Autoencoder-based Deep Metric Learning for Network Intrusion Detection

Giuseppina Andresini[a,*], Annalisa Appice[a,b], Donato Malerba[a,b]

[a] *Department of Informatics, Università degli Studi di Bari Aldo Moro, via Orabona, 4 - 70125 Bari - Italy*
[b] *Consorzio Interuniversitario Nazionale per l'Informatica - CINI, Italy*

Abstract

[1] Nowadays intrusion detection systems are a mandatory weapon in the war against the ever-increasing amount of network cyber attacks. In this study we illustrate a new intrusion detection method that analyses the flow-based characteristics of the network traffic data. It learns an intrusion detection model by leveraging a deep metric learning methodology that originally combines autoencoders and Triplet networks. In the training stage, two separate autoencoders are trained on historical normal network flows and attacks, respectively. Then a Triplet network is trained to learn the embedding of the feature vector representation of network flows. This embedding moves each flow close to its reconstruction, restored with the autoencoder associated with the same class as the flow, and away from its reconstruction, restored with the autoencoder of the opposite class. The predictive stage assigns each new flow to the class associated with the autoencoder that restores the closest reconstruction of the flow in the embedding space. In this way, the predictive stage takes advantage of the embedding learned in the training stage, achieving a good prediction per-

*Fully documented templates are available in the elsarticle package on CTAN.

*Corresponding author (Tel: +39 (0)805443262 Fax: +39(0)805443269)
Email addresses: giuseppina.andresini@uniba.it (Giuseppina Andresini), annalisa.appice@uniba.it (Annalisa Appice), donato.malerba@uniba.it (Donato Malerba)

[1] This version of the contribution has been accepted for publication, after peer review but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1016/j.ins.2021.05.016. Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.elsevier.com/about/policies-and-standards/copyright

formance in the detection of new signs of malicious activities in the network traffic. In fact, the proposed methodology leads to better predictive accuracy when compared to competitive intrusion detection architectures on benchmark datasets.

Keywords: Network Intrusion Detection, Deep metric learning, Triplet network, Autoencoder

1. Introduction

Metric learning (ML) is a machine learning paradigm based on distance metrics. It aims to measure the similarity between samples by reducing the distance between similar samples and increasing the distance between dissimilar samples [38]. In recent years, the success of deep learning (DL) has led to the introduction of deep metric learning (DML), that combines deep learning and metric learning [50].

To improve the performance, traditional ML approaches need to learn a linear mapping to project samples into a non-linear feature space that often is not directly within the classification structure. On other hand, DL techniques are able to process nonlinear, raw data to extract a useful representation of data directly in the classification structure [38]. Therefore, DML approaches allow us to deal with the non-linearity problem by taking advantage of the structure of DL networks, thereby addressing one of the major problems affecting conventional ML algorithms [38].

In particular, DML approaches improve the performance of traditional ML approaches by taking advantage of the structure of DL networks to process nonlinear, raw data and extract a useful representation of data directly in the classification structure [38]. However, while DL performance declines significantly in the case of learning from imbalanced data [37], DML approaches can be well-suited to deal with the class imbalance condition [23]. The common mechanism of deep-based metric learning algorithms is to train a deep network to produce an embedding of the input vector, so that a loss function related to

sample distance can be minimised.

Up to now DML approaches have been mainly investigated in computer vision to address problems of face verification and face identification [38].

In these problems DML techniques are used to perform an end-to-end deep learning process that compares two images on an embedding feature space. Specifically, DML approaches learn hierarchical non-linear transformations, to map data samples into a new feature space that is more suitable for comparing or matching operations. This is done by exploiting deep neural network architectures that minimise a loss function related to a sample distance and unify feature learning and metric learning into a joint learning framework [50].

There are several state-of-the-art deep networks

(e.g. Siamese network and Triplet network) that are being used for DML[32, 67, 31]. [32, 67, 31].

As pointed out in [38], the performance of these DML networks basically depends on a metric loss function (e.g. contrastive loss, triplet loss) and a sampling strategy. These are considered as a whole with all of the components of the network. In fact, the way in which samples are presented to the network and the relationship between them are both related to the metric loss function. In this paper, we take the study of DML outside computer vision by focusing on the task of network intrusion detection. This is a crucial cyber-security problem, where deep learning is recognised as a relevant approach to learn signs of malicious activity [13], also under drifting conditions (e.g. new variants of seen attacks) [20].

Although network traffic produces large amounts of data to fuel deep learning, one issue to address is the imbalanced condition of network data—network intrusions and malicious behaviour commonly represent a very small subset of all the network traffic [47]. [43]. A few recent studies have explored approaches to re-sample training data [34, 47] [83, 43] and make deep learning robust to intrusion imbalance. Both down-sampling and over-sampling techniques are often used in combination with deep neural networks to deal with the imbalance data in various domains (e.g. in medical domains [84, 35]). However, an inevitable

consequence of down-sampling is the loss of information, while generating artificial samples may cause overfitting, noise or class overlap.

In particular, In this study, we investigate DML as a means to deal with the phenomenon of intrusion class imbalance thanks to performing deep learning DL in new feature spaces that maximize distances between opposite samples (i.e. between samples belonging to the normal class and the attack class, respectively), independently of the amount of data in each class.

We note that in DML a new feature space representation is learned, while the number of training samples is unchanged. Therefore, DML approaches As DML approaches learn new feature space representations without changing the number of training samples, they allow us to learn a classifier which is robust to imbalance without suffering from the traditional issues of re-sampling (loss of information, overfitting or noise). In particular, we propose a novel methodology, named RENOIR (autoencodeR-based neural nEtwork for INtrusiOn DetectIon Systems with tRiplet loss function), that defines an innovative Triplet network approach for network intrusion detection. It processes the flow-based characteristics of the network traffic data to detect signs of malicious activities.

The Triplet network [32], together with the Siamese network [14], is one of the most popular DML techniques. However, several recent studies have proved that Triplet networks are, in general, more effective than Siamese networks [22, 16] [22, 63, 16], due to their simultaneous learning of similar and dissimilar relationships. Specifically, a Triplet network uses triplets of samples (instead of pairs, as in Siamese networks). Every triplet is commonly composed of a training sample selected as an anchor, a training sample labelled in the same class as the anchor, as well as a training sample labelled in the opposite class. The Triplet network takes triplets as input and learns an embedding space, where distances between samples labelled with opposite classes are greater than distances between samples labelled with the same class. Although Triplet networks are emerging as a relevant approach in DML, existing approaches based on Triplet networks commonly suffer from poor convergence. The convergence problem is mainly caused by a random selection of samples for the triplet construction in

the training set [78]. In fact, the traditional Triplet network implementations randomly select a single training sample labelled in the same class as the anchor and a single training sample labelled in the opposite class.

The main innovation of this study is the introduction of a new triplet construction strategy based on autoencoders, to handle the problem of triplet convergence and make effective the use of Triplet networks for network intrusion detection. This strategy learns two independent autoencoders on normal flows and attacks, separately. It uses these autoencoders, instead of traditional sampling, to derive both the positive and negative information for the triplet construction. In particular, every triplet is constructed considering the positive and negative pseudo-samples that are the unique reconstructions of the considered anchor, restored through the two autoencoders. So, these autoencoder-originated pseudo-samples replace the real training samples that the traditional Triplet networks would randomly select from the normal part and the attack part of the training set.

We point out that our methodology renounces any random choice during the triplet construction. In addition, autoencoders capture class-specific information, to facilitate the disentanglement among the classes. In principle, the autoencoder trained on the normal samples can contribute to recovering denoised normal samples, but it should see attacks as anomalies, and so reconstruct them badly. Vice-versa, the autoencoder trained on the attacks should denoise the attacks, seeing the normal flows as anomalies and reconstructing them badly. Basing the triplet construction, and consequently the triplet-learned embedding, on the pseudo-samples reconstructed through these two autoencoders allows us to exploit possible patterns existing among the normal and attack classes, given the class of the anchor sample of the triplet.

A further advantage of our proposal is that we also leverage the data compression ability of autoencoder models to speed up the classification of any new query sample. Traditional Triplet network approaches predict the class of a query sample by performing a time-consuming k-nearest neighbour (kNN) search within the triplet embedding of the training set [36, 78]. However, our ap-

proach Our methodology assigns a query sample to the class of the autoencoder that yields the nearest-neighbour pseudo-sample of the query sample under the umbrella of the triplet-learned embedding. Therefore,

thanks to the consideration of autoencoder models instead of training data, our prediction strategy can actually reduce the effort spent with a traditional kNN search, by exploring the distance between a query sample and its two autoencoder-restored pseudo-samples only.

In short, the main contributions of this work are reported in the following.

- We define an innovative DML methodology for network intrusion detection problems, which couples autoencoders to Triplet networks, in order to address the problem of convergence during triplet learning. The autoencoder information contributes to the construction of an embedding space that potentially moves each flow close to its reconstruction, restored through the autoencoder associated with the same class, and away from its reconstruction, restored with the autoencoder of the opposite class. This fuels a mechanism to achieve better performance in separating classes, as proved in the experimental study.

- We formulate a novel, efficient strategy for predicting the class of any new network flow by taking advantage of the autoencoders learned from the two classes within the embedding learned on the triplet-space.

- We perform an extensive evaluation of the effectiveness of the presented methodology, by investigating the viability of the proposed learning components in the imbalance scenario, as well as the ability of our methodology to increase both accuracy and efficiency, compared to competitive, state-of-the-art approaches (comprising DML-based) taken from the recent literature on network intrusion detection.

- Although the main focus of this study is on binary classification (regardless of the attack type), we present a preliminary investigation of the potential of the binary methodology in multi-class classification. To this

aim, we perform an experiment, where we resort to a combination strategy (e.g. one-versus-all or one-versus-one) [49] to frame the proposed binary methodology in the multi-class scenario. This allows us to perform the task of attack type classification, in addition to attack detection.

This paper is organised as follows. The related works are presented in Section 2. The formulated machine learning methodology is described in Section 3, while the implementation details are reported in Section 4. The findings in the evaluation of the proposed binary strategy are discussed in Section 5, while the preliminary results achieved in the multi-class scenario are reported in Section 6. Finally, Section 7 refocuses on the purpose of the research, draws conclusions and proposes future developments.

2. Related works

Recent trends in cybersecurity research have shown machine learning and, in particular, DL to be an extremely relevant approach in network intrusion detection. In this paper we revamp an intrusion detection pipeline that is based on a DML architecture, therefore, we mainly focus the literature overview on recent studies applying DL (see Section 2.1) and DML (see Section 2.2), to discriminate intrusions from normal network traffic. In addition, as DML approaches are well-suited to deal with the class imbalance condition [23], we briefly revise the main approaches investigated in the recent literature to handle imbalanced data when training network intrusion detection patterns (see Section 2.3).

2.1. Deep learning (DL)

Deep learning DL is predominant in the recent literature on network intrusion detection. In particular, several recent studies [73, 30, 76] [56, 76, 13, 2, 29] have definitely assessed that DL techniques can achieve an important gain in accuracy compared to conventional machine learning techniques.

This is thanks to the capability of deep neural networks (DNNs) to deal with the high-dimensionality and non-linearity of a large volume of data to produce high-level data feature representations.

In recent years, many Many DNNs have been investigated regarding the problem of network intrusion detection. [6, 30, 7, 40, 21, 79, 80, 19].

In particular, several studies have investigated the use of Convolutional Neural Networks (CNNs) [30, 46, 40, 40], Long Short-Term Memory Networks (LSTMs) [21], Recurrent Neural Networks (RNNs) [21] and Generative Adversarial Networks (GAN) [60, 80, 79, 80, 19] for intrusion detection problems. Moreover, various recent studies have explored the use of deep autoencoder architectures both for anomaly detection [56, 65, 6] and feature extraction [39, 4] in network flow data. The authors of [81] experiment a stacked sparse autoencoder to extract relevant features that are taken as input from XGBoost. In particular, In a recent study [7], autoencoders are trained autoencoders are trained in [7] to build a multi-channel representation of network flows. This representation is used to train a multi-channel CNN for separating normal network flows from attacks. In [5], the authors extend the multi-channel CNN introduced in [7] with a mechanism to re-assign class labels of training normal samples that are close to the boundary with the opposite class, in order to deal with the class imbalance condition of network intrusions.

2.2. Deep metric learning (DML)

A few recent studies have started to investigate DML approaches in cybersecurity. Both Siamese networks [14] and Triplet networks [32] are the two main types of DNNs investigated in DML.

Both architectures are popular DL techniques that have had great success in many computer vision tasks [27, 54, 61, 41].

Siamese networks [67] learn a contrastive loss to quantify similarity between sample pairs. Although they are mainly investigated in computer vision [61, 41], speech recognition [26] and natural language processing [68], [77, 68], Recent studies have started exploring them in network intrusion detection [10, 11, 36, 55]. Specifically, Siamese networks are studied in [10, 11] to handle the class imbalance condition in network intrusion detection systems. Both studies solve a multi-class classification problem by coupling the attack classification to the

intrusion detection.

