GAN Augmentation to Deal with Imbalance in Imaging-based Intrusion Detection

Giuseppina Andresini[a,*], Annalisa Appice[a,b], Luca De Rose[a], 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, Bari, Italy*

Abstract

[1]

Nowadays attacks on computer networks continue to advance at a rate outpacing cyber defenders' ability to write new attack signatures. This paper illustrates a deep learning methodology for the binary classification of the network traffic. The basic idea is to represent network flows as 2D images and use this imagery representation of the network traffic to train a Generative Adversarial Network (GAN) and a Convolutional Neural Network (CNN). The GAN is trained to produce new images of unforeseen network attacks by augmenting the training data used to learn a CNN-based intrusion detection model. The advantage is that the 2D data mapping technique used builds images of the network flows, which allow us to take advantage of deep learning architectures with convolution layers. In addition, the GAN-based data augmentation allows us to deal with the possible imbalance of malicious traffic that is commonly rarer than the normal traffic in the network traffic. Specifically, it is used to simulate

*Corresponding author (Tel: +39 (0)805443262 Fax: +39(0)805443269)
Email addresses: giuseppina.andresini@uniba.it (Giuseppina Andresini), annalisa.appice@uniba.it (Annalisa Appice), l.derose@studenti.uniba.it (Luca De Rose), 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.future.2021.04.017. Accepted Version is subject to the publisher's Accepted Manuscript terms of use https://www.elsevier.com/about/policies-and-standards/copyright

unforeseen attacks to train a robust intrusion detection model. The proposed methodology leads to better predictive accuracy when compared to competitive intrusion detection architectures on four benchmark datasets.
Keywords: Intrusion detection, Deep learning, Data augmentation, Image encoding

1. Introduction

Nowadays intrusion detection systems represent one of the most powerful weapons of public and private organisations in the war against the ever-increasing amount of network cyber attacks [1]. They are a mandatory line of the network defence process to forewarn security administrators of signs of malicious network traffic. In this scenario, machine learning, with the ability to conduct predictive analysis at a scale beyond human mean, plays a crucial role in modern intrusion detection systems, due to its ability to detect new variants of attack [2].

With the recent boom of deep learning in machine learning, the use of deep neural networks has definitely emerged as a valuable candidate solution for intrusion detection problems [3, 4, 5, 6]. The non-linear activation layers of deep neural networks may facilitate the discovery of effective models that keep their effectiveness also under drifting conditions. This makes deep learning a definitely relevant approach in network security, since thousands of zero-day intrusions commonly occur due to the addition of various protocols yielding small variants of previously known cyber-attacks [7].

Although several studies [8, 4] have proved that deep learning algorithms gain accuracy on traditional machine learning algorithms when large amounts of data are available for training, even deep learning performance declines significantly in the case of learning from imbalanced data [9, 10, 11]. The network traffic produces large amounts of data for deep learning, but the imbalance of malicious data is a crucial issue to handle with these data: network intrusions and malicious behaviour commonly represent a very small subset of all the net-

work traffic [10]. On the other hand, malicious behaviour detection is highly critical for the health of a network. In general, the relative lack of malicious data may cause improper inductive bias in a deep neural network, which cannot be accepted since the high intrusion detection rate gained at the expense of numerous false alarms will lead the loss of a high number of relevant packets [12].

As a traditional solution to intrusion imbalance, data may be re-sampled (i.e the rare data are over-sampled, while the major data are down-sampled) before learning to achieve the balanced condition. However, this solution may lead to over-fitting with learning, since the same over-sampled data are repeatedly learned. To limit over-fitting, various data augmentation methods have been formulated (e.g. SMOTE [13], ADASYN [14], RFMSE [15]), in order to avoid cloning rare data. These methods sample rare data and use them to generate new, artificial, rare-similar samples. However, these approaches often create noise by ignoring the location of general data adjacent to the rare data. Recent studies in deep learning have initiated the investigation of augmenting rare data through a generative adversarial process performed with GANs (Generative Adversarial Networks)[16].

GANs allow us to create artificial data samples that are very similar to the original data, by addressing over-fitting, class overlapping and noise. This is made possible as GANs can re-sample data by specifying the desired rare class with an architecture that is designed to discriminate the real signal (intrusion) from the fake one.

Because of the excellent performance of GANs in image data augmentation [17], in this paper we investigate the viability of training GANs in intrusion detection problems once an appropriate image representation of network flows has been adopted. Following this idea, we illustrate a deep learning intrusion detection methodology, named MAGNETO (iMAge based Gan enhanced convolutional NEural neTwOrk), which leverages the power of image-defined GANs to deal with intrusion imbalance when learning from historical network flows. In particular, we handle the imbalance condition in the binary classification of

network traffic by focusing on the task of detecting any cyber-attack at the network level. Further investigations are requested to extend this study to classify attack families and deal with the class imbalance in the classification of attack families. This may reduce the workload of human analysts who investigate the functionality of the network traffic.

The primary contribution of this study is the evaluation of the effectiveness of learning an intrusion detection model from an appropriate two-dimensional (2D) imagery representation of network flows. The adopted imagery representation, based on the study of [18], allows us to delineate potential data patterns (e.g. edges, shading changes, shapes), that appear on neighbour pixels once these pixels encode similar features of network flows. Leveraging the collection of images representing historical network flows, we are able to train a 2D Convolutional Neural Network (CNN) and build an accurate intrusion detection model. This is a classification model that takes advantage of convolution filters on spatially close features of network flows, in order to discriminate intrusions from normal behaviour. From this point of view, this study is boosted by the amazing results recently achieved with convolutions in image classifications [19]. In particular, it contributes to validating the idea that computer vision can aid in solving cybersecurity problems by providing the techniques to account for the spatial continuity phenomenon among traffic characteristics. These characteristics have been commonly overlooked in state-of-the-art intrusion detection methods. The empirical validation confirms that by making this step, we have achieved an important milestone in cybersecurity. In fact, the proposed image-based classification model gains accuracy compared to various intrusion detection models, comprising those already using convolutions or different deep learning architectures, to yield the final data flow classifications.

