.7971	0.8359
Std. Desv. (F1)		0.1356	0.0449	0.0437
Median (% Increase v.s. No IP&Port features)		0.00%	-4.58%	1.68%
Mean (% Increase v.s. No IP&Port features)		0.00%	5.16%	10.28%

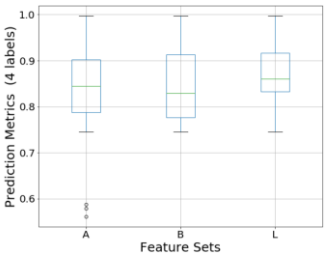

Fig 11. Prediction results table for different feature sets options (4 labels detection) for the CICDDoS2019 dataset (left diagram) and their associated box-plots (right diagram). There is an improvement obtained by using the new transformed features, not only by increasing the median value but also by reducing the dispersion of values.

Prediction Metrics (2 labels)		Feature Sets: A No IP and Port features	B IP and Port frequency features	L New transformed features (IP,Port,IP+Port,IP1,IP2,IP3,IP4), 3000,3
Adaboost	Accuracy	0.9999	1.0000	1.0000
	F1	1.0000	1.0000	1.0000
	Precision	0.9999	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000
Linear SVM	Accuracy	0.9999	0.9975	1.0000
	F1	1.0000	0.9988	1.0000
	Precision	1.0000	0.9977	1.0000
	Recall	0.9999	0.9998	1.0000
Logistic Regression	Accuracy	0.9998	0.9986	0.9999
	F1	0.9999	0.9993	1.0000
	Precision	0.9999	0.9986	0.9999
	Recall	0.9999	1.0000	1.0000
Random Forest	Accuracy	1.0000	1.0000	1.0000
	F1	1.0000	1.0000	1.0000
	Precision	1.0000	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000
GBM	Accuracy	0.9967	0.9990	1.0000
	F1	0.9984	0.9995	1.0000
	Precision	1.0000	1.0000	1.0000
	Recall	0.9967	0.9990	1.0000
MLP	Accuracy	0.9999	0.9999	1.0000
	F1	1.0000	1.0000	1.0000
	Precision	1.0000	1.0000	1.0000
	Recall	1.0000	1.0000	1.0000
Median (F1)		1.0000	0.9998	1.0000
MAD (F1)		0.0001	0.0003	0.0001
Mean (F1)		0.9997	0.9996	1.0000
Std. Desv. (F1)		0.0007	0.0006	0.0001
Median (% Increase v.s. No IP&Port		0.00%	-0.02%	0.01%
Mean (% Increase v.s. No IP&Port		0.00%	-0.01%	0.03%

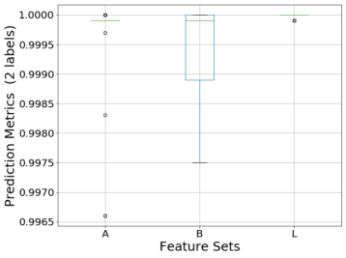

Fig 12. Prediction results table for different feature sets options (2 labels detection) for the CICDDoS2019 dataset (left diagram) and their associated box-plots (right diagram). There is an improvement obtained by using the new transformed features, but it is not noticeable due to the imbalance of the dataset and the already high prediction results using only the numerical features.

Wilcoxon one-sided paired rank-sum test for difference of results between models:			
4 Labels No IP and Port features vs. New transformed features (option L)		2 Labels No IP and Port features vs. New transformed features (option L)	
p-value	Significant results?	p-value	Significant results?
1.969E-04	Yes	3.003E-04	Yes

Fig. 13. Results of the Wilcoxon paired rank-sum test to check significance of the better results obtained when adding the new transformed features (CICDDoS2019)

Similar to what happens with CICIDS2017 (Section 4.1.1), we can obtain a feature importance score for the new features using several algorithms (e.g. Random Forest, GBM). Fig. 14 presents the feature importance scores for all the features of CICDDoD2019, including the proposed new features. We can observe that the new features are among the most important, despite the excellent prediction results already obtained with the original numeric features. The feature importance is obtained using Random Forest (based on Gini impurity)[34].