In [10], the authors prove that a Siamese network is able to classify rare attack class samples well, by taking advantage of both a down-sampling strategy in the training stage and a distance-based approach in the predictive stage.

In particular, a A Siamese network is trained on the pairs of similar samples (belonging to the same class) and the pairs of dissimilar samples (belonging to opposite classes), respectively. Sample pairs are constructed from a subset of training samples comprising 50 samples for each distinct class (Normal, DoS, Probe, R2L and U2R). To predict the class of a testing sample, a distance score is computed for each class. This score is based on the distances of the test sample from ten training samples sampled for the study class. The class with the best score is finally predicted.

In [11], the imbalanced condition is dealt with by opting for an ensemble, without resorting to any data-level balancing techniques. The authors prove that an ensemble of binary classification methods, which comprises a Siamese network, a DNN and an XGBoost binary classifier, can be used to gain accuracy in separating normal samples from intrusions. Network flows detected as intrusions by the ensemble are subsequently classified in different attack classes (DoS, Probe, R2L and U2R), using a multi-class XGBoost classifier.

In [36], a Siamese network is trained to find an embedding function that produces a feature representation of the data which minimises the distance between samples. The decision engine of this solution integrates a kNN classifier to yield final predictions. The proposed pipeline is tested to classify each network flow as normal or malicious (regardless of the attack type).

Finally, a Siamese convolutional network is also investigated in [55] for the dimensionality reduction in the binary classification of the network traffic. The proposed approach integrates both a fuzzy allocation scheme to transform raw data into fuzzy values and a modality transformation technique to convert the fuzzy memberships into a 2D image. In addition, it trains both a Siamese convolutional network to reduce the input data dimensionality to a 1D feature space and various machine learning models for the binary classification on the

resulting 1D space.

The proposed approach is four-stepped: (i) a fuzzy allocation scheme transforms raw data into fuzzy values; (ii) a modality transformation technique converts the fuzzy memberships into a 2D image; (iii) a Siamese convolutional network is trained to reduce the input data dimensionality to a 1D feature space; (iv) various machine learning models are trained to implement the binary classification task on the resulting 1D space.

Triplet networks [32] use triplets of samples (instead of pairs as in Siamese networks) to learn an embedding space, where distances between samples labelled with opposite classes are greater than distances between samples labelled with the same class. Several recent studies have proved that Triplet networks are, in general, more effective than Siamese networks [22, 16], due to their simultaneous learning of similar and dissimilar relations.

Triplet networks were first introduced for face recognition tasks [61]. In this seminal study on Triplet networks [61], the authors introduce the concept of a triplet loss function as a metric that, derived from nearest-neighbour classification, [75], allows us to project face imagery data onto a compact Euclidean space, where distances correspond to a measure of face similarity. Similarly to Siamese networks, Triplet networks are mainly used in computer vision [61, 67, 31]. However, a few studies investigate the making of this DML model in cybersecurity [9, 85]. [9, 85, 87, 44] For example, in [9], an adaptive Triplet loss is proposed to compare normal and attack workloads in web intrusion detection systems. In [85], a Triplet network is designed to improve the detection of anomalous in-vehicle CAN bus messages—a standardised serial communication protocol widely used in automobile internal control systems. According to the triplet theory, each triplet associates an annotated CAN bus message selected as an anchor, with both a positive message and a negative message. The triplet loss is trained to close the distance between the anchor and the positive message and extend the distance between the anchor and the negative message. Finally, it is used to rank the bus CAN messages and find abnormal messages according to this ranking.

Besides the specific attention paid to cybersecurity, the majority of the current research in Triplet networks focuses mainly on the definition of robust triplet loss functions [31, 66, 74]. [31, 33, 17, 18, 74]. In particular, the comparative study in [31] proves the superiority of the soft-margin loss compared with several competitors. This motivates our decision to consider this loss in the methodology proposed in this paper. Further studies investigate strategies to address the convergence problem. For example, the authors of [64] propose a moderate positive mining approach, to dynamically select the positive pairs of samples, by minimising the intra-class variance. The authors of [66] describe a multi-class N-pair loss that replaces the traditional analysis conducted on a single negative sample per triplet with a joint comparison with several negative samples. Finally, the authors of [78] reduce the bias in the triplet random selection by continuously adapting the selection of triplets to the distribution shift.

2.3. Dealing with imbalanced data

Finally, recent studies of network intrusion detection have investigated various approaches (comprising DML-based) to handle the problem of data imbalance also in combination with the task of attack classification. As already discussed above, DML-based techniques are described in [10, 11], in order to deal with the class imbalance condition in the attack classification problem.

A different approach is investigated in [47], where the class imbalance problem is handled by reworking the near-neighbour subset of an imbalanced set of training network flows. The near-neighbour set collects the highly similar samples that augment the difficulty for the classier to learn the differences between the classes. To overcome this issue, the k-means algorithm is used to reduce the majority samples in the near-neighbour set, while the minority samples are augmented to create a new training set. A deep neural network is finally trained on the new training set, to separate normal samples from attacks and recognise the attack category.

In [51] the authors take advantage of the auto-learning ability of a reinforcement-

learning loop to adapt the well-known data augmentation technique SMOTE, in order to remodel the sample behaviours for better performance. The idea of using data augmentation is explored also outside intrusion detection [69]. In [15] the authors investigate how to use adversarial reinforcement learning to address the training bias associated with an unbalanced network intrusion detection dataset, by outperforming the performance of traditional data augmentation (e.g. SMOTE, ADASYN) techniques.

In [48] the authors define a conditional variational autoencoder architecture that integrates the intrusion labels inside the decoder layers. They test the proposed architecture to perform multi-class classification of intrusion data.

Finally, the authors of [34] tackle the class imbalance problem by introducing an imbalanced data filter and convolutional layers to the typical GAN, thus generating new representative samples for the minority classes (rare attacks).

3. Methodology

In this Section we describe a DML -based network intrusion detection methodology, named RENOIR, that achieves an innovative combination of autoencoders and Triplet networks. The aim is to learn robust intrusion detection models to detect new signs of malicious activity in network traffic. Since the focus of this study is on a binary classification task, all the attack classes are assigned to the same label regardless of the attack type. In addition, RENOIR is formulated to process the flow-based characteristics of the network traffic, which aggregate the information on all the packets in the network traffic. Our decision to focus on the flow-level data is supported by the considerations reported in [71], which highlight that the flow characteristics are more stable than the byte or packet-based characteristics. In fact, it is more practical to process flow characteristics to track long-term changes in network behaviour. Examples of flow characteristics handled by RENOIR are those commonly available with the majority of network devices, e.g. the duration of a flow, the protocol type and the total number of forwarding and backwarding packets transmitted between two hosts.

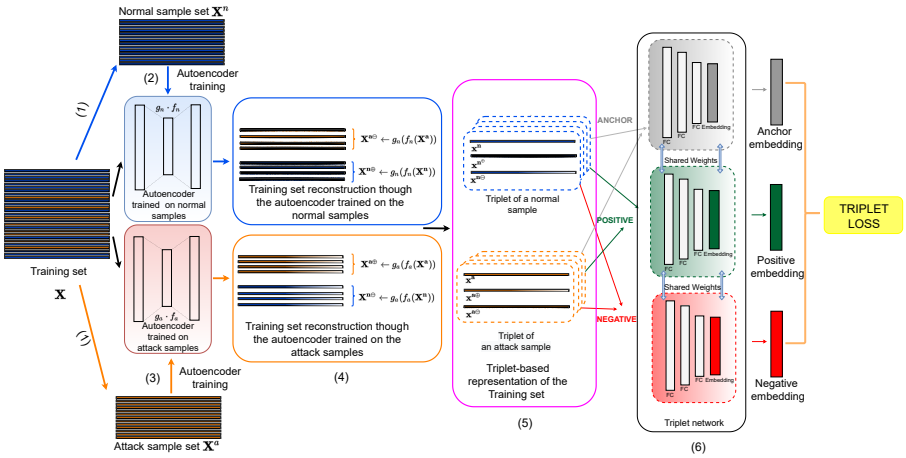


Figure 1: The RENOIR training stage. (1) The input training set $\mathbf{X}$ is partitioned into the normal sample set $\mathbf{X}^n$ and the attack sample set $\mathbf{X}^a$. (2) The normal autoencoder $g_n \cdot f_n$ is trained on $\mathbf{X}^n$, while (3) the attack autoencoder $g_a \cdot f_a$ is trained on $\mathbf{X}^a$. (4) Both $g_n \cdot f_n$ and $g_a \cdot f_a$ are used to restore each training sample $\mathbf{x} \in \mathbf{X}$ (5) and construct the triplets $(\mathbf{x}, \mathbf{x}^{\oplus}, \mathbf{x}^{\ominus})$, so that $\mathbf{x}^{\oplus}$ is the reconstruction of $\mathbf{x}$ restored through the autoencoder in the same class as $\mathbf{x}$, while $\mathbf{x}^{\ominus}$ is the reconstruction of $\mathbf{x}$ restored through the autoencoder in the opposite class of $\mathbf{x}$. (6) These triplets are processed as the input of a Triplet network to learn an embedding of the training set.

The DML methodology fulfilled by RENOIR consists of both a training stage (Figure 1) and a predictive stage (Figure 2). In the training stage, RENOIR analyses a vector-type representation of characteristics of historical single flows to train two class-specific autoencoders. These autoencoders are separately learned on normal and attack flows, respectively. They are injected into a Triplet network, in order to build robust triplets and learn a new embedding input space that separates normal flows better from attacks. In the testing stage, RENOIR uses the trained autoencoders to efficiently predict the class of a new network flow under the umbrella of the embedding learned. The detailed description of the training and predictive stage are reported in Sections 3.1 and 3.2, respectively. The adopted notation is introduced in Table 1. We point out that RENOIR, thanks to the formulation of an appropriate DML strategy based on Triplet networks, defines a network intrusion detection system that is able to take advantage of the knowledge enclosed in both the similarity relationships

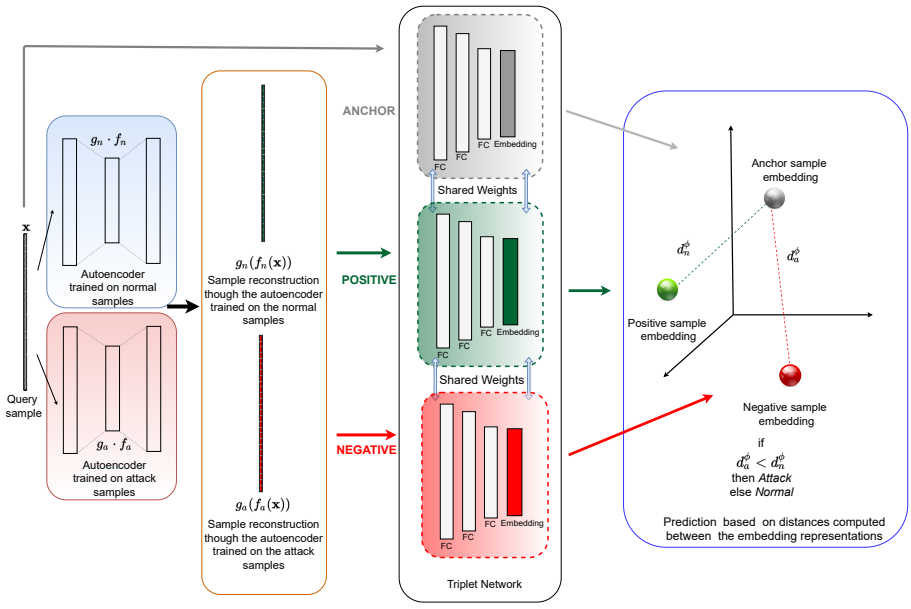


Figure 2: The RENOIR predictive stage. A query sample $\mathbf{x}$ is reconstructed through the autoencoders $g_n \cdot f_n$ and $g_a \cdot f_a$, respectively (left). The distances are computed between $\mathbf{x}$ and $g_n(f_n(\mathbf{x}))$, as well as between $\mathbf{x}$ and $g_a(f_a(\mathbf{x}))$ in the embedding learned through the Triplet network. These distances are processed to predict the class of $\mathbf{x}$.

Table 1: Notation

Symbol	Description
$\mathbf{X}$	training data matrix $\mathbf{X} \subset \mathbb{R}^D$
$\mathbf{X}^n$	subset of samples in $\mathbf{X}$ whose label is normal
$\mathbf{X}^a$	subset of samples in $\mathbf{X}$ whose label is attack
$\mathbf{x}$	triplet anchor sample with $\mathbf{x} \in \mathbf{X}$
$\mathbf{x}^\oplus$	triplet positive counterpart of $\mathbf{x}$
$\mathbf{x}^\ominus$	triplet negative counterpart of $\mathbf{x}$
$g_n \cdot f_n$	normal autoencoder trained on $\mathbf{X}^n$
$g_a \cdot f_a$	attack autoencoder trained on $\mathbf{X}^a$
$g_n(f_n(\mathbf{x}))$	reconstruction of $\mathbf{x}$ through $g_n \cdot f_n$
$g_a(f_a(\mathbf{x}))$	reconstruction of $\mathbf{x}$ through $g_a \cdot f_a$
ϕ: $\mathbb{R}^D \mapsto \mathbb{R}^d$	embedding space learned through a Triplet network

between network flows, belonging to the same class, and the dissimilarity relationships between network flows, belonging to the opposite class. In fact, by accounting for these relationships, it learns a new embedding representation of the input flow-based characteristics, which can improve its ability to separate the attacks from the normal traffic.