The additional contribution is the improvement of the performance of the proposed imaging-based intrusion detection methodology by the introduction of a GAN architecture for augmenting images of intrusions and balancing the training data processed to learn the classification model. From this point of view, this study shows that the image augmentation through GANs can be

more effective than the application of traditional data augmentation techniques in dealing with the imbalance of intrusions. It limits the overfitting and makes the classification model finally learned more robust to the class overlapping, noise and change (possibly occurring with new attacks). This arises in the increase in the number of intrusions detected, which is not achieved at the expense of any significant increase in the number of false alarms.

We note that both 2D CNNs and GANs have been widely explored, also in combination, in the computer vision literature. 2D CNNs have also been explored in intrusion detection in [20, 21], but these studies resort to improper image transformations of non-image data, which do not capture possible spatial continuity among traffic characteristics (see details in Section 2.1). This improper transformation has made the subsequent use of convolutions less effective. This is confirmed by our comparative empirical validation (see Section 5.5). On the other hand, GANs have recently been used to augment samples in rare attack families, but both the augmentation and the subsequent classification are always trained in 1D space (see details in Section 2.2). The augmented training set is used to train a Random Forest in [10], and a Logistic Regression or an SVM or a Feed-forward Deep Neural Network in [22]. So, to the best of our knowledge, the novelty of this study is:

- The specific formulation adopted for the considered learning components, that allows us to capture and model the phenomenon of data continuity across traffic characteristics when generating the image form of data flows. In addition, the proposed formulation allows us to account for this proper imagery representation of the network flows, in order to deal with the imbalance issue (by training GANs on imagery data) and learn a robust classification model (by training 2D CNNs on balanced training sets), as in computer vision investigations.

- The effectiveness of the combination of these components in a methodology that actually outperforms the intrusion detection accuracy of several state-of-the-art competitors on various benchmark data sets.

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 strategy are discussed in Section 5. Finally, Section 6 refocuses on the purpose of the research, draws conclusions and illustrates possible future developments.

2. Related works

The preeminence of deep learning to intrusion detection has been repeatedly assessed in the recent literature [5, 23]. With the boom of deep learning, various deep learning architectures – Autoencoders [23, 24, 25, 26], Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs) [27, 28, 29, 6, 30], Generative Adversarial Networks (GANs) [31, 32, 33, 34] and Convolutional Neural Networks (CNNs) [35, 36, 20, 21, 23, 37, 38] – have been recently investigated for intrusion detection problems [5]. Since in this paper we revamp an intrusion detection pipeline that couples CNN and GAN architectures, we mainly focus this literature overview on both recent studies applying CNNs to discriminate intrusions from normal network traffic (see Subsection 2.1) and machine learning approaches, comprising GANs, to deal with data imbalance in deep learning (see Subsection 2.2).

2.1. Training CNNs for Intrusion Detection

The CNNs are a family of robust, popular neural networks designed to process input data stored in arrays [17]. They are mainly investigated in computer vision for processing 2-dimensional (2D) arrays of images [39, 40] or audio spectrograms [41]. However, they are also used for processing 3D arrays (videos and volumetric images) [42], as well as 1D arrays (signals) [43]. Regardless of their dimension, CNNs perform well where there is spatial or temporal ordering in input data. Both 1D and 2D CNNs have been recently experimented in intrusion detection problems [35, 36, 20, 21, 23, 37, 44].

Kwon et al. [35] process 1D arrays of network flow features studying the performance of 1D CNNs in network intrusion detection problems. They investigate how to optimise the structure of the architecture to gain accuracy in the intrusion detection ability. To this aim, they compare three 1D CNNs designed with growing architectural depth and investigate the impact of the depth of internal layers on the accuracy of the classifier. Even though these authors assume that the deeper the architecture, the higher the accuracy, their evaluation proves that the addition of new layers to a CNN architecture does not guarantee gain in accuracy. This result inspires the idea that a light architecture (with a few convolution layers) can already achieve high accuracy in intrusion detection.

Andresini et al. [36] also process 1D arrays of network flow features by evaluating the performance of 1D CNNs coupled with autoencoders and multi-channel convolutions. First they consider two autoencoders separately learned on normal and attack flows, respectively. Then they use samples reconstructed with these autoencoders to define two new feature vectors. This allows the representation of each network flow as a multi-channel sample. Finally, they adopt multi-channel parametric convolutions in the 1D CNNs, to learn the effect of each channel on the others in the final intrusion detection model.

In place of 1D CNNs, Li et al. [20], as well as Kim et al. [21], consider 2D CNNs. In fact, they focus their research on how network flows can be mapped into 2D image arrays that express latent features of input data within a 2D data representation. In particular, Li et al. [20] describe a quantization method to convert the value of each numeric feature into an 8-digit binary pixel. In this way, feature vectors describing features of network flows can be converted into 8×8 pixels. The input representation built with this method is finally processed as the input of two popular CNNs, that is, ResNet50 [45] and GoogLeNet [46]. Kim et al. [21] extend the method described in [20] by introducing an RGB-like encode of the data. They determine an $N \times N$ encoding of network flow data, which is built on $(10 \times N + M)/8$ pixels, where N is the number of numeric features and M the number of categorical features. This input representation is processed in combination with GoogLeNet Inception V3 [46]. Experiments

described in [21] prove that their approach outperforms the seminal one in [20].