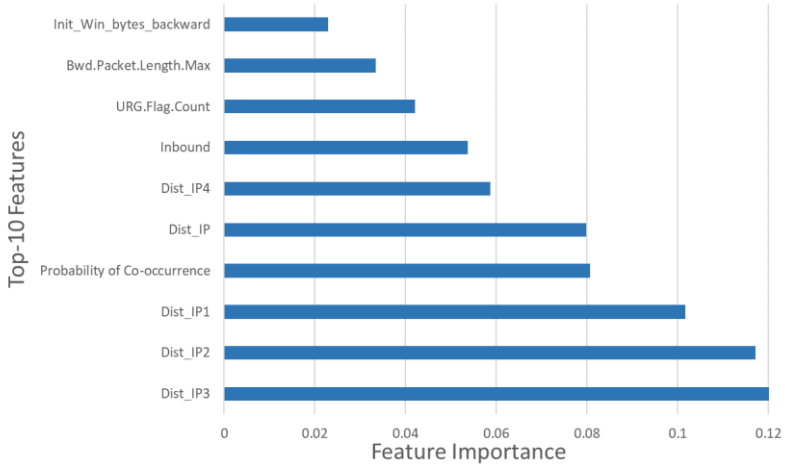

Fig. 14.In the same way as the results obtained for CICIDS2017, with CICDDoS2019 the cosine distance for the embeddings of IP, IP1, IP2, IP3, IP4, and the output probability of the proposed model (Prob. of Co-occurrence), are among the 10 most relevant features (feature importance provided by Random Forest).

4.2.2 IP addresses co-occurrence prediction

In the same way as presented in Section 4.1.2, the same features obtained with the proposed model (Section 3.3) can also be used to predict whether a pair of source and destination addresses are part of the same connection. As in Section 4.1.2, we will use a test set formed by source and destination addresses never seen during training, some of them associated to real traffic and others to randomly selected addresses (fake connections). All source and destination addresses (for real and fake connections) are extracted from the set of network addresses in CICDDoS2019.

Fig. 15 provides the prediction results for the detection of real vs. fake pairs of source and destination addresses for the CICDDoS2019 dataset. Prediction metrics are high and even better than with CICIDS2017 (Section 4.1.2), which is understandable since DDoS attacks have a more clearly marked pattern based on the addresses of the attacker and victim.

	Feature Sets	
	A No IP and Port features	L New transformed features (IP, Port, IP+Port, IP1, IP2, IP3, IP4), 3000, 3
Accuracy	0.5002	0.9801
F1	0.4996	0.9805
Precision	0.5002	0.9618
Recall	0.4990	0.9999
AUC	0.5004	0.9996

Fig. 15. Table of prediction results for the detection of real vs. false pairs of source and destination addresses (CICDDoS2019)

5. CONCLUSION

The transformation of IP and Port network addresses into features that can be easily incorporated into machine learning models is an open problem with different current proposals. This work presents a novel alternative treatment of these features that shows several advantages over current methods, and provides a significant improvement in the prediction metrics of modern classifiers when including the new transformed features.

The source and destination IP and Port addresses are categorical variables of high cardinality. This work proposes a method to encode them into a small set of scalar values that represent a 'distance' between pairs of source and destination network address elements (NAEs). The proposed encoding model (a neural network) is trained using real and fake connections in a self-supervised framework and its output represents the probability that the source and destination addresses share a connection. To create an operational architecture, we apply hash functions to reduce the initial range of values of some NAEs and an embedding transformation to all NAEs, to allow a semantically meaningful 'distance' calculation. The benefits of the proposed method are: (1) Distances are constructed to represent the likelihood that source and destination addresses share a connection. (2) The use of these distances as new features (replacing the original addresses) provides an improvement in the prediction capabilities of the ML models. (3) The embedding tables are very small (thanks to the hash function). (4) Low resources required for the encoding model. (5) The nature of the encoding model, which is based on a neural network, allows incremental training, this together with the previous two points allows easy re-training of the model, which is needed in all data network applications with a constantly changing network traffic.

The experimental results that have been obtained using two different datasets (CICIDS2017 and CICDDoS2019) show that the transformed features significantly improve the classification (detection) metrics provided by most of the usual ML models. Likewise, the results demonstrate that the probability estimate given by the proposed model is an excellent predictor of the probability that two network addresses will share the same network connection.

As future lines of work, we intend to explore the application of new embedding frameworks based on recent ideas that consider the context more extensively, such as the attention mechanism [35] and subsequent developments e.g. the Transformer architecture [36].