3.1. Training stage

The training stage of RENOIR is described in Algorithm 1. During this stage the characteristics of historical network flows are analysed, in order to learn an embedding feature space to separate normal network flows better from attacks. In particular, the training stage of RENOIR consists of three phases:

1. We learn two independent autoencoders to derive a new representation of the training set, which is able to discriminate between *normal* samples and *attack* samples.
2. We use both autoencoders to construct the two unique positive and negative counterparts of each anchor sample and use them to build the *triplet* representation of the training samples.
3. We take advantage of the triplets built in the previous phase to train a Triplet network for the classification task.

3.1.1. Autoencoder training

Let us consider $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ as a set of N training samples, where each $\mathbf{x}_i \in \mathbb{R}^D$ is a row vector corresponding to an input network flow defined over D features, and y_i is the corresponding binary label denoting a *normal* or an *attack* sample. Furthermore, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N \times D}$ denote the data matrix of N D-dimensional random variables $\mathbf{x} \in \mathbb{R}^D$, so that $\mathbf{X}^n = \mathbf{X}_{|y_i=normal}$, resp. $\mathbf{X}^a = \mathbf{X}_{|y_i=attack}$, represent the subset of samples in $\mathbf{X}$, whose label is normal, resp. attack. We process $\mathbf{X}^n$ and $\mathbf{X}^a$ separately to learn two independent autoencoders, denoted as normal autoencoder $g_n \cdot f_n$ and attack autoencoder $g_a \cdot f_a$, respectively. We use both autoencoders to construct the two triplet counterparts of each anchor. In accordance with the theory on autoencoders described in [1] , each autoencoder can be viewed as being composed of two functions: an encoder f—mapping the input vector $\mathbf{x}$ to a hidden representation $\mathbf{h}$ via a deterministic mapping $\mathbf{h} = f(\mathbf{x})$, parameterized by θ_f—and a decoder g—mapping back the resulting hidden representation $\mathbf{h}$ to a reconstructed vector in the input space $\hat{\mathbf{x}} = g(\mathbf{h})$, parameterized by

Algorithm 1: Training stage

input : $\mathcal{D}$: set of training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with $y_i \in \{attack, normal\}$ so that $\mathbf{X}$ denotes the data matrix of N D-dimensional random variables $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{X}^n = \mathbf{X}_{|y_i=normal}$ and $\mathbf{X}^a = \mathbf{X}_{|y_i=attack}$

output: $(g_n \cdot f_n,\ g_a \cdot f_a,\ \phi)$: the learned intrusion detection model

begin

`/* Autoencoder training */`

$g_n \cdot f_n \leftarrow \text{trainAutoencoder}(\mathbf{X}^n)$

$g_a \cdot f_a \leftarrow \text{trainAutoencoder}(\mathbf{X}^a)$

`/* Compute reconstructed vectors using the autoencoders` $g_n \cdot f_n$ `and` $g_n \cdot f_n$ `and use them to create triplets */`

$\mathbf{T} \leftarrow \emptyset$

foreach $(\mathbf{x}, y) \in \mathcal{D}$ **do**

  if $y = normal$ **then**

    $\mathbf{x}^{\oplus} \leftarrow g_n(f_n(\mathbf{x}))$

    $\mathbf{x}^{\ominus} \leftarrow g_a(f_a(\mathbf{x}))$

  else

    $\mathbf{x}^{\oplus} \leftarrow g_a(f_a(\mathbf{x}))$

    $\mathbf{x}^{\ominus} \leftarrow g_n(f_n(\mathbf{x}))$

  `/* Triplet construction */`

  $\mathbf{t} = [\mathbf{x}, \mathbf{x}^{\oplus}, \mathbf{x}^{\ominus}]$

  $\mathbf{T} \leftarrow \mathbf{T} \cup \{\mathbf{t}\}$

`/* Triplet network training */`

$\phi \leftarrow \text{trainTripletNetwork}(\mathbf{T})$

return $g_n \cdot f_n,\ g_a \cdot f_a,\ \phi$

θ_g. Since the activation produced by the top layer in the decoder network corresponds to a reconstructed vector in the same input space, we maintain that this vector can be considered as a new learned feature vector to represent network flows. Specifically, each autoencoder is employed to build a new feature vector $\hat{\mathbf{x}} = g(f(\mathbf{x})) \in \mathbb{R}^D$ that is a reconstruction from a sample $\mathbf{x}$ through the autoencoder.

In principle, the normal autoencoder $g_n \cdot f_n$ should aid in recovering a denoised representation of $\mathbf{X}$, which highlights attacks as anomalies. Based upon this consideration, normal flows can be considered as positive samples, while attacks can be seen as negative samples from $g_n \cdot f_n$. Vice-versa, the attack

autoencoder $g_a \cdot f_a$ aids in recovering a denoised representation of $\mathbf{X}$, which highlights normal flows as anomalies. Therefore, $g_a \cdot f_a$ can see normal flows as negative samples and attacks as positive samples from its point of view.

This conjecture inspires our idea of using the representations of $\mathbf{X}$ reconstructed through both $g_n \cdot f_n$ and $g_a \cdot f_a$ to derive the unique positive and negative component of each triplet.

3.1.2. Triplet construction

According to the theory reported in [32, 61], each triplet $(\mathbf{x}, \mathbf{x}^{\oplus}, \mathbf{x}^{\ominus})$ comprises: (1) $\mathbf{x}$—a training sample selected from $\mathbf{X}$ as the triplet anchor, (2) $\mathbf{x}^{\oplus}$—a positive counterpart of $\mathbf{x}$ and (3) $\mathbf{x}^{\ominus}$—a negative counterpart of $\mathbf{x}$. The traditional implementation of Triplet networks suffers from the convergence problem since it selects $\mathbf{x}^{\oplus}$ and $\mathbf{x}^{\ominus}$ as two "random" samples of $\mathbf{X}$, so that $\mathbf{x}^{\oplus}$ is labelled in the same class as the anchor and $\mathbf{x}^{\ominus}$ is labelled in the opposite class to the anchor. In the second phase of the proposed methodology, we avoid the convergence problem of random sampling by taking advantage of the autoencoder representations of the anchors, restored through $g_n \cdot f_n$ and $g_a \cdot f_a$, respectively.

Specifically, $\mathbf{x}^{\oplus}$ and $\mathbf{x}^{\ominus}$ are built as the unique reconstructions of the anchor $\mathbf{x}$, restored through the autoencoder trained on samples labelled in the same class as $\mathbf{x}$, and the autoencoder trained on samples labelled in the opposite class to $\mathbf{x}$, respectively. For example, let us consider a training sample $\mathbf{x}$ assigned to the label $\mathbf{y} = \textit{normal}$ as the anchor. We build $\mathbf{x}^{\oplus} = g_n(f_n(\mathbf{x}))$ as the positive triplet counterpart of $\mathbf{x}$, and $\mathbf{x}^{\ominus} = g_a(f_a(\mathbf{x}))$ as the negative triplet counterpart of $\mathbf{x}$, respectively. Vice-versa, let us consider a training sample $\mathbf{x}$ assigned to the label $\mathbf{y} = \textit{attack}$ as the anchor. We build $\mathbf{x}^{\oplus} = g_a(f_a(\mathbf{x}))$ as the positive triplet counterpart of $\mathbf{x}$, and $\mathbf{x}^{\ominus} = g_n(f_n(\mathbf{x}))$ as the negative triplet counterpart of $\mathbf{x}$, respectively.

3.1.3. Triplet Network training

Finally, by leveraging the collection of sample triplets constructed from $\mathbf{X}$, we train a Triplet network that embeds every triplet component into a d-

dimensional Euclidean space $\phi\colon \mathbb{R}^D \mapsto \mathbb{R}^d$ learned to better quantify the similarity between sample components within each triplet. This Triplet network processes $\mathbf{x}$, $\mathbf{x}^{\oplus}$ and $\mathbf{x}^{\ominus}$ across three base feed-forward networks with shared parameters. The network learns the embedding $\phi\colon \mathbb{R}^D \mapsto \mathbb{R}^d$ by optimising a triplet loss function. This loss function minimises the distance between the embedding vectors of both the anchor $\mathbf{x}$ and its positive triplet counterpart $\mathbf{x}^{\oplus}$, while it maximises the distance between the anchor $\mathbf{x}$ and its negative triplet counterpart $\mathbf{x}^{\ominus}$. We compute the soft-margin triplet loss proposed in [31] and formulated as follows:

$$\mathcal{L}_{soft} = \sum_{\mathbf{x}\in\mathbf{X}} \ln(1 + exp(\|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\oplus})\|^2 - \|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\ominus})\|^2)), \tag{1}$$

where $\|\mathbf{a}-\mathbf{b}\|$ denotes the Euclidean distance[2] computed between vectors $\mathbf{a}$ and $\mathbf{b}$. We note that the adopted soft-margin triplet loss differs from the traditional triplet loss function introduced in [61], as it is independent of any extra parameter (while the traditional loss metric depends on the margin α that needs to be carefully selected). The experiments illustrated in Section 5.3.4 investigate the effectiveness of the soft-margin triplet loss in the proposed methodology.

3.2. Predictive stage

The predictive stage of RENOIR is described in Algorithm 2. Let us consider a query sample $\mathbf{x} \in \mathbb{R}^D$. First the sample reconstructions $g_n(f_n(\mathbf{x}))$ and $g_a(f_a(\mathbf{x}))$ are restored through both the normal autoencoder $g_n \cdot f_n$ and the attack autoencoder $g_a(f_a(\mathbf{x}))$, respectively. Then the Euclidean distance is computed to compare $\mathbf{x}$ to both $g_n(f_n(\mathbf{x}))$ and $g_a(f_a(\mathbf{x}))$, respectively. The Euclidean distance is computed within the embedding space ϕ, so that:

$$d_n^{\phi}(\mathbf{x}) = \|\phi(\mathbf{x}) - \phi(f_n(g_n(\mathbf{x})))\|^2, \tag{2}$$

$$d_a^{\phi}(\mathbf{x}) = \|\phi(\mathbf{x}) - \phi(f_a(g_a(\mathbf{x})))\|^2. \tag{3}$$

[2] In principle, any distance function can be computed.

Algorithm 2: Predictive stage

input : (1) a new network flow (query sample) $\mathbf{x} \in \mathbb{R}^D$
(2) the intrusion detection model composed of the autoencoders $g_n \cdot f_n$ and $g_a \cdot f_a$, as well as the embedding ϕ

output: the predicted class $\hat{y}$

```
begin
    d_n^φ = EuclideanDistance(φ(x), φ(g_n(f_n(x))))
    d_a^φ = EuclideanDistance(φ(x), φ(g_a(f_a(x))))
    if d_a^φ < d_n^φ then
        ŷ ← attack
    else
        ŷ ← normal
    return ŷ
```

Finally, if $d_a^\phi(\mathbf{x}) < d_n^\phi(\mathbf{x})$, then $\mathbf{x}$ is classified as an attack. Otherwise $\mathbf{x}$ is classified as a normal network flow.

4. Implementation details

RENOIR has been implemented in Python 2.7. The source code is available online.[3] The deep neural network architectures are developed in Keras 2.3[4] – a high-level neural network API with TensorFlow[5] as the back-end. The pre-processing step includes the operation to scale the input numeric features, using the Min-Max scale (as implemented in the Scikit-learn 0.22.2 library[6]). This operation is performed to process features with values in comparable ranges. In addition, for datasets containing categorical attributes the pre-processing includes the implementation of the one-hot-encoder mapping – a transformation commonly used in the intrusion detection literature [46, 40] – to transform input categorical features into numerical features.

For each dataset we conduct an automatic hyper-parameter optimization,

[3] https://github.com/gsndr/RENOIR
[4] https://keras.io/
[5] https://www.tensorflow.org/
[6] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

Table 2: Hyper-parameter search space for both the autoencoders and the Triplet network.

	autoencoders	Triplet network
batch size	$\{2^5, 2^6, 2^7, 2^8, 2^9\}$	$\{2^5, 2^6, 2^7, 2^8, 2^9\}$
learning rate	[0.0001, 0.01]	[0.0001, 0.01]
dropout	[0,1]	[0,1]
#neurons for hidden layer	-	$\{2^7, 2^8, 2^9\}$

using the tree-structured Parzen estimator algorithm, as implemented in the Hyperopt library. [12]. This hyper-parameter optimization is performed by using 20% of the entire training as a validation set, according to the Pareto Principle [52]. In particular, we randomly select the validation set with the stratified sampling procedure [58]. So, in RENOIR, we automatically choose the configuration of the parameters, which achieves the best validation loss. The values of the hyper-parameters, automatically explored with the tree-structured Parzen estimator, are reported in Table 2.