2D-CNNs are also investigated in malware detection. For example, in [47] an application binary file is first read as a sequence of vectors of 8-bit unsigned integer values. Then each binary value is converted into a decimal value (e.g. the binary value [11111111] is converted into the decimal value [255]). Finally, the resulting decimal vector is reshaped into a 2D matrix that represents the grey-scale image input to the 2D CNN. A similar imagery encoding approach is described in [48].

All the above mentioned studies with 2D CNN architectures are close to the research described here. In fact, similarly to the study presented in this paper, they transform the vector representation of network flows (or of malicious applications) into images to train a 2D CNN architecture. However, in these studies, features of network flows are arranged into imagery pixels of a 2D grid by following the order according to which the features are stored in the original vector form. As there is no certain correlation between the consecutive features of a vector, these imagery encoding methods cannot guarantee that the produced images will show continuity in the intensities of feature values associated with neighbour pixels – salt-and-pepper images are plausibly generated. On the other hand, shifting the attention from cybersecurity to medicine, a proper image encoding method to transform gene vector data to a well-organised image form has been recently introduced in [18]. This image encoding method differs from the imaging approaches experimented in cybersecurity until now, since it is able to associate similar gene features to neighbour pixels, by creating the demanded continuity in the neighbour pixel intensity. In this paper, we adapt the gene imaging encoding method described in [18] to fit the considered cybersecurity domain. In the comparative analysis with the state-of-the-art intrusion detection systems that use image encoding methods, we provide the evidence that modelling the continuity across features in the network traffic image generation can aid in accuracy.

2.2. Dealing with Imbalance in Deep Learning

Due to the ubiquity of imbalanced data, techniques to improve the performance of deep learning from imbalanced data are formulated with respect to various domains. Lee et al. [49] describe a two-stepped learning pipeline to deal with the imbalanced problem in the plankton classification with CNNs. In the first step, they randomly under-sample majority classes and use the sample to pre-train a CNN. In the second step, they fine-tune the CNN with all the data. Pouyanfar et al. [50] apply a real-time augmentation technique to balance the landscape image dataset processed to train a CNN. They augment the training set with a batch of new artificial images that are produced by applying traditional image transformation operations (i.e. shear, brightness, rotation, shift and slip) to the real images. They use transfer learning with an Inception network [46], pre-trained using ImageNet data [51], to fine-tune the model. Wang et al. [9], as well as Lin et al. [52] describe new loss functions that down-weight the loss assigned to well-classified examples, by increasing the contribution of rare class samples when training DNNs and CNNs, respectively.

By focusing the attention on imbalance in intrusion detection, Zhang et al. [53] describe an intrusion detection pipeline that integrates imbalanced class processing with 1D CNNs. The imbalance condition is handled by combining intrusion data augmentation through the Synthetic Minority Over-Sampling Technique (SMOTE) and normal traffic under-sampling through clustering based on the Gaussian Mixture Model (GMM). Andresini et al. [54] describe a completely different approach to deal with the imbalance condition in network intrusion detection. Instead of augmenting the training set by creating new artificial intrusions, they learn the decision boundary that separates the normal training samples from the intrusions. They re-assign the normal training samples that are close to the boundary to the opposite class and train the multi-channel 1D CNN, already introduced in [36], on the modified training set.

A recent research trend in deep learning has mainly invested in dealing with imbalance through data augmentation with GANs. These adversarial deep neural networks [16] are mainly used in data augmentation [55, 56, 57, 58, 59, 10],

as well as in anomaly detection [60, 61, 62, 32]. They contribute to laying the basis for adversarial training, in order to improve the robustness of neural networks to new adversarial samples injected into the training set [63, 64]. There are several research directions conducted in adversarial learning, e.g. defence distillation [65] and gradient masking [66], to defend training from adversarial samples. In this paper, we focus on Generative Adversarial Learning.

GANs for data augmentation are mainly experimented in medicine [55, 56, 57] and remote sensing [58, 59], where they are commonly coupled with 2D CNNs as deep neural networks for image classification. A few studies have started the exploration of GANs for dealing with imbalance in cybersecurity. In [10] GANs are used for data augmentation of network traffic data represented in 1D arrays of flow features, while a Random Forest is subsequently trained with the augmented training set. A similar approach is illustrated in [22], with GANs used for data augmentation, while Logistic Regression, SVMs or Feed-forward Deep Neural Networks are trained for the classification. Shin et al. [67] propose the use of Sequence Generative approaches (SeqGAN and Seq2Seq) to generate new data in a sequence of network flows. Finally, Wang et al. [68] use the random feature nullification to build an adversary-resistant deep learning model in malware detection.

All the above mentioned studies with GANs for data augmentation are close to the research described here, as they use GANs to generate rare adversarial samples and achieve a balanced condition before training the classification model. However, to the best of our knowledge, the GANs that have already been experimented for data augmentation in network intrusion detection consider array data, while the highest performance of GANs is commonly achieved with image data [17].