ACKNOWLEDGEMENTS

The preparation of the article and the study of algorithms were funded with grant RTI2018-098958-B-I00 from Proyectos de I+D+i «Retos investigación», Programa Estatal de I+D+i Orientada a los Retos de la Sociedad, Plan Estatal de Investigación Científica, Técnica y de Innovación 2017-2020. Spanish Ministry for Science, Innovation and Universities; the Agencia Estatal de Investigacón (AEI) and the Fondo Europeo de Desarrollo Regional (FEDER).

REFERENCES

[1] M. Wang, Y. Cui, X. Wang, S. Xiao, J. Jiang, Machine Learning for Networking: Workflow, Advances and Opportunities, IEEE Netw. 32 (2018) 92–99. https://doi.org/10.1109/MNET.2017.1700200.

[2] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O.M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, J. Internet Serv. Appl. 9 (2018) 16. https://doi.org/10.1186/s13174-018-0087-2.

[3] K. Potdar, T.S. Pardawala, C.D. Pai, A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers, Int. J. Comput. Appl. 175 (2017) 7–9. https://doi.org/10.5120/ijca2017915495.

[4] M.J. Davis, Contrast Coding in Multiple Regression Analysis: Strengths, Weaknesses, and Utility of Popular Coding Structures, J. Data Sci. 8 (2010) 61–73.

[5] P. Cerda, G. Varoquaux, Encoding high-cardinality string categorical variables, IEEE Trans. Knowl. Data Eng. (2019) 1–1. https://doi.org/10.1109/TKDE.2020.2992529.

[6] H. Jia, Y. Cheung, J. Liu, A New Distance Metric for Unsupervised Learning of Categorical Data, IEEE Trans. Neural Networks Learn. Syst. 27 (2016) 1065–1079. https://doi.org/10.1109/TNNLS.2015.2436432.

[7] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, J. Attenberg, Feature Hashing for Large Scale Multitask Learning, in: Proc. 26th Annu. Int. Conf. Mach. Learn., Association for Computing Machinery, New York, NY, USA, 2009: pp. 1113–1120. https://doi.org/10.1145/1553374.1553516.

[8] A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, in: CRC Press, 1997. www.cacr.math.uwaterloo.ca/hac.

[9] D. Micci-Barreca, A Preprocessing Scheme for High-Cardinality Categorical Attributes in Classification and Prediction Problems, SIGKDD Explor. Newsl. 3 (2001) 27–32. https://doi.org/10.1145/507533.507538.

[10] W. Duch, K. Grudzinski, G. Stawski, Symbolic Features In Neural Networks, in: Proc. 5th Conf. Neural Networks Their Appl., 2000: pp. 180–185.

[11] K. Grakabczewski, N. Jankowski, Transformations of Symbolic Data for Continuous Data Oriented Models, in: O. Kaynak, E. Alpaydin, E. Oja, L. Xu (Eds.), Artif. Neural Networks Neural Inf. Process. --- ICANN/ICONIP 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003: pp. 359–366.

[12] M.-L. Shyu, K. Sarinnapakorn, I. Kuruppu-Appuhamilage, S.-C. Chen, L. Chang, T. Goldring, Handling nominal features in anomaly intrusion detection problems, in: 15th Int. Work. Res. Issues Data Eng. Stream Data Min. Appl., 2005: pp. 55–62. https://doi.org/10.1109/RIDE.2005.10.

[13] Z. Sulc, H. Rezanková, Dimensionality Reduction of Categorical Data: Comparison of HCA and CATPCA Approaches, in: Conf. Appl. Math. Stat. Econ. 2015, 2015: p. Vol. 18.

[14] J. Wieting, D. Kiela, No Training Required: Exploring Random Encoders for Sentence Classification, 7th Int. Conf. Learn. Represent. ICLR 2019. (2019). http://arxiv.org/abs/1901.10444 (accessed September 17, 2020).

[15] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: 1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc., International Conference on Learning Representations, ICLR, 2013. http://ronan.collobert.com/senna/ (accessed September 17, 2020).

[16] C. Guo, F. Berkhahn, Entity Embeddings of Categorical Variables, (2016). http://arxiv.org/abs/1604.06737 (accessed September 17, 2020).

[17] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, 4th Int. Conf. Inf. Syst. Secur. Priv. (2018). https://doi.org/10.5220/0006639801080116.

[18] I. Sharafaldin, A.H. Lashkari, S. Hakak, A.A. Ghorbani, Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy, in: 2019 Int. Carnahan Conf. Secur. Technol., 2019: pp. 1–8. https://doi.org/10.1109/CCST.2019.8888419.