Each autoencoder architecture comprises 3 fully-connected (FC) layers of $32 \times 16 \times 32$ neurons and one dropout layer, in order to prevent the overfitting phenomenon. The mean squared error (*mse*) is used as the loss function. The classical rectified linear unit (*ReLu*) [25] is selected as the activation function for each hidden layer, while for the last layer the *Linear* activation function is used.

The Triplet network is implemented with three base feed-forward networks with shared weights. Each base network is a deep neural network with three intermediate layers (with the number of neurons chosen with the hyper-parameter optimization), an embedding layer with 512 neurons [24] [57, 24] and two dropout layers. The soft-margin triplet loss as described in Equation 1 is used as the loss function. The (*ReLu*) function is selected as the activation function for each hidden layer, while for the embedding layer the *Sigmoid* activation function [28] is used. The *Sigmoid* activation function is commonly used for the embedding layer [53] instead of a *Linear*, in order to guarantee that each dimension will be between 0 and 1. This architecture assigns samples to normal or attack classes by returning the Euclidean distance returned by the embedding layers. The

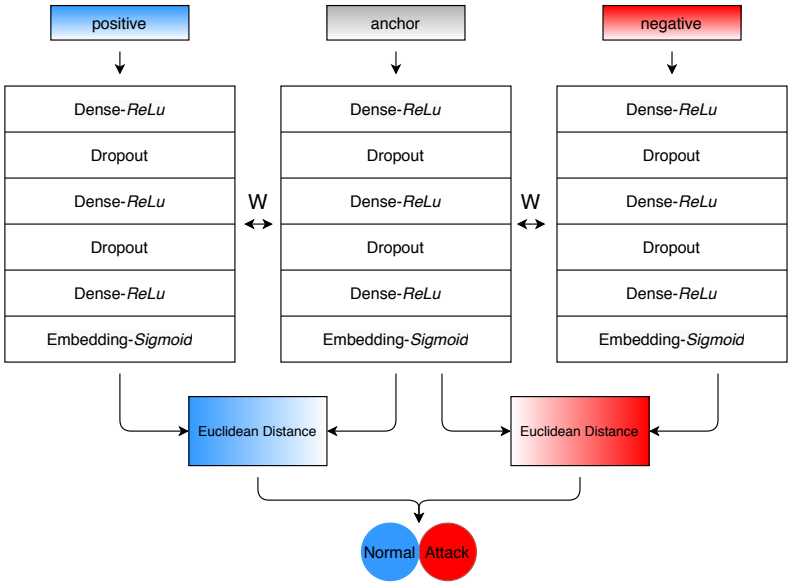


Figure 3: The Triplet network of RENOIR. The architecture comprises three base feed-forward networks with shared weights (W).

details of the configuration of this architecture are reported in Figure 3.

5. Empirical evaluation

For the empirical evaluation, we consider three benchmark datasets (see Section 5.1) to evaluate the effectiveness of the network intrusion detection methodology implemented by RENOIR. Each dataset includes both a labelled training set – processed to learn the network intrusion detection model – and a testing set – considered to evaluate the network intrusion detection ability of the trained model. In particular, the performance of RENOIR is measured in terms of accuracy and efficiency (see Section 5.2).

Table 3: Dataset description. For each dataset we collect: the number of attributes, the total number of network flow samples collected in the dataset, the number of normal network flows (and their percentage of the total size), as well as the number of attacking flows (and their percentage of the total size).

		Dataset		
		KDDCUP99	**AAGM17**	**CICIDS17**
Attributes	Total	**42**	**80**	**79**
	Binary	6	3	18
	Categorical	3	-	-
	Numerical	32	76	60
	Class	1	1	1
Training set	Total	**494021**	**100000**	**100000**
	Normal flows	97278 (19.7%)	80000 (80%)	80000 (80%)
	Attacking flows	396743 (80.3%)	20000 (20%)	20000 (20%)
Testing set	Total	**311029**	**100000**	**900000**
	Normal flows	60593 (19.5%)	80000 (80%)	720000 (80%)
	Attacking flows	250436 (80.5%)	20000 (20%)	180000 (20%)

5.1. Dataset description and experimental methodology

We consider three benchmark intrusion detection datasets, that is, KDDCUP99,[7], AAGM17[8] and CICIDS17[9].

KDDCUP99 was introduced in the KDD Tools Competition in 1999. The dataset comprises four categories of attacks: Denial of Service Attack (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L) and Probing Attack. The dataset contains network flows simulated in a military network environment and recorded as vectors of 42 flow-level characteristics. The original dataset comprises a training set of 4.898.431 samples and a testing set of 311.027 samples. To keep the cost of the learning stage under control, the original dataset comprises a reduced training set, denoted as 10%KDDCUP99Train, that contains 10% of the training data taken from the original dataset. In addition, the entire dataset is imbalanced in both the training and testing set, where the percentage of attacks is higher than that of normal flows (80.3% vs 19.7% in the training set and 80.5% vs 19.5% in the testing set). In this study we consider

[7] `http://kdd.ics.uci.edu//databases//kddcup99//kddcup99.html`
[8] `https://www.unb.ca/cic/datasets/android-adware.html`
[9] `https://www.unb.ca/cic/datasets/ids-2017.html`

10%KDDCUP99Train for the learning stage, while we use the entire testing set, denoted as KDDCUP99Test, for the evaluation stage.[10] This experimental scenario, with both 10%KDDCUP99Train and KDDCUP99Test, is commonly used for the evaluation of network intrusion detection systems also in recent studies (e.g. [73, 59]).

AAGM was collected by the Canadian Institute for Cybersecurity in 2017. This dataset contains the network traffic captured from Android applications––both malware and benign –obtained by installing Android apps on real smartphones in a semi-automated manner [42]. After running the apps on the real Android smartphones (NEXUS 5), the generated traffic was captured and transformed into samples labelled in two classes (attack and normal) using CICFlowMeter. Every network flow sample is spanned over 80 attributes. Specifically, attacking samples represent the malicious traffic generated by some popular adware families (Airpush, Dowgin, Kemoge, Mobidash and Shuanet) and malware families (AVpass, FakeAV, FakeFlash/FakePlayer, GGtracker and Penetho). In this study, we use the subset of the entire network traffic log that was initially constructed by [5]. Specifically, this subset comprises a training set with 100K samples and a testing set with 100k samples. Both training and testing samples are randomly selected from the entire log using the stratified random sampling. This dataset is imbalanced in both the learning and the evaluation stage. In fact, the number of normal network flows is significantly higher than the number of attacks (80% vs 20%). This resembles the common set-up of an anomaly detection learning task that often occurs in a network.

Finally, CICIDS2017 was collected by the Canadian Institute for Cybersecurity in 2017. The original dataset is a 5-day log collected from Monday July 3, 2017 to Friday July 7, 2017 [62]. The first day (Monday) contains only benign traffic, while the other days contain various types of attacks, in addition to normal network flows. Every network flow sample is spanned over 79 at-

[10] 10%KDDCUP99Train and KDDCUP99Test are populated with the data stored in kddcup.data_10_percent.gz and corrected.gz at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

tributes. The dataset is commonly used in the evaluation of anomaly detection methodologies with the training performed on the first day [82, 8]. However, a few recent studies consider these data also in the evaluation of classification methodologies, as we do in this paper [46, 40, 59, 3]. In our experimental study, we consider the training and testing sets of CICIDS2017, built according to the strategy described by [46, 40]. The dataset comprises six attack families: Brute Force Attack, Heartbleed Attack, BotNet, DoS/DDoS Attack, Web Attack and Infiltration Attack. Specifically, we build one training set with 100K samples, and one testing set with 900K samples. Both training and testing samples are randomly selected from the entire 5-day log using the stratified random sampling procedure, in order to select 80% of normal flows and 20% of attacks, as in the original log.

A summary of the characteristics of the datasets described above is reported in Table 3. We note that the traffic is imbalanced in all the datasets. In both AAGM17 and CICIDS2017, the number of normal network flows is significantly higher than the number of attacks. On the contrary, in KDDCUP99 the number of attacks is higher than the number of normal flows.

5.2. Evaluation metrics

The overall accuracy performance of the proposed methodology is measured by analysing the F1-score of the intrusion detection models learned. This is the harmonic mean of Precision and Recall, where Precision measures the ability of an intrusion detection system to identify only the attacks, while Recall can be thought of as the system's ability to find all the attacks. The higher the F1-score, the better the balance between Precision and Recall achieved by the algorithm. On the contrary, the F1-score is not so high when one measure is improved at the expense of the other. In addition, we consider Accuracy (that is measured in the evaluation of various competitors). This is the ratio of flows correctly labelled on all the flows tested. The mathematical formulation of these accuracy metrics is reported in Table 4.

The efficiency performance is evaluated with the computation time spent

Table 4: Evaluation metrics: Accuracy, Precision, Recall and F1-score. These metrics are computed by accounting for the number of true positive—TP (number of attacks correctly detected), the number of true negative—TN (number of normal samples correctly detected), the number of false positive—FP (number of normal samples incorrectly detected as attacks) and the number of false negative—FN (number of normal samples incorrectly detected as attacks).

Metric	Mathematical formulation
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1-score	$2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$

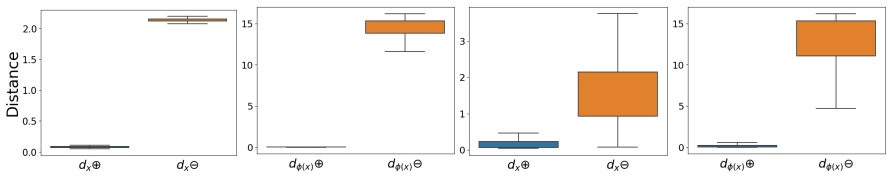
training the intrusion detection model and the average time spent processing every testing sample. They are collected on a Linux machine with an Intel(R) Core(TM) i7-9700F CPU @ 3.00GHz and 32GB RAM. All the experiments have been executed on a single GeForce RTX 2080. The training TIME is measured in minutes, while the testing TIME is measured in milliseconds.

5.3. Results

The performance of RENOIR is measured by reporting the distances between the triplet components, the Accuracy and the F1-score of the intrusion detection models, as well as the efficiency of both the training stage and the predictive stage. For the hyper-parameter optimisation of RENOIR, a validation set is used during the training of the model (20% of the entire training set is randomly selected for the validation with stratified sampling, as described in Section 4).

5.3.1. Embedding analysis

We start by investigating how the embedding learned during the training stage of RENOIR can actually disclose knowledge that contributes to separating attacking flows from normal ones. To this aim, we explore how the embedding space moves each sample closer to its positive reconstruction (recovered through



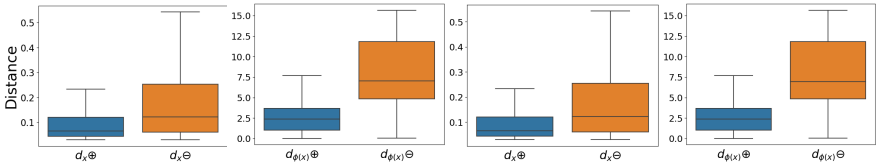
(a) Training (b) Training + ϕ (c) Testing (d) Testing + ϕ

Figure 4: Embedding analysis (KDDCUP99 dataset): Euclidean distances computed between: (1) each sample and its positive triplet counterpart ($d\oplus$ – left box) and (2) each sample and its negative triplet counterpart ($d\ominus$ – right box). The distances are computed both in the original input feature space ($d_{\mathbf{x}}\oplus$ and $d_{\mathbf{x}}\ominus$) and in the embedded feature space ($d_{\phi(\mathbf{x})}\oplus$ and $d_{\phi(\mathbf{x})}\ominus$), respectively. Figures 4a and 4b show the box plots of the distances of the training samples, while Figures 4c and 4d show the box plots of the distances of the testing samples.

the autoencoder labelled with the same class as the anchor sample), while moving each sample away from its negative reconstruction (recovered through the autoencoder labelled with the opposite class to the anchor sample). This is done by augmenting the gap between opposite distances and consequently separating normal samples better from attacks.

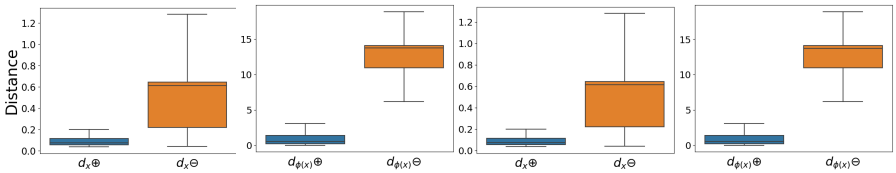
Figures 4, 5 and 6 show the box plots of the Euclidean distances computed as $d_{\mathbf{x}}\oplus = \|\mathbf{x} - \mathbf{x}^{\oplus}\|^2$ and $d_{\mathbf{x}}\ominus = \|\mathbf{x} - \mathbf{x}^{\ominus}\|^2$ by ignoring the embedding ϕ, as well as the Euclidean distances computed as $d_{\phi(\mathbf{x})}\oplus = \|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\oplus})\|^2$ and $d_{\phi(\mathbf{x})}\ominus = \|\phi\mathbf{x} - \phi(\mathbf{x}^{\ominus})\|^2$ by considering ϕ. The plots are constructed in both the training set and the testing set of each dataset.