3. The intrusion detection methodology of MAGNETO

In this Section, we describe MAGNETO – a supervised deep learning methodology for learning a robust network intrusion model, by dealing with possible

intrusion data imbalance. Let us consider:

- $\mathbf{X^{1D}}$ – a 1D feature vector that comprises M features $X_1, \ldots, X_M$. Each feature X_i describes an independent characteristic of a network flow (e.g. the number of transmitted packets and the number of failed login).
- Y – a binary dependent target with "normal" and "attack" labels.
- ($\mathbf{T}$, $\mathbf{Y}$) – a training set that collects N historical network flows (training samples), spanned on $\mathbf{X^{1D}}$ and labelled with Y. $\mathbf{T}$ is the $N \times M$ matrix that represents the training samples on the rows and the features of $\mathbf{X^{1D}}$ on the columns. $\mathbf{Y}$ is the $N \times 1$ vector that collects the labels of the training samples. Every training sample $\mathbf{e_i} \in (\mathbf{T}, \mathbf{Y})$ is the couple composed of the i-th row of $\mathbf{T}$ and the i-th label of $\mathbf{Y}$, respectively.

According to the imbalance condition, we expect the label "attack" to be a rare class in $\mathbf{Y}$ (i.e. the number of attacks is significantly lower than the number of normal network flows in the network traffic). MAGNETO inputs ($\mathbf{T}$, $\mathbf{Y}$) and learns the intrusion detection function $\mathbf{X^{1D}} \mapsto Y$ through a three-stepped methodology. This methodology cascades:

- An image encoding step that transforms the training samples from the 1D feature vector form to a 2D image form – one 2D grey-scale image is constructed for each 1D representation of a network flow.
- A GAN-based imagery data augmentation procedure that trains a GAN architecture – specifically an Auxiliary Classifier GAN – ACGAN [69] – to learn the distribution of the imagery training set. It uses the *generator* of the ACGAN to build new images of artificial attacks and balance the training set.
- A 2D CNN architecture that is trained on the augmented imagery training set to discriminate attacks from normal network flows.

A block diagram of this methodology is illustrated in Figure 1, while the adopted notation is introduced in Table 1.

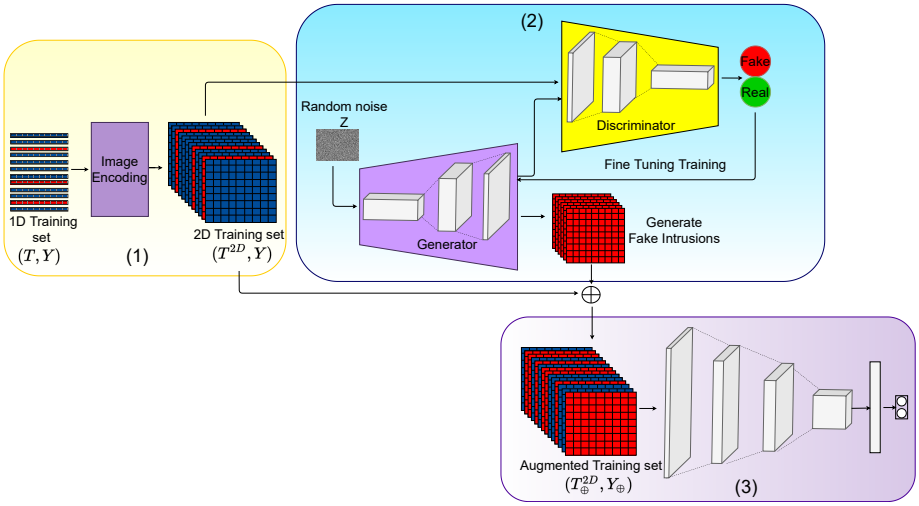


Figure 1: The block diagram of MAGNETO that takes the training set $(\mathbf{T}, \mathbf{Y})$ as input. (1) It converts $\mathbf{T}$ into $\mathbf{T^{2D}}$ by transforming each training sample from the non-image form to the grey-scale image form. (2) It processes $(\mathbf{T^{2D}}, \mathbf{Y})$ to train an ACGAN architecture and uses the ACGAN generator to build new images of artificial attacks and balance the training set. (3) It processes the augmented training set $(\mathbf{T}^{\mathbf{2D}}_{\oplus}, \mathbf{Y}_{\oplus})$ to train a 2D CNN architecture that discriminates attacks from normal network flows.

3.1. Image encoding

In the first step we map the 1D feature vector representation of the input data onto the 2D image representation. First we find the spatial locations that the features of $\mathbf{X^{1D}}$ occupy in a Cartesian plane and use these locations to assign every feature of $\mathbf{X^{1D}}$ to one pixel frame of $\mathbf{X^{2D}}$ – the 2D grid with size $n \times m$. Both n and m are user-defined parameters of the image encoding step – they are selected so that $M \leq nm$.[2] Then we use $\mathbf{X^{2D}}$ as the design to transform every non-image sample into its image form. In particular, the grey-scale image of a sample is constructed by assigning each feature value of

[2]The satisfaction of this constraint is the necessary condition to allow, in theory, the assignment of each feature of $\mathbf{X^{1D}}$ to a distinct pixel frame of $\mathbf{X^{2D}}$. In Section 5.3.3, we investigate the sensitivity of the proposed intrusion detection methodology to the set-up of the image grid size.