[19] S. Mathur, B. Coskun, S. Balakrishnan, Detecting Hidden Enemy Lines in IP Address Space, in: Proc. 2013 New Secur. Paradig. Work., Association for Computing Machinery, New York, NY, USA, 2013: pp. 19–30. https://doi.org/10.1145/2535813.2535816.

[20] B. Li, M.H. Gunes, G. Bebis, J. Springer, A Supervised Machine Learning Approach to Classify Host Roles on Line Using SFlow, in: Proc. First Ed. Work. High Perform. Program. Netw., Association for Computing Machinery, New York, NY, USA, 2013: pp. 53–60. https://doi.org/10.1145/2465839.2465847.

[21] H. Jiang, Y. Liu, J.N. Matthews, IP geolocation estimation using neural networks with stable landmarks, in: 2016 IEEE Conf. Comput. Commun. Work. (INFOCOM WKSHPS), 2016: pp. 170–175. https://doi.org/10.1109/INFCOMW.2016.7562066.

[22] J. Wang, I.C. Paschalidis, Botnet Detection Based on Anomaly and Community Detection, IEEE Trans. Control Netw. Syst. 4 (2017) 392–404. https://doi.org/10.1109/TCNS.2016.2532804.

[23] S.E. Coull, F. Monrose, M. Bailey, On Measuring the Similarity of Network Hosts: Pitfalls, New Metrics, and Empirical Analyses, 2011.

[24] A. Jakalan, J. Gong, Q. Su, X. Hu, A.M.S. Abdelgder, Social relationship discovery of IP addresses in the managed IP networks by observing traffic at network boundary, Comput. Networks. 100 (2016) 12–27. https://doi.org/https://doi.org/10.1016/j.comnet.2016.02.012.

[25] M. Ring, A. Dallmann, D. Landes, A. Hotho, IP2Vec: Learning Similarities Between IP Addresses, in: 2017 IEEE Int. Conf. Data Min. Work., 2017: pp. 657–666. https://doi.org/10.1109/ICDMW.2017.93.

[26] Y. Goldberg, O. Levy, word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method, (2014). http://arxiv.org/abs/1402.3722 (accessed September 17, 2020).

[27] mlopezm/Monte-Carlo-simulation-for-hash-collision: Monte carlo simulation for NAEs hash collision, (n.d.). https://github.com/mlopezm/Monte-Carlo-simulation-for-hash-collision.

[28] B.H. Bloom, Space/Time Trade-Offs in Hash Coding with Allowable Errors, Commun. ACM. 13 (1970) 422–426.

https://doi.org/10.1145/362686.362692.
[29] D. Eddelbuettel, A. Lucas, J. Tuszynski, H. Bengtsson, S. Urbanek, M. Frasca, B. Lewis, M. Stokely, H. Muehleisen, D. Murdoch, J. Hester, W. Wu, Q. Kou, T. Onkelinx, M. Lang, V. Simko, K. Hornik, R. Neal, K. Bell, M. De Queljoe, I. Suruceanu, B. Denney, R Package "digest" , 2020. https://cran.r-project.org/web/packages/digest/digest.pdf (accessed September 17, 2020).
[30] A. Appleby, SMHasher & MurmurHash, Github. (2016). https://github.com/aappleby/smhasher .
[31] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, in: 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., International Conference on Learning Representations, ICLR, 2015. https://arxiv.org/abs/1412.6980v9 (accessed September 21, 2020).
[32] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, ArXiv. (2016). https://arxiv.org/abs/1603.04467 (accessed September 17, 2020).
[33] F. Pedregosa, V. Michel, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, J. Vanderplas, D. Cournapeau, F. Pedregosa, G. Varoquaux, A. Gramfort, B. Thirion, O. Grisel, V. Dubourg, A. Passos, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830. http://scikit-learn.sourceforge.net. (accessed September 17, 2020).
[34] G. Louppe, L. Wehenkel, A. Sutera, P. Geurts, Understanding Variable Importances in Forests of Randomized Trees, in: Proc. 26th Int. Conf. Neural Inf. Process. Syst. - Vol. 1, Curran Associates Inc., Red Hook, NY, USA, 2013: pp. 431–439.
[35] D. Bahdanau, K. Cho, Y. Bengio, Neural Machine Translation by Jointly Learning to Align and Translate, ArXiv. (2014). http://arxiv.org/abs/1409.0473 (accessed October 8, 2020).
[36] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention Is All You Need, 2017.