These plots show that distances between samples and their negative reconstructions are greater than distances between samples and their positive reconstructions, independently of the use of embedding. However, computing distances in the embedding space can actually increase the gap between opposite distances, by moving each sample closer to the autoencoder of its class and away from the autoencoder of the opposite class. This behaviour, that was expected in the training samples (as shown in Figures 4b, 5b and 6b), was also evident in the testing samples (as shown in Figures 4d, 5d and 6d), although the testing samples comprise network flows which were unseen at training time.



(a) Training (b) Training+ϕ (c) Testing (d) Testing+ϕ

Figure 5: Embedding analysis (AAGM17 dataset): Euclidean distances computed between: (1) each sample and its positive triplet counterpart ($d\oplus$ – left box) and (2) each sample and its negative triplet counterpart ($d\ominus$ – right box). The distances are computed both in the original input feature space ($d_{\mathbf{x}}\oplus$ and $d_{\mathbf{x}}\ominus$) and in the embedded feature space ($d_{\phi(\mathbf{x})}\oplus$ and $d_{\phi(\mathbf{x})}\ominus$), respectively. Figures 5a and 5b show the box plots of the distances of the training samples, while Figures 5c and 5d show the box plots of the distances of the testing samples.



(a) Training (b) Training + ϕ (c) Testing (d) Testing + ϕ

Figure 6: Embedding analysis (CICIDS17 dataset): Euclidean distances computed between: (1) each sample and its positive triplet counterpart ($d\oplus$ – left box) and (2) each sample and its negative triplet counterpart ($d\ominus$ – right box). The distances are computed both in the original input feature space ($d_{\mathbf{x}}\oplus$ and $d_{\mathbf{x}}\ominus$) and in the embedded feature space ($d_{\phi(\mathbf{x})}\oplus$ and $d_{\phi(\mathbf{x})}\ominus$), respectively. Figures 6a and 6b show the box plots of the distances of the training samples, while Figures 6c and 6d show the box plots of the distances of the testing samples.

5.3.2. Ablation study

We proceed with the analysis by performing an ablation study. This study aims at investigating:

- How the proposed methodology can leverage information synthesized through autoencoders in the training stage, in order to learn an embedding that aids in separating normal network flows better from attacks, without suffering from the convergence problem.

- How the proposed methodology can take advantage of coupling the embedding to the autoencoders in the predictive stage, in order to replace the kNN-based classification adopted in traditional DML classifiers (e.g. [36, 53, 86, 78]), [36, 53, 78]), by speeding up the analysis of each new

network flow and gaining accuracy in predicting the class of the flow.

To this aim, we consider four architecture configurations as baselines. These are in turn defined by removing the autoencoder information or the embedding from the training stage and/or the predictive stage of RENOIR. In particular, we consider the following baseline architectures:

- T+kNN—which gives away autoencoder information in both the training and the predictive stage. In the training stage it learns the embedding by training a traditional Triplet network on triplets populated with positive and negative samples, that are randomly sampled in the training set. In the predictive stage, it follows the traditional implementation of DML classifiers and assigns every new query sample to the class predicted by performing the kNN algorithm on the embedding of the training set $\phi(\mathbf{X})$.

- A+A—which gives away the embedding. In the training stage it only learns two autoencoders on the normal training samples and the training attack samples, separately. In the predictive stage it assigns every new query sample to the class associated with the autoencoder that restores the closest reconstruction of the sample.

- AT+kNN—which gives away autoencoder information in the predictive stage. In the training stage it replays RENOIR. In the predictive stage it assigns every new query sample to the class predicted by performing the kNN algorithm on the embedding of the training set $\phi(\mathbf{X})$.

- AT+AkNN— which considers autoencoder information and embedding, but the autoencoders are coupled to kNN in the predictive stage. In the training stage it replays RENOIR. In the predictive stage it assigns every new query sample to the class predicted by performing the kNN algorithm on the training space $\phi(\mathbf{X}) \bullet \phi(\mathbf{X}^{\oplus}) \bullet \phi(\mathbf{X}^{\ominus})$ constructed through the concatenation operator.

Similarly to RENOIR, all the configurations that train a Triplet network (T+kNN, AT+kNN and AT+AkNN) implement the soft-margin loss function.

The predictive stage of the configurations that use the kNN algorithm (T+kNN, AT+kNN, AT+AKNN) are run with k ranging between 1 and 30. As T+kNN uses random sampling for the triplet construction, we repeat the training and predictive stages of this configuration on five trials. In each dataset the best trial of T+kNN (in terms of accuracy performance achieved on the testing set) is selected for the comparative analysis.

The performance of the compared configurations is measured in terms of Accuracy, F1-score, TIME spent completing the training stage (Training TIME), as well as TIME spent, on average, predicting the class of each testing sample (Testing TIME). We point out that the intrusion detection system focuses mainly on the discovery of signs of malicious activity in the traffic, so the label "attack" is handled as the positive class of the binary classification for the computation of F1-score. In any case, for completeness, we also report the recall analysis for both categories of the experiment Recall(Attack) and Recall(Normal).

The overall results, reported in Table 5, show that RENOIR is commonly more accurate than all its baselines. The only exception is observed with AT+kNN on CICIDS17, where RENOIR is the runner-up in terms of Accuracy and F1-score. In any case, the recall analysis on CICIDS17 reveals that RENOIR is the best method in terms of intrusion detection ability, while it is the runner-up in terms of normal flow detection ability. On the other hand, AT+kNN is the runner-up in terms of Accuracy and F1-score on both KDDCUP99 and AAGM17. In general, RENOIR and AT+kNN achieve close accuracy performances in all the datasets, although the predictive stage of AT+kNN is always significantly slower than the predictive stage of RENOIR. This behaviour is not surprising, since AT+kNN replays the training stage of RENOIR, while it uses the computation-demanding kNN algorithm in the predictive stage. Additional general considerations concerning the recall analysis can be formulated. In fact, we note that RENOIR is always the best method in terms of sample detection ability in one class (the normal class in KDDCUP99 and AAGM17 and the attack class in CICIDS17) and the runner-up in the sample detection ability in the opposite class.

In general, this analysis confirms the viability of our idea of injecting autoencoder information into both the training stage and the predictive stage of the proposed DML methodology. Interestingly, basing predictions on autoencoders always speeds up the prediction stage (i.e. the time that both RENOIR and A+A spend on average yielding a prediction is significantly less than the prediction time spent by T+kNN, AT+kNN and AT+kNN), while performing autoencoder-based classification in the embedding space allows RENOIR to achieve higher accuracy levels.

Additional considerations concern the time spent completing the training stage. In fact, the highest performance that RENOIR achieves in the predictive stage is at the cost of the highest complexity of the training stage. In particular, coupling the computation of the autoencoder information to the training of the Triplet network adds significant complexity to the training stage (i.e. the time that RENOIR, AT+kNN and AT+AkNN spend on average completing the training stage is greater than the training time spent by both T+kNN and A+A).

Finally, we explore the sensitivity of the accuracy performance of RENOIR and its baseline architectures by diminishing the amount of training samples processed. To this aim, we consider four trials of AAGM17 with 100% (baseline), 75%, 50% and 25% training samples, respectively. The training samples are extracted from the original training set (AAGM17Train) in a stratified manner. The F1-score of RENOIR, T+kNN, A+A, AT+kNN and AT+AkNN, collected by diminishing the number of training samples and measured on the testing set (AAGM17Test), is plotted in Figure 7. Diminishing the number of training samples leads to a decrease in the F1-score of all the compared algorithms.

However, RENOIR continues outperforming its baselines independently of the training set size. In any case, we note that the difference between RENOIR and AT+kNN becomes negligible when decreasing the amount of training samples. We recall that RENOIR and AT+kNN perform the same training stage, but produce a different predictive stage. Figure 8 compares the Testing TIME of both RENOIR and AT+kNN. The Testing TIME of RENOIR does not change with the training set size, since RENOIR decides the class of a testing sample

Table 5: Ablation study: Overall Accuracy, F1-score, Recall(Attack), Recall(Normal) and TIME measured on KDDCUP99Test, AAGM17Test and CICIDS2017Test for RENOIR, T+kNN, A+A, AT+kNN and AT+AkNN. Training TIME is the time spent in minutes completing the training stage. Testing TIME is the time spent, on average, in milliseconds computing the prediction of each testing sample during the predictive stage. If kNN is used in the prediction stage, bestK denotes the value of k which achieves the highest accuracy performance on the testing set. The best results are in bold.

Dataset		Architecture				
		RENOIR	T+kNN	A+A	AT+kNN	AT+AkNN
KDDCUP99Test	Accuracy	**93.50**	82.85	91.41	93.25	89.86
	F1-score	**95.80**	88.75	94.39	95.63	93.94
	Recall(Attack)	92.05	85.49	89.68	91.77	**97.54**
	Recall(Normal)	**99.40**	70.41	98.56	99.36	58.12
	Training TIME	95.04	53.70	**41.37**	95.04	95.04
	Testing TIME	0.013	17.49	**0.002**	3.99	8.00
	bestK	-	26	-	2	14
AAGM17Test	Accuracy	**89.63**	67.59	65.63	88.63	55.73
	F1-score	**71.90**	31.76	47.56	68.82	23.29
	Recall(Attack)	66.40	37.71	**77.94**	62.72	33.60
	Recall(Normal)	**95.43**	75.05	66.55	95.11	61.26
	Training TIME	28.57	22.48	**6.20**	28.57	28.57
	Testing TIME	0.04	9.87	**0.078**	6.38	6.56
	bestK	-	2	-	6	2
CICIDS17Test	Accuracy	98.24	58.63	95.25	**98.30**	57.69
	F1-score	95.70	27.33	88.43	**95.78**	27.52
	Recall(Attack)	**97.28**	38.90	90.80	97.01	40.16
	Recall(Normal)	98.48	63.56	96.36	**98.61**	62.08
	Training TIME	26.74	21.60	**5.09**	26.74	26.74
	Testing TIME	0.01	14.21	**0.001**	14.80	11.88
	bestK	-	2	-	1	2

based on the distance between the sample and its reconstructions, restored with the autoencoders previously trained from the normal training samples and the attack training samples, respectively. On the other hand, the Testing TIME of AT+kNN decreases as the training set size decreases, since AT+kNN decides the class of a testing sample by performing a k-nearest neighbour search in the training space. In any case, the predictive stage of RENOIR is still more efficient than the predictive stage of AT+kNN even when decreasing the size of the training set.

5.3.3. Convergence discussion

After completing the ablation study, we explore in depth how the triplet convergence problem can jeopardize the performance of a traditional Triplet network. To this aim, we examine closely the performance of T+kNN, as this is the baseline configuration (described in Section 5.3.2) that, following the tradi-

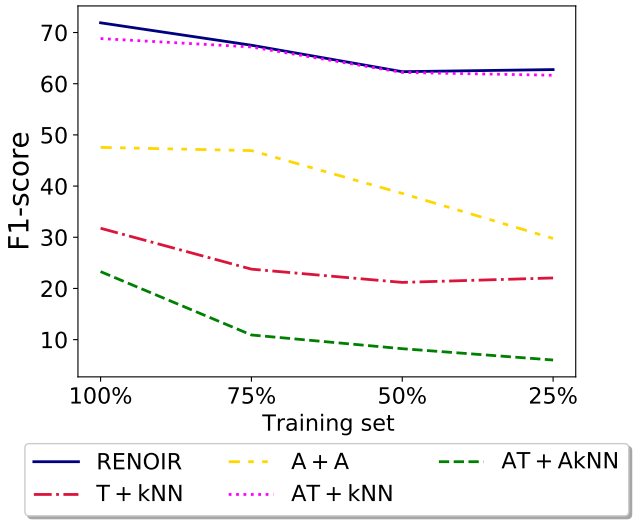


Figure 7: F1-score of RENOIR, T+kNN, A+A, AT+kNN and AT+AkNN on AAGM17Test by varying the size of the training set.

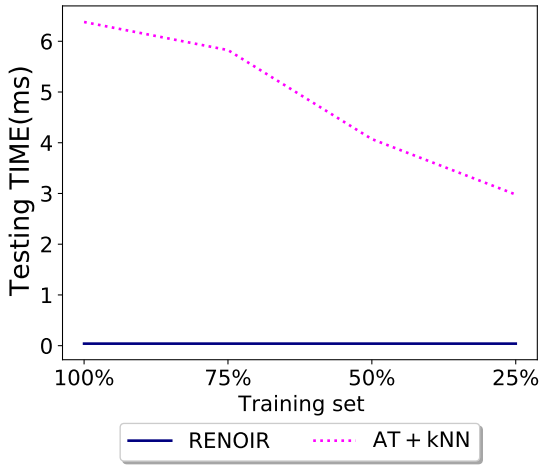


Figure 8: Testing TIME spent, on average, in milliseconds computing the prediction of each testing sample during the predictive stage of RENOIR and AT+kNN on AAGM17Test by varying the size of the training set.