Table 1: Notation

Symbol	Description
$\mathbf{X^{1D}}$	1D feature vector with M independent features $X_1, \ldots, X_m$
$\mathbf{X^{2D}}$	2D grid covering $n \times m$ pixel frames
Y	binary target with labels in the set $\{normal, attack\}$
$\mathbf{T}, \mathbf{Y}$	training set comprising N network flows spanned on $\mathbf{X}^{1D}$ and labelled on Y
$\mathbf{T}$	training data matrix with size $N \times M$
$\mathbf{Y}$	training label vector with size $N \times 1$
$\mathbf{T}'$	transpose of $\mathbf{T}$ – data matrix with size $M \times N$
$\mathbf{T^{2D}}$	training data hypercube of size $N \times n \times m$
$\mathbf{T}^{\mathbf{2D}}_{\oplus}$	augmented training data hypercube

the sample to a pixel frame according to the feature-pixel association defined in $\mathbf{X^{2D}}$. Mathematically this step allows us to transform the training data matrix $\mathbf{T}$ of size $N \times M$ into the training data hypercube $\mathbf{T^{2D}}$ of size $N \times n \times m$.[3]

In this paper $\mathbf{X^{2D}}$ is determined by replicating the imagery encoding pipeline introduced in [18]. Specifically:

- We determine $\mathbf{T}'$ – the matrix of size $M \times N$ – that is obtained by transposing $\mathbf{T}$. This matrix represents traffic characteristics of $\mathbf{X^{1D}}$ on rows and training samples on columns.

- We apply t-SNE [70][4] – a non-linear dimensionality reduction technique that is well-suited for the visualization of high-dimensional data in a 2D Cartesian space. Specifically, we use t-SNE to transform the matrix $\mathbf{T}'$ with size $M \times N$ to the matrix $\mathbf{tsne}(\mathbf{T}')$ with size $M \times 2$. In this way, we assign each traffic characteristic $X_i \in \mathbf{X^{1D}}$ to a 2D point of a Cartesian plane with coordinates $\mathbf{tsne}(\mathbf{T}')[i, 1]$ and $\mathbf{tsne}(\mathbf{T}')[i, 2]$, respectively. The transformation is completed in two stages, according to the theory reported in [70]. Initially, t-SNE determines a probability distribution over each pair of traffic characteristics acquired on the training data flows (i.e.

[3]The feature locations, determined for the image encoding of the training set, are also used to encode each non-image testing network flow into a 2D grey-scale image.

[4]In principle, any linear or non-linear dimensionality reduction technique can be used.

on the row vectors of $\mathbf{T}'$). The similarity of each traffic characteristic X_j to characteristic X_i (with $i \neq j$) is measured as the conditional probability, $p(X_j|X_i)$, that X_i would pick X_j as its neighbour, if neighbours were picked in proportion to their probability density under a Gaussian centred at X_i. Thus, similar characteristics are assigned to a higher probability, while dissimilar characteristics are assigned to a lower probability. Finally, t-SNE defines a similar probability distribution over the traffic characteristics in the Cartesian map, and it minimizes the Kullback–Leibler divergence (KL divergence) between the two distributions with respect to the locations of the characteristics in the map. Once t-SNE transformation has been completed, the rows of $\mathbf{tsne}(\mathbf{T}')$ still represent the traffic characteristics of $\mathbf{X^{1D}}$, while the columns of $\mathbf{tsne}(\mathbf{T}')$ define 2D coordinates that allow us to visualize the characteristics of $\mathbf{X^{1D}}$ as points of a Cartesian plane.

- We use the convex hull algorithm [71] to find the smallest rectangle (minimum bounding box) containing all the 2D points associated via the t-SNE transformation with the features of $\mathbf{X^{1D}}$, and rotate it to frame feature points to the 2D Cartesian plane in either a horizontal or vertical form. The rotated rectangle can be represented by min(x), max(x), min(y) and max(y)—the minimum and maximum coordinates along the 2D Cartesian axes x and y, respectively. It is segmented into a 2D grid $\mathbf{X^{2D}}$ of $n \times m$ equally-sized rectangular pixel frames with length $\frac{max(x)-min(x)}{n}$ and width $\frac{max(y)-min(y)}{m}$.

  Finally, every feature of $\mathbf{X^{1D}}$ is one-to-one assigned to the pixel frame of $\mathbf{X^{2D}}$ that hosts its point location of $\mathbf{tsne}(\mathbf{T}')$, as it is rotated with the convex hull algorithm.

One technical issue that may occur with the pipeline described above is that locations of various features of $\mathbf{X^{1D}}$ may enter into collision since they fall in the same pixel frame. In [18], intra-pixel collisions have been addressed by constructing a pseudo-feature for each group of colliding features. The pseudo-

feature, that is the average of the intra-pixel colliding features, is assigned to the pixel frame, while the original colliding features are definitely discarded. This is an intra-pixel collision technique that performs feature construction. However, in our opinion it presents a theoretical issue – it requires the computation of the average of values measured on heterogeneous characteristics, while the average operator is an aggregate of homogeneous values of the same characteristic.

To overcome this issue, we introduce a different intra-pixel collision technique that operates as a feature selector instead of a feature constructor. In particular, it abstains from averaging heterogeneous features, opting for a supervised evaluation of the ability of colliding features to discriminate attacks from normal network flows. To this aim, we rank each group of features entering into collision on a pixel frame, according to their Mutual Info. This measure is commonly used for feature scoring in classification problems [72], as it quantifies the amount of information obtained about the target through observing each feature to be analyzed.

An example of collision is shown in Figure 2 that sketches the transformation of the non-image traffic characteristics of CICIDS17[5] data flows into their imagery form. This example highlights groups of traffic characteristics that collide during the imagery transformation (e.g. Fwd Packet Length Max—X_7, Avg Fwd Segment Size—X_{54} and Fwd Packet Length Mean—X_9). The feature of the collision group with the highest Mutual Info (i.e. Fwd Packet Length Max) is selected and assigned to the pixel frame, while the remaining features colliding at the same pixel frame are discarded by achieving a feature selection objective. The effectiveness of the proposed Mutual Info-based intra-pixel collision technique is empirically investigated in Section 5.3.1.