Table 6: Convergence problem analysis: accuracy of T+kNN on five trials of the training stage.

Dataset	bestK	Accuracy	F1-score	Recall(Attack)	Recall(Normal)
KDDCUP99Test	26	82.18	88.60	85.94	66.63
	14	57.60	68.60	57.40	58.94
	29	62.28	72.14	60.65	69.05
	26	71.78	80.56	72.62	68.29
	26	**82.85**	**88.75**	**85.49**	**70.41**
AAGMTest	2	64.85	28.53	35.08	72.29
	2	64.54	28.55	35.44	71.81
	2	61.91	27.13	35.46	68.53
	2	58.65	24.89	34.26	64.75
	2	**67.59**	**31.76**	**37.71**	**75.05**
CICIDS2017Test	2	58.35	26.26	37.08	63.67
	2	58.40	25.75	36.07	63.98
	2	58.39	26.11	36.75	63.79
	2	58.11	24.89	34.69	63.97
	2	**58.63**	**27.33**	**38.90**	**63.56**

tional Triplet network theory, uses the random sampling in the triplet construction. Table 6 reports the accuracy results collected with T+kNN by repeating the training and predictive stages on five distinct trials for all the datasets. Since the random sampling is used in the triplet construction, the Triplet loss is learned with a triplet training set that potentially changes at each trial execution. Changes in the triple training set may reasonably cause the variability of the performance that we observe on the various trials. Based upon these considerations, we draw the conclusion that the variability of the results of T+kNN highlights the effect of the convergence problem, since the accuracy of T+kNN actually depends on how training samples are randomly sampled for the triplet construction. This provides evidence of the weakness of basing the triplet construction on traditional random sampling, while the ablation study has empirically proved the viability of autoencoder information as an effective means to handle this issue. In our proposal, once the normal and attack autoencoders of a specific training set are trained, they provide the information for the "unique" way to construct the triplets of the study training set. In fact, there is one only reconstruction of the same anchor through each specific autoencoder. This definitely avoids the convergence problem of the random sampling.

Table 7: Margin triplet loss and soft-margin triplet loss.

Triplet loss	Mathematical formulation
Margin triplet loss [61]	$\sum_{\mathbf{x}\in\mathbf{X}} max(\|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\oplus})\|^2 - \|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\ominus})\|^2 + \alpha, 0)$
Soft-margin triplet loss [31]	$\sum_{\mathbf{x}\in\mathbf{X}} \ln(1 + exp(\|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\oplus})\|^2 - \|\phi(\mathbf{x}) - \phi(\mathbf{x}^{\ominus})\|^2))$

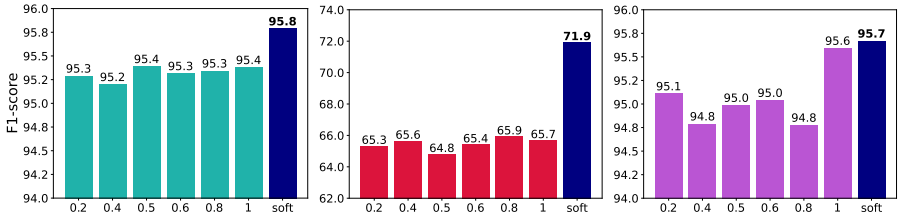
5.3.4. Triplet loss analysis

We analyse the accuracy of the proposed methodology along the margin triplet loss adopted. To this aim, we compare the performance of the soft-margin triplet loss, that we have selected for the implementation in RENOIR (as described in Section 3.1) to the performance of the traditional margin triple loss. The mathematical formulation of both loss functions is reported in Table 7. We point out that both loss functions combine distances computed between samples within triplets. However, the performance of the traditional margin triple loss may depend on a user-defined margin threshold α. In this study the two loss functions are computed with the positive and negative samples of a triplet, built by resorting to autoencoders (as described in Section 3.1). The experiments are performed by varying α of the margin triplet loss among 0.2, 0.4, 0.5, 0.6, 0.8 and 1.

The F1-score, reported in Figure 9 for all the datasets, shows that the overall conclusions drawn by [31] are still valid even when the Triplet network implementation is enhanced with autoencoders, as presented in this paper. In particular, the accuracy of the traditional margin triplet loss may vary with α (by achieving the highest F1-score with $\alpha = 0.5$ in KDDCUP99, $\alpha = 0.8$ in AAGM17 and $\alpha = 1$ in CICIDS17). In any case, in all the datasets, computing the soft-margin triplet loss in the training stage allows us to achieve the highest F1-score in the predictive stage, without requiring the set-up of any margin parameter.

5.3.5. Competitor analysis

Finally, we assess the significance and novelty of RENOIR with respect to several competitors, selected from the recent state-of-the-art literature on net-



(a) KDDCUP99Test (b) AAGM17Test (c) CICIDS17Test

Figure 9: Margin triplet loss analysis: F1-score of Triplet networks trained with the traditional margin triplet loss computed by varying the margin among 0.2, 0.4, 0.5, 0.6, 0.8 and 1, as well as the soft-margin triplet loss.

work intrusion detection. In particular, we consider the following competitors that perform binary intrusion detection (like RENOIR) and are based on various categories of deep learning architectures:

- Convolutional Neural Network-based competitors (CNN): CNN4 [30], GreyscaleCNN [46], [40] and RGBCNN [40];
- Long Short-Term Memory Neural Network-based competitor (LSTM): BLSTM [21];
- Recurrent Neural Network-based competitor (RNN): BRNN [21];
- Deep Neural Network-based competitors (DNN): DNN 4 Layers [73], DNN3 [72], DNN4 [30], DBN [45] , A+DBN [45], WnD [76] and MLP [76];
- GAN Network-based competitors (GAN): AnoGAN [60] [80], Efficient GAN [79], ALAD [80] and MAD-GAN [19];
- Autoencoder-based competitors: SAE [76], AIDA [6], MINDFUL [7] and THEODORA [5].

The codes of AIDA [6],[11] MINDFUL [7][12] and THEODORA[13] are publicly available to repeat the experiments reported in this study. The results of the

[11] https://github.com/gsndr/AIDA
[12] https://github.com/gsndr/MINDFUL
[13] https://github.com/gsndr/THEODORA

Table 8: Competitor analysis: Accuracy and F1-score measured on the testing sets of KDD-CUP99, AAGM17 and CICIDS17. The results of competitors are collected from the reference papers. The best results are in bold. "-" denotes that no value is reported in the reference paper.

Dataset	Category	Description	Accuracy	F1-score
KDDCUP99	RENOIR	Autoencoder+DML	**93.50**	**95.80**
	MINDFUL [7]	Autoencoder+CNN	92.49	95.13
	CNN4 [30]	CNN	92.47	-
	DNN4Layers [73]	DNN+Text-based encoding	93.00	95.50
	DNN-3 [72]	DNN	93.00	95.50
	DNN4 [30]	DNN	92.88	-
	DBN [45]	DNN	91.40	-
	A+DBN [45]	Autoencoder+DBN	92.10	-
	BLSTM [21]	LSTM	-	93.27
	BRNN [21]	RNN	-	91.82
	THEODORA [5]	Autoencoder+CNN	92.97	95.46
	AIDA [6]	Autoencoder+MLP	92.36	95.04
	AnoGAN [60] [80]	GAN	-	88.65
	Efficient GAN [79]	GAN		93.72
	ALAD [80]	GAN	-	95.01
	MAD-GAN [19]	GAN	-	90.00
AAGM17	RENOIR	Autoencoder+DML	**89.63**	**71.90**
	MINDFUL [7]	Autoencoder+CNN	86.15	51.62
	THEODORA [5]	Autoencoder+CNN	87.62	65.92
	AIDA [6]	Autoencoder+MLP	86.06	57.78
CICIDS17	RENOIR	Autoencoder+DML	**98.24**	**95.70**
	MINDFUL [7]	Autoencoder+CNN	97.90	94.93
	Grey-scaleCNN [46][40]	CNN	-	82.00
	RGBCNN [40]	CNN	-	89.00
	THEODORA [5]	Autoencoder+CNN	98.03	95.25
	AIDA [6]	Autoencoder + MLP	94.50	85.80

remaining competitors are taken from the reference papers, as their code is not publicly available to repeat the experiments. However, the compared algorithms, except for Grey-scaleCNN and RGBCNN, have been evaluated by their authors on the same training and testing sets described in Section 5.1. This makes this comparative study reliable. Additionally, we note that both Grey-scaleCNN and RGBCNN were originally evaluated in [46, 40] by considering a training set and a testing set of CICIDS17, which are not publicly available. In any case, the version of the dataset CICIDS17, that is described in Section 5.1 and that we use in this study, reproduces the training and testing ratios reported in [46, 40]. Finally, similarly to [46, 40], we have also used the stratified sampling strategy to select the samples of the original log that populate the training set and the testing set of CICIDS17. This allows us to reproduce experimental conditions that are similar to (although not the same as) the conditions of the evaluation described in [46, 40].

We point out that both GAN-based and autoencoder-based competitors are the nearest related to RENOIR. The GAN-based competitors [60, 80, 79, 80, 19] deal with the class imbalance by mainly using generative adversarial learning for

anomaly detection (i.e. to detect the samples of the minority class as anomalies). On the other hand, the autoencoder-based competitors take advantage of autoencoder information. In particular, THEODORA [5] learns autoencoder information through a multi-channel CNN and uses a label re-assignment mechanism to deal with rare samples.

For all the methods in this comparative study we collect the Accuracy and F1-score, as these metrics are commonly provided in the reference studies. The results, reported in Table 8 for all the datasets, show that RENOIR outperforms all its competitors, including the GAN-based competitors (evaluated on KDDCUP99 in the reference studies) and THEODORA (evaluated in all the datasets).

Finally, we complete this evaluation by comparing the performance of RENOIR to the performance of Vec2im-SIAM [55] — a DML-based network intrusion detection methodology that has been recently formulated for the binary classification of network flows [55]. In [55] the authors evaluate the Accuracy of Vec2im-SIAM using the dataset NSL-KDD.[14] This dataset is introduced in [70] as a revised version of KDDCUP99, that is obtained by removing the duplicate samples from KDDCUP99. Although the code of Vec2im-SIAM is not publicly available, we reproduce the same experimental settings described in [55], since it used KDDTrain$^+$_20Percent as the training set and KDDTest$^+$ as the testing set. Both datasets are publicly available. In particular, KDDTrain$^+$_20Percent is a 20% subset of the NSL-KDD training set, which comprises 25192 network flows (13449 normal flows and 11743 attacks). KDDTest$^+$ comprises 22544 network flows (9711 normal flows and 12833 attacks). The Accuracy of both RENOIR and Vec2im-SIAM is plotted in Figure 10. We note that RENOIR achieves competitive performance also in the DML literature by outperforming Vec2im-SIAM. Additional results comparing the performance of RENOIR to that of DML-based intrusion detection competitors are collected in the multi-class scenario and discussed in Section 6.

[14] https://www.unb.ca/cic/datasets/nsl.html

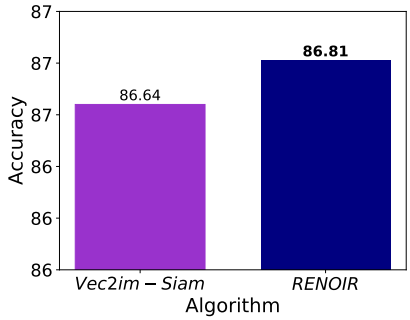


Figure 10: Accuracy of Vec2im-SIAM and RENOIR on KDDTest$^+$. The Accuracy of Vec2im-SIAM is collected from the reference paper [55].

Table 9: Number of samples per category in both KDDTrain$^+$ and KDDTes$^+$.

Dataset	**Total**	**DoS**	**Probe**	**R2L**	**U2R**	**Normal**
KDDTrain$^+$	125973	45927 (37%)	11656 (9.11%)	995 (0.85%)	52 (0.04%)	6734 (53%)
KDDTest$^+$	22544	7458 (33%)	2421 (11%)	2754 (12.1%)	200 (0.9%)	9711 (43%)

6. Multi-class scenario extension

We analyse the effectiveness of the proposed methodology also in a multi-class scenario of network intrusion detection. To this purpose, we re-consider the dataset NSL-KDD, but in the multi-class version. The multi-class NSL-KDD comprises normal flows and four categories of attack: Denial of Service Attack (DoS), User to Root Attack (U2R), Remote to Local Attack (R2L) and Probing Attack. For this experiment we adopt the original data setting described in [70], that includes KDDTrain$^+$ as the training set (without down-sampling) and KDDTest$^+$ as the testing set. The number of samples collected in both the training and testing sets for each category is reported in Table 9. We note that both U2R and R2L are rare attacks. We select this dataset as it has been recently used in the evaluation of various multi-class intrusion detection methods.

In order to use RENOIR in the multi-class scenario, we explore the performance of four configurations formulated according to various multi-class com-

bination strategies.

- OVA—We use the One-Versus-All (OVA) combination strategy [49]. Specifically, we perform the binary training stage of RENOIR on five trials—one trial for each category in the multi-class dataset. In each trial, the entire training set is considered with the selected category as the positive class and the left-out categories assigned to the negative class. In this way, RENOIR learns the embedding of each category (in addition to the autoencoder for each category). The classification of a testing sample is performed by computing the distance of the sample from the autoencoder reconstruction in each category. Each distance is computed in the associated embedding space. Finally, the category associated with the minimum distance computed is predicted.

- OVO—We use the One-Versus-One (OVO) combination strategy [49]. In particular, we perform the binary training stage of RENOIR on ten trials—one trial for each pair of categories in the dataset. In each trial we consider a subset of the training set, comprising the samples labelled with the two selected categories, and generate a binary classification model for the selected categories. In the testing stage each binary classifier is used to predict a category (by performing the predictive stage of RENOIR). Hence, we assign a testing sample to the category with the majority counts.

- B+OVO—We perform the intrusion detection and the attack classification in two phases. In the first phase, we use the original formulation of RENOIR to separate normal flows from attack flows. In the second phase, we assign an attack category to the flows predicted as intrusions. The attack classification is decided by using the combination of RENOIR models generated to discriminate between each pair of attack categories and combined through the OVO strategy.

- B+XGBoost—We use the binary classification of the original formulation of RENOIR, in order to distinguish between normal flows and attack flows.

Finally, we use the multi-class XGBoost model, learned from the attack samples, to assign an attack category to the flows predicted as intrusions.

We note that the combination B+XGBoost reproduces the multi-class, DML-based pipeline described in [11]. In particular, B+XGBoost differs from [11] as it uses RENOIR instead of the ensemble of Siamese network, DNN and XGBoost binary classifier, in order to separate the normal flows from the intrusions.

For the comparative study, we consider the performance of several recent competitors that handle the same multi-class problem addressed here, by integrating various techniques to deal with rare attacks. In particular, we consider:

- DML multi-class competitors: SIAM-IDS[10][11] and I-SiamIDS [11].
- DL multi-class competitors with rare data augmentation: DSSTE+AlexNet[47], IGAN-IDS[34] and ID-CAVE[48].
- Deep Reinforcement Learning competitors: AESMOTE[51] and AE-RL[15].

Both the multi-class configurations of RENOIR (OVA, OVO, B+OVO and B+XGBoost) and the multi-class competitors (SIAM-IDS, I-SiamIDS, DSSTE+AlexNet, IGAN-IDS, ID-CAVE, AESMOTE and AE-RL) are run with the same publicly available training and testing sets. The performance of the competitors is collected from the reference papers.

Figures 11a, 11b and 11c report the Precision (P), Recall (R) and F1-scores (F1-score), respectively, of OVA, OVO, B+OVO and B+XGBoost. They are computed per class (with each class versus the others). The top-ranked performance in terms of F1-score is obtained by the two-phase strategies (B+OVO B+XGBoost), except for the class U2R, for which the best F1-Score is achieved by OVO.

For the comparative analysis, we analyse the Macro-Average F1, Micro-Average F1 and Weighted F1. The results of Macro-Average F1, Micro-Average F1 and Weighted F1 of both the multi-class configurations of RENOIR and its competitors are reported in Table 10. All the competitors have been evaluated

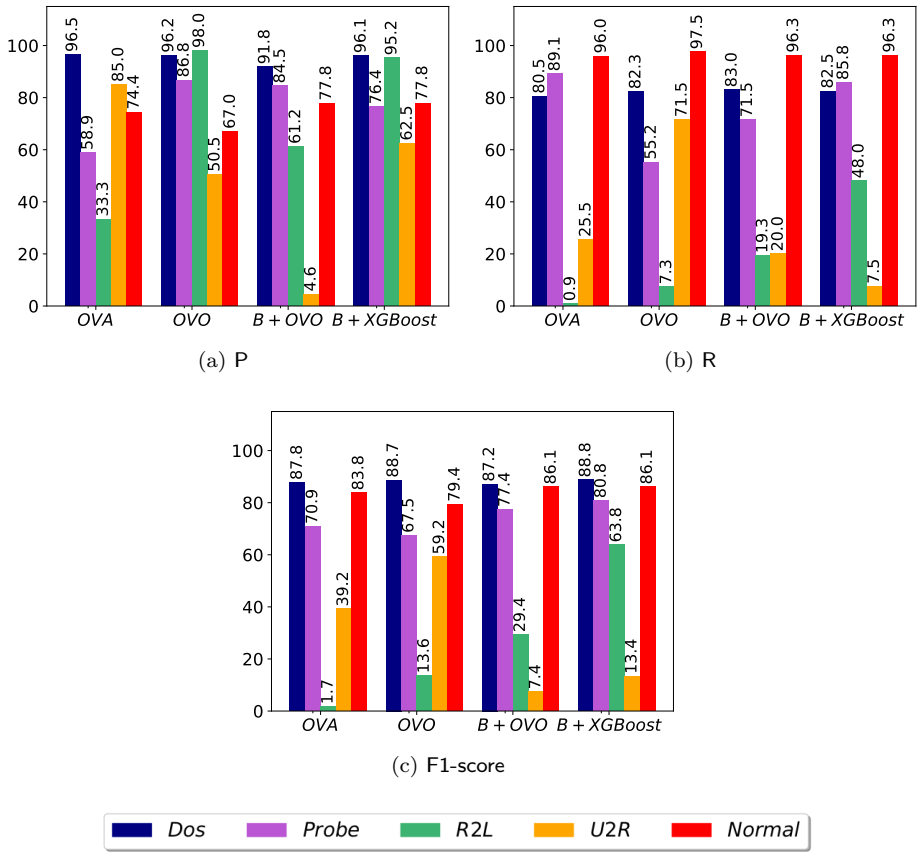


Figure 11: Multi-class NSL-KDD: Precision–P (Figure 11a), Recall–R (Figure 11b) and F1-score (Figure 11c) measured on the testing set for each class by OVA, OVO, B+OVO and B+XGBoost.

in the same experimental setting considered for running the multi-class configurations of RENOIR. No values of the Macro-Average F1 are reported in the reference papers for DSSTE+AlexNet, IGAN-IDS, AESMOTE and AE-RL. These aggregate metrics confirm that the best multi-class configuration of RENOIR is B+XGBoost. In addition, we note that the DML-based algorithms—SIAM-IDS and I-SiamIDS are the nearest-related competitors to RENOIR. Interestingly, all the multi-class configurations of RENOIR outperform SIAM-IDS that, similarly to our methodology, adopts a distance-based approach in the embedding space for the multi-class classification. On the other hand, only the configuration

B+XGBoost of RENOIR outperforms I-SiamIDS. This is an expected outcome, as both algorithms use a DML methodology for the binary classification and resort to a multi-class XGBoost-based re-classification of the attacks. So, this experiment contributes to assess the effectiveness of our methodology in separating normal samples from attacks. In addition, it highlights that resorting to XGBoost for the attack classification can achieve good performance. On the other hand, B+XGBoost also outperforms the remaining competitors, that have been defined in the recent literature to deal with the multi-class problem in the imbalanced scenario of NSL-KDD. The only exception is IGAN-IDS, that uses a deep Generative Adversarial Network process to generate new samples for the minority classes. Therefore, this improvement is achieved at the cost of a complex process of augmentation of the training set. In any case, this result suggests that new milestones may be reached in the future by injecting Generative Adversarial Learning mechanisms into DML approaches to network intrusion detection.

In general, this study provides initial evidence that the proposed methodology, although originally formulated for a binary task, may also have potential in the multi-class scenario. In any case, This conclusion paves the way for a systematic investigation of the topic in the future, as additional work (that is out of the scope of this paper) is required to obtain a multi-class version with a single training stage.

7. Conclusions

Although several effective deep learning methodologies have been recently formulated in the network intrusion detection literature, they are commonly defined as neglectful of possible learning issues, due to the imbalance condition of the network traffic data. As DML techniques are commonly good at dealing with imbalanced data, we present a novel DML methodology, named RENOIR, that defines an innovative Triplet network approach by leveraging autoencoder information for effective network intrusion detection.

Table 10: Results of the Macro-average F1, the Micro-Average (OA) and the weighted F1 obtained by our methodology compared with several competitors that perform multi-class classification in NSL-KDD. The best results are in bold. The results of the competitors are collected from the reference papers. "-" denotes that no value is reported in the reference paper.

Algorithm	Macro-Average F1	Micro-Average (OA)	Weighted F1
OVA	58.40	77.80	73.33
OVO	61.68	76.69	72.99
B+OVO	57.50	79.13	77.89
B+XGBoost	**66.58**	83.91	83.05
SIAM-IDS[10][11]	56.14	76.96	75.28
I-SiamIDS[11]	66.54	79.90	78.49
DSSTE+AlexNet[47]	-	82.84	81.66
IGAN-IDS[34]	-	**84.45**	**84.17**
AESMOTE[51]	-	82.09	82.43
AE-RL[15]	-	80.16	79.40
ID-CAVE[48]	59.27	80.10	79.08

We evaluate the effectiveness of RENOIR using three benchmark datasets that contain imbalanced network flows, collected in different years and scenarios. The empirical validation proves that class-specific autoencoders disclose useful patterns to separate samples belonging to opposite classes. In addition, autoencoders allow us to prevent the problem of triplet convergence that commonly occurs when the random sampling strategy is used for the triplet construction. Furthermore, the empirical validation confirms that this predictive algorithm efficiently yields accurate predictions, thus gaining accuracy compared to recent state-of-the-art competitors (comprising DML-based methods).

One limitation of the proposed methodology is that it was originally formulated to address a binary classification problem to detect attacks in the network traffic, regardless of the attack category. However information on the attack type (e.g. Probe, DoS, R2L or U2R) may be also desired in a network intrusion detection system. In this study, we perform a preliminary investigation of the potential of the proposed methodology in the attack classification. This

investigation is conducted by using combination strategies (one-versus-all or one-versus-one) to combine the binary models trained by RENOIR in the multi-class scenario. Interestingly, we achieve encouraging results that are competitive with those produced by recent, multi-class competitors (comprising multi-class competitors formulated in DML). In any case, this solution requires a burden of computation, since multiple Triplet networks are learned simultaneously. As future work, we plan to continue the investigation in this direction, by exploring how the Triplet network strategy can be extended to account for the multi-class profile of the data in a single training stage.

Another limitation of the proposed methodology is that it does not give detailed information on the structure and characteristics of the attacks. Therefore, explainable artificial intelligence may be an additional research direction here. We plan to explore how the information provided by the explanation may aid in retrieving an explanation of the attack signature.

Finally, the described learning stage is performed in a batch manner, without integrating any concept drift detection mechanism to properly fit learning to an evolving streaming environment.So, a further research direction is would be to extend the methodology to adapt the network intrusion detection model continuously when new data arrive.

Acknowledgments

We acknowledge the support of the MIUR-Ministero dell'Istruzione dell'Università e della Ricerca through the project "TALIsMan -Tecnologie di Assistenza personALizzata per il Miglioramento della quAlità della vitA" (Grant ID: ARS01_01116), the PON RI 2014-2020 funding scheme, as well as the project "Modelli e tecniche di data science per la analisi di dati strutturati", funded by the University of Bari "Aldo Moro". The authors also wish to thank Lynn Rudd for her help in reading the manuscript.

References

[1] Aggarwal, C. C. (2018). *Neural Networks and Deep Learning - A Textbook*.

[2] Ahmad, Z., Khan, A., Shiang, C., & Ahmad, F. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, (pp. 1–29).

[3] Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., & Janicke, H. (2019). A novel hierarchical intrusion detection system based on decision tree and rules-based models. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)* (pp. 228–233). IEEE.

[4] AL-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*, *41*, 1 – 11.

[5] Andresini, G., Appice, A., Caforio, F., & Malerba, D. (2021). Improving cyber-threat detection by moving the boundary around the normal samples. In Y. Maleh, Y. Baddi, M. Shojaafar, & M. Alaza (Eds.), *Machine Intelligence and Big Data Analytics For Cybersecurity Applications* Studies in Computational Intelligence (pp. 105–127).

[6] Andresini, G., Appice, A., Di Mauro, N., Loglisci, C., & Malerba, D. (2019). Exploiting the auto-encoder residual error for intrusion detection. In *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)* (pp. 281–290). IEEE.

[7] Andresini, G., Appice, A., Mauro, N. D., Loglisci, C., & Malerba, D. (2020). Multi-channel deep feature learning for intrusion detection. *IEEE Access*, *8*, 53346–53359.

[8] Angelo, P., & Costa Drummond, A. (2018). Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling. *Security and Privacy*, *1*, 1–13.

[9] Araujo, F., Ayoade, G., Al-Naami, K., Gao, Y., Hamlen, K., & Khan, L. (2019). Improving intrusion detectors by crook-sourcing. In *Proceedings of the 35th Annual Computer Security Applications Conference* ACSAC '19 (pp. 245–246).

[10] Bedi, P., Gupta, N., & Jindal, V. (2020). Siam-ids: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Procedia Computer Science*, *171*, 780 – 789. Third International Conference on Computing and Network Communications (CoCoNet'19).

[11] Bedi, P., Gupta, N., & Jindal, V. (2021). I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems. *Applied Intelligence*, *51*, 1133–1151.