Final considerations concern the advantage of adopting this image encoding for the subsequent steps of the learning methodology. It accounts for possible patterns of spatial continuity that arise among the network traffic features – features which tend to assume similar values on the training samples are as-

[5]This dataset is subsequently used in the empirical validation described in Section 5.

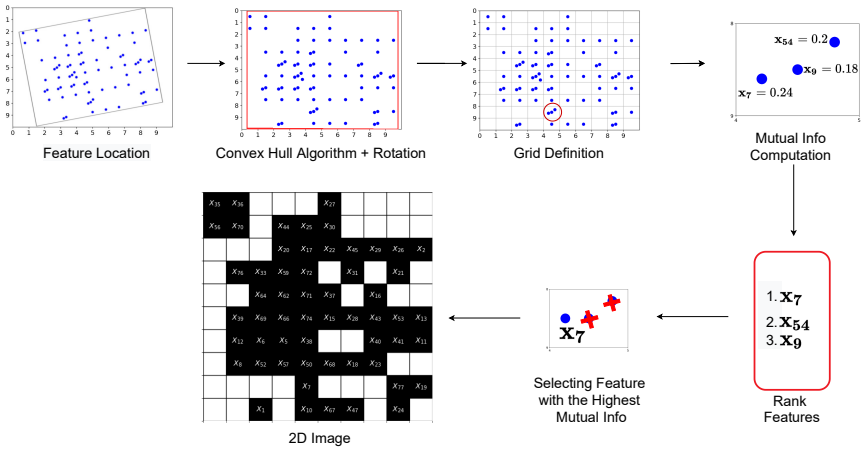
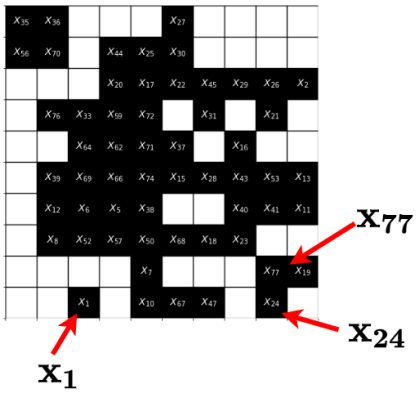
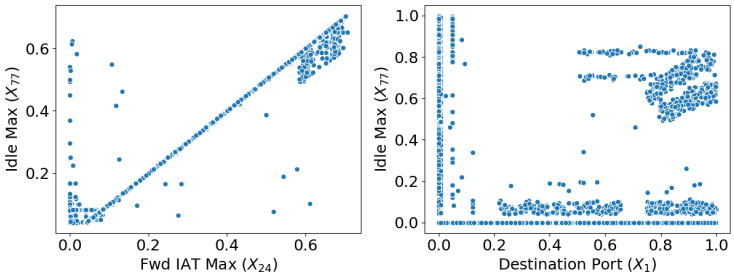


Figure 2: Mutual Info-based technique to manage intra-pixel feature collisions in the CICIDS17 dataset (see details in Section 5.1). x_7, x_{54} and x_9 denote the traffic characteristics: Fwd Packet Length Max, Avg Fwd Segment Size and Fwd Packet Length Mean, respectively.

sociated with neighbouring pixel frames. Figure 3a shows the image form of the traffic characteristics of CICIDS17. So, it allows us to build images of the network traffic, which depict a continuity in the grey-scale intensities at neighbouring pixels. For example, the traffic characteristics Idle Max (X_{77}) and Fwd IAT Max (X_{24}), that are contiguous on the image grid (since they are assigned to close pixel frames), commonly measure similar values on the data flows of CICIDS17 (as shown in Figure 3b). On the other hand, Idle Max (X_{77}) and Destination Port (X_1), that are assigned to distant pixel frames on the image grid, measure dissimilar values on the same data flows (as shown in Figure 3c). This data continuity in the imagery representation of data flows may be helpful for the use of convolution layers in both the imagery data augmentation performed through an ACGAN and the image classification performed through a 2D CNN architecture. To further explain the presence of the pixel continuity phenomenon with the adopted image transformation, let us consider Figure 4 that shows the 2D images of both a normal network flow (Figures 4a and 4b) and an attack network flow (Figures 4c and 4d). Figures 4a and 4c show the grey-scale images of the selected samples as they are built through the image



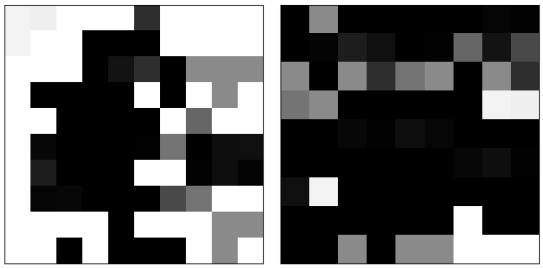
(a) Image representation



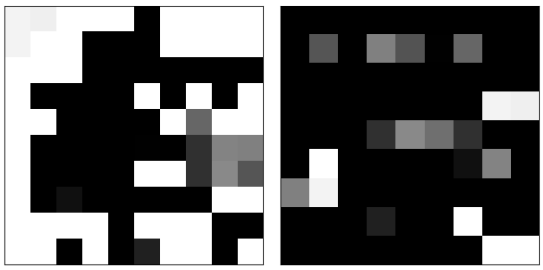
(b) Contiguous characteristics (c) Distant characteristics

Figure 3: The image representation of the traffic characteristics of dataset CICIDS17 (Figure 3a). The training samples of CICIDS17 are plotted along the contiguous traffic characteristics Idle Max (X_{77}) and Fwd IAT Max (X_{24}), respectively (Figure 3b), as well as along the distant traffic characteristics Idle Max (X_{77}) and Destination Port (X_1), respectively (Figure 3c). All the features are scaled between 0 and 1.

encoding step illustrated above. Figures 4b and 4d show the grey-scale images of the same samples, which are built by the naive method. The naive method assigns consecutive sample feature values to consecutive pixel frames by proceeding from the left to the right and from the top to the bottom of the 2D grid. We note that a progressive gradation of grey continuity is evident on the neighbour pixels of the images built by MAGNETO, while a salt and pepper grey distribution emerges in the images built with the naive method.



(a) Normal (MAGNETO) (b) Normal (naive method)



(c) Attack (MAGNETO) (d) Attack (naive method)

Figure 4: Image representations of both a normal sample and an attack sample collected in dataset CICIDS17 (see details in Section 5.1). Figures 4a and 4c show the 2D grey-scale images of the normal and the attack sample, respectively, as they are built by MAGNETO. Figures 4b and 4d show the the 2D grey-scale images of the normal and the attack sample, respectively, as they are built by the naive method.

3.2. GAN-based imagery data augmentation

In the second step, we train an Auxiliary Classifier GAN (ACGAN) [69] to generate synthetic images of artificial attacks and achieve the balance of the training set.

GANs are generative deep modelling approaches, originally formulated to work in the unsupervised manner [16], which have been recently used for generating synthetic images in computer vision applications [17]. They allow the construction of synthetic data that are similar to the training repository of real data. Based upon the seminal theory illustrated in [16], a GAN architecture is a two-player minimax game, in which the generator network (G) competes

against the adversary discriminator network (D). The training stage proceeds by alternately updating the parameters of both G and D, optimising the objective function. G takes noise samples as input and uses them to generate new examples of the data. D takes the fake images created by G and the real training samples as input and tries to distinguish real and fake samples.

The recent literature describes several families of GANs, both unsupervised (as in the original formulation) and supervised [73]. In this study we consider a supervised GAN – Auxiliary Classifier GAN (ACGAN) [69] – with convolution layers.[6] We opt for a supervised GAN as supervision allows us to take advantage of labels to jointly learn an inference mechanism to separate samples in different classes. This contributes to improving the quality of the data augmentation. In particular, the ACGAN architecture uses labels to directly apply cross-entropy loss to discriminator D, giving the discriminator an additional role in the classification. Specifically, the objective function optimised by the two-player minimax game of ACGAN is defined as follows:

$$min_{\mathsf{G}} max_{D} V(\mathsf{G},\mathsf{D}) = E_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log\mathsf{D}(\mathbf{x}|y)] + E_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[log(1-\mathsf{D}(\mathsf{G}(\mathbf{z}|y))), \quad (1)$$

where:

- $E_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log\mathsf{D}(\mathbf{x}|y)]$ is the log probability that D predicts the real image $\mathbf{x}$ as genuine, given the label y;
- $E_{\mathbf{z}\sim p_{\mathbf{z}}(\mathbf{z})}[log(1-\mathsf{D}(\mathsf{G}(\mathbf{z}|y)))$ is the log probability that D predicts the G's generated data as fake, given the label y and with the latent variable $\mathbf{z}$ randomly drawn from prior $p_{\mathbf{z}}(\mathbf{z})$.

[6] In principle, any supervised GAN architecture, e.g. Conditional Generative Adversarial Nets (CGANs), [74] or even an unsupervised GAN may be used in alternative to ACGANs. We select the ACGAN architecture by accounting for the conclusions recently drawn in [75], which prove that this architecture is able to generate better quality and globally coherent images than several GAN competitors.

In addition, the output of D includes both the probability that the image is real and its class label. Finally, the convolution layers allow us to take advantage of the spatial continuity in the imagery pixels. The implementation details of the architecture are reported in Section 4.

In the algorithm, we first process the imagery training set $(\mathbf{T^{2D}}, \mathbf{Y})$ to train the ACGAN architecture. Subsequently, we use the generator of the trained ACGAN to construct a number of artificial images with the "attack" class balancing the number of intrusions in the training set. Finally, we inject the artificial attacks into the original training set, thus building the augmented, balanced imagery training set $(\mathbf{T}^{\mathbf{2D}}_{\oplus}, \mathbf{Y}_{\oplus})$. Some examples of images of artificial attacks produced with the ACGAN are shown in Figure 5.

3.3. 2D CNN

In the third step we train a 2D Convolutional Neural Network (2D CNN) [76] to discriminate attacks from normal network flows, based on the image form of the samples.

The main advantage of training a 2D CNN architecture here is capturing possible spatial contiguity in images [77]. Since the produced image representation of the input data allows us to depict potential data patterns arising at neighbour pixels, a 2D CNN allows us to take advantage of the specific continuity behaviour embedded in the considered feature space grid inside the input region of the training data.

The 2D CNN implemented in MAGNETO is designed, according to the theory reported in [76], as a special kind of a fully connected feed-forward neural network, introducing local filters (convolution) and weight sharing.