[12] Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *ICML* (pp. 115–123).

[13] Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information*, *10*, 1–35.

[14] Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., & Shah, R. (1993). Signature verification using a siamese time delay neural network. (pp. 737–744). volume 7.

[15] Caminero, G., Lopez-Martin, M., & Carro, B. (2019). Adversarial environment reinforcement learning algorithm for intrusion detection. *Computer Networks*, *159*, 96 – 109.

[16] Chechik, G., Sharma, V., Shalit, U., & Bengio, S. (2010). Large scale online learning of image similarity through ranking. *J. Mach. Learn. Res.*, *11*, 1109–1135.

[17] Chen, W., Chen, X., Zhang, J., & Huang, K. (2017). Beyond triplet loss: A deep quadruplet network for person re-identification. (pp. 1–10). volume abs/1704.01719.

[18] Dai, C., Feng, J., & Zhou, R. (2020). Learning domain-specific features from general features for person re-identification. *IEEE Access*, *PP*, 1–1.

[19] Dan, L., Dacheng, C., Baihong, J., Lei, S., Jonathan, G., & See-Kiong, N. (2019). Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In *Artificial Neural Networks and Machine Learning* (pp. 703–716).

[20] Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for internet of things. *Future Generation Computer Systems*, *82*, 761 – 768.

[21] Elsherif, A. (2018). Automatic intrusion detection system using deep recurrent neural network paradigm. In *J. Inf. Secur. Cybercrimes Res. (JISCR)* (pp. 28–41).

[22] Gao, Y., Li, Y.-F., Chandra, S., Khan, L., & Thuraisingham, B. (2019). Towards self-adaptive metric learning on the fly. (pp. 503–513).

[23] Gautheron, L., Habrard, A., Morvant, E., & Sebban, M. (2020). Metric learning from imbalanced data with generalization guarantees. *Pattern Recognition Letters*, *133*, 298 – 304.

[24] Ge, W., Huang, W., Dong, D., & Scott, M. R. (2018). Deep metric learning with hierarchical triplet loss. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (pp. 272–288). Cham: Springer International Publishing.

[25] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *AISTATS* (pp. 315–323). JMLR.org.

[26] Gresse, A., Quillot, M., Dufour, R., Labatut, V., & Bonastre, J. (2019). Similarity metric based on siamese neural networks for voice casting. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6585–6589).

[27] Guillaumin, M., Verbeek, J., & Schmid, C. (2009). Is that you? metric learning approaches for face identification. In *2009 IEEE 12th International Conference on Computer Vision* (pp. 498–505).

[28] Han, J., & Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In J. Mira, & F. Sandoval (Eds.), *From Natural to Artificial Neural Computation* (pp. 195–201). Berlin, Heidelberg: Springer Berlin Heidelberg.

[29] Hassan, M. M., Gumaei, A., Alsanad, A., Alrubaian, M., & Fortino, G. (2020). A hybrid deep learning model for efficient intrusion detection in big data environment. *Information Sciences*, *513*, 386 – 396.

[30] He, Y. (2019). Identification and processing of network abnormal events based on network intrusion detection algorithm. *I. J. Network Security*, *21*, 153–159.

[31] Hermans, A., Beyer, L., & Leibe, B. (2017). In Defense of the Triplet Loss for Person Re-Identification. *arXiv preprint arXiv:1703.07737*, (pp. 1–17).

[32] Hoffer, E., & Ailon, N. (2015). Deep metric learning using triplet network. In A. Feragen, M. Pelillo, & M. Loog (Eds.), *Similarity-Based Pattern Recognition* (pp. 84–92). Cham: Springer International Publishing.

[33] Hu, S., Feng, M., Nguyen, R. M. H., & Lee, G. H. (2018). Cvm-net: Cross-view matching network for image-based ground-to-aerial geo-localization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7258–7267).

[34] Huang, S., & Lei, K. (2020). Igan-ids: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Networks*, *105*, 1–11.

[35] Jia, W., Yang, M., & Wang, S. (2017). Three-category classification of magnetic resonance hearing loss images based on deep autoencoder. *Journal of Medical Systems*, *41*, 1–11.

[36] Jmila, H., Ibn Khedher, M., Blanc, G., & El Yacoubi, M. A. (2019). Siamese network based feature learning for improved intrusion detection. In T. Gedeon, K. W. Wong, & M. Lee (Eds.), *Neural Information Processing* (pp. 377–389). Cham: Springer International Publishing.

[37] Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*, 1–54.

[38] Kaya, M., & Bilge, H. (2019). Deep metric learning: A survey. *Symmetry*, *11*, 1–26.

[39] Kherlenchimeg Zolzaya, N. N. (2018). Network intrusion classifier using autoencoder with recurrent neural network. In *ICESS* (pp. 94–100).

[40] Kim, T., Suh, S. C., Kim, H., Kim, J., & Kim, J. (2018). An encoding technique for cnn-based network anomaly detection. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 2960–2965). IEEE.

[41] Kuai, Y., Wen, G., & Li, D. (2019). Masked and dynamic siamese network for robust visual tracking. *Information Sciences*, *503*, 169 – 182.

[42] Lashkari, A. H., Kadir, A. F. A., Gonzalez, H., Mbah, K. F., & Ghorbani, A. A. (2017). Towards a network-based framework for android malware detection and characterization. In *PST* (pp. 233–234). IEEE Computer Society.

[43] Lee, J., & Park, K. (2019). Gan-based imbalanced data intrusion detection system. *Personal and Ubiquitous Computing*, (pp. 1–8).

[44] Li, P., Yi, J., Zhou, B., & Zhang, L. (2019). Improving the robustness of deep neural networks via adversarial training with triplet loss. In S. Kraus (Ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019* (pp. 2909–2915). ijcai.org.

[45] Li, Y., Ma, R., & Jiao, R. (2015). A hybrid malicious code detection method based on deep learning. In *International journal of security and its applications* (pp. 205–216). volume 9.

[46] Li, Z., Qin, Z., Huang, K., Yang, X., & Ye, S. (2017). Intrusion detection using convolutional neural networks for representation learning. In *ICONIP* (pp. 858–866). Springer International Publishing.

[47] Liu, L., Wang, P., Lin, J., & Liu, L. (2021). Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access*, *9*, 7550–7563.

[48] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors*, *17*, 1–17.

[49] Lorena, A. C., de Leon Ferreira de Carvalho, A. C. P., & Gama, J. (2008). A review on the combination of binary classifiers in multiclass problems. *Artif. Intell. Rev.*, *30*, 19–37.

[50] Lu, J., Hu, J., & Zhou, J. (2017). Deep metric learning for visual understanding: An overview of recent advances. *IEEE Signal Processing Magazine*, *34*, 76–84.

[51] Ma, X., & Shi, W. (2020). Aesmote: Adversarial reinforcement learning with smote for anomaly detection. *IEEE Transactions on Network Science and Engineering*, (pp. 1–1).

[52] Macek, K. (2008). Pareto principle in datamining: an above-average fencing algorithm. In *Acta polytechnica* (pp. 55–59). volume 48(6).

[53] Medela, A., & Picón, A. (2019). Constellation loss: Improving the efficiency of deep metric learning loss functions for optimal embedding. *CoRR*, *abs/1905.10675*.

[54] Mignon, A., & Jurie, F. (2012). Pcca: A new approach for distance learning from sparse pairwise constraints. (pp. 2666–2672).

[55] Moustakidis, S. P., & Karlsson, P. (2020). A novel feature extraction methodology using siamese convolutional neural networks for intrusion detection. *Cybersecurity*, *3*, 1–13.

[56] Naseer, S., Saleem, Y., Khalid, S., Bashir, M. K., Han, J., Iqbal, M. M., & Han, K. (2018). Enhanced network anomaly detection based on deep neural networks. *IEEE Access*, *6*, 48231–48246.

[57] Nawaz, S., Calefati, A., Ahmed, N., & Gallo, I. (2018). Hand written characters recognition via deep metric learning. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)* (pp. 417–422).

[58] Parsons, V. L. (2017). Stratified sampling. In *Wiley StatsRef: Statistics Reference Online* (pp. 1–11). American Cancer Society.

[59] Qu, X., Yang, L., Guo, K., Ma, L., Feng, T., Ren, S., & Sun, M. (2019). Statistics-enhanced direct batch growth self-organizing mapping for efficient dos attack detection. *IEEE Access*, *7*, 78434–78441.

[60] Schlegl, T., Seeböck, P., Waldstein, S. M., Schmidt-Erfurth, U., & Langs, G. (2017). Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In M. Niethammer, M. Styner, S. Aylward, H. Zhu, I. Oguz, P.-T. Yap, & D. Shen (Eds.), *Information Processing in Medical Imaging* (pp. 146–157). Cham: Springer International Publishing.

[61] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 815–823).

[62] Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. (pp. 108–116).

[63] Shaw, B., Huang, B., & Jebara, T. (2011). Learning a distance metric from a network. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 24* (pp. 1899–1907). Curran Associates, Inc.

[64] Shi, H., Yang, Y., Zhu, X., Liao, S., Lei, Z., Zheng, W., & Li, S. Z. (2016). Embedding deep metric for person re-identification: A study against large variations. In B. Leibe, J. Matas, N. Sebe, & M. Welling (Eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I* (pp. 732–748). Springer volume 9905 of *Lecture Notes in Computer Science*.

[65] Shone, N., Ngoc, T. N., Phai, V. D., & Shi, Q. (2018). A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *2*, 41–50.

[66] Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems* NIPS'16 (p. 1857–1865). Red Hook, NY, USA: Curran Associates Inc.

[67] Song, H. O., Xiang, Y., Jegelka, S., & Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *Computer Vision and Pattern Recognition (CVPR)* (pp. 4004–4012).

[68] Souza, J., Oliveira, L., Gumiel, Y., Carvalho, D., & Moro, C. (2020). Exploiting siamese neural networks on short text similarity tasks for multiple domains and languages. (pp. 357–367).

[69] Sun, J., Lang, J., Fujita, H., & Li, H. (2018). Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. *Information Sciences*, *425*, 76–91.

[70] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *CISDA* (pp. 1–6).

[71] Velan, P., Medková, J., Jirsík, T., & Čeleda, P. (2016). Network traffic characterisation using flow-based statistics. In *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium* (pp. 907–912).

[72] Vigneswaran, R. K., Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2018). Evaluating shallow and deep neural networks for network intrusion detection systems in cyber security. In *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1–6).

[73] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, *7*, 41525–41550.

[74] Wang, J., Zhou, F., Wen, S., Liu, X., & Lin, Y. (2017). Deep metric learning with angular loss. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 2612–2620).

[75] Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, *10*, 207–244.

[76] Yan, J., Jin, D., Lee, C. W., & Liu, P. (2018). A comparative study of offline deep learning based network intrusion detection. In *10th International Conference on Ubiquitous and Future Networks* (pp. 299–304).

[77] Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2015). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, *4*, 259–272.

[78] Yu, B., Liu, T., Gong, M., Ding, C., & Tao, D. (2018). Correcting the triplet selection bias for triplet loss. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), *Computer Vision – ECCV 2018* (pp. 71–86). Cham: Springer International Publishing.

[79] Zenati, H., Foo, C. S., Lecouat, B., Manek, G., & Chandrasekhar, V. R. (2018). Efficient gan-based anomaly detection. *ArXiv*, *abs/1802.06222*.

[80] Zenati, H., Romain, M., Foo, C. S., Lecouat, B., & Chandrasekhar, V. R. (2018). Adversarially learned anomaly detection. *2018 IEEE International Conference on Data Mining (ICDM)*, (pp. 727–736).

[81] Zhang, B., Yu, Y., & Li, J. (2018). Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)* (pp. 1–6).

[82] Zhang, Y., Chen, X., Jin, L., Wang, X., & Guo, D. (2019). Network intrusion detection: Based on deep hierarchical network and original flow data. *IEEE Access*, *7*, 37004–37016.

[83] Zhang, Y., Zhang, H., Zhang, X., & Qi, D. (2018). Deep learning intrusion detection model based on optimized imbalanced network data. In *2018 IEEE 18th International Conference on Communication Technology (ICCT)* (pp. 1128–1132).

[84] ZhangYu-Dong, ZhangYin, HouXiao-Xia, Chenhong, & WangShui-Hua (2018). Seven-layer deep neural network based on sparse autoencoder for voxelwise detection of cerebral microbleed. *Multimedia Tools and Applications*, (p. 10521–10538).

[85] Zhou, Li, & Shen (2019). Anomaly detection of can bus messages using a deep neural network for autonomous vehicles. *Applied Sciences*, *9*, 1–12.

[86] Zhou, M., Tanimura, Y., & Nakada, H. (2020). One-shot learning using triplet network with knn classifier. In Y. Ohsawa, K. Yada, T. Ito, Y. Takama, E. Sato-Shimokawara, A. Abe, J. Mori, & N. Matsumura (Eds.), *Advances in Artificial Intelligence* (pp. 227–235). Cham: Springer International Publishing.

[87] Zhuang, B., Lin, G., Shen, C., & Reid, I. (2016). Fast training of triplet-based deep binary embedding networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–12).