Every convolutional layer convolves a set of filters that are replicated along the whole input to process small local parts of the input [76]. In particular, given an image $\mathbf{x}$, the k-th feature map at location (i, j) in a given convolutional l-th layer is determined by the weight matrix $\mathbf{W}^l_k$ and the bias vector $\mathbf{b}^l_k$ of the k-th filter on the l-th layer with a non-linear activation function $\sigma()$, such that:

$$h^l_{i,j,k} = \sigma(\mathbf{W}^l_k * \mathbf{x}^l_{i,j} + \mathbf{b}^l_k), \tag{2}$$

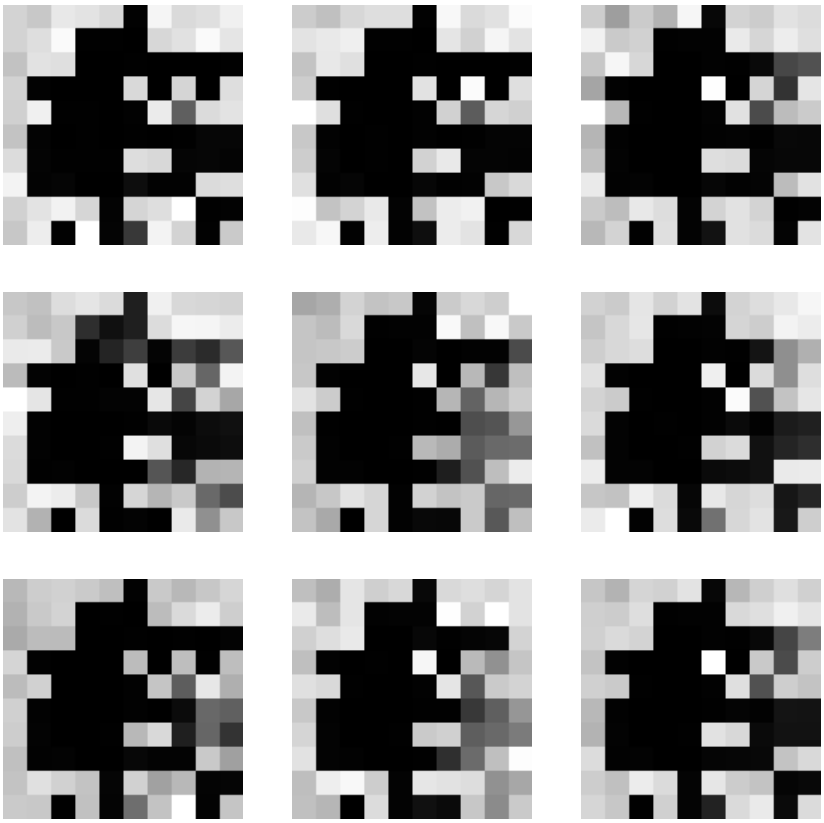


Figure 5: Images of artificial attacks generated through the generator G of the ACGAN that is trained on the imagery training set of CICIDS17.

where $\mathbf{x}_{i,j}^{l}$ is the input patch centered at location (i, j) of the l-th layer and $*$ represents the convolution function. The kernel $\mathbf{W}_{k}^{l}$ is shared for each possible location (i, j), thus reducing the model complexity and making the network easier to train. The theory in [76] also describes the use of pooling as a means to generate a lower resolution version of a convolutional layer, by offering better translation invariance. However, recent studies have highlighted that including explicit pooling operations does not always improve the performance of CNNs [78]. Based upon these conclusions, here we leave out pooling layers. The

implementation details of the designed 2D CNN are reported in Section 4.

In short, in this step we process the augmented imagery training set $(\mathbf{T}_{\oplus}^{2D}, Y_{\oplus})$ to train the 2D CNN architecture. The higher layers of this architecture use broader filters, which work on lower resolution inputs to process more complex parts of the input. The output layer uses softmax as the activation function [17], to deal with the classification task and discriminate the images of attacks from the images of normal network flows.

4. Implementation details

We have implemented both the pre-processing step and the three steps – image encoding, GAN-based data augmentation and 2D CNN – of the MAGNETO methodology in Python 3.6. The source code is available online.[7] The deep neural network architectures have been developed in Keras 2.3.1[8] – a high-level neural network API with TensorFlow 2.2[9] as the back-end.

The pre-processing step includes the operation to scale the input numeric features, using the Min-Max scale (as it is implemented in the Scikit-learn 0.22.2 library[10]). This operation is done to process features with values in comparable ranges. In addition, pre-processing includes the implementation of the one-hot-encoder mapping – a transformation commonly used in the intrusion detection literature [20, 21, 35] – to transform input categorical features into numerical features. Finally, it integrates an autoencoder to condense large one-hot-encoding input vectors that may introduce data sparseness in the input data. Autoencoders [79] are unsupervised neural networks that consist of both an encoder – mapping the input to a hidden code in a bottleneck layer – and a decoder – producing the reconstructed input from the hidden code. The output

[7] https://github.com/Kyanji/MAGNETO

[8] https://keras.io/

[9] https://www.tensorflow.org/

[10] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

of the bottleneck layer is used to obtain a non-linear representation of the input with reduced dimensionality.[11]

The autoencoder architecture implemented in MAGNETO comprises 3 fully-connected (FC) layers of $128 \times 64 \times 128$ 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 ($ReLu$) [83] is selected as the activation function for each hidden layer, while for the last layer the *Linear* activation function is used. In the pre-processing step, implemented for MAGNETO, we scale the output of the encoder with the Min-Max scaler and process it in place of the original input in the subsequent steps.

As regards the first step, the image encoding technique is implemented to convert the (pre-processed) input data from the 1D original feature vector to the 2D image form. By default, the imagery form of the data is produced with a 10×10 grid. In any case, we perform the experiments, illustrated in Section 5.3.3, to analyse the sensitivity of the performance of MAGNETO to the set-up of the image size.

In the second step, the ACGAN architecture is implemented following the description in [84]. [12]. In the generator, the Conv2DTranspose layers are alternated with Batch Normalization layers to stabilize learning, by normalizing the input to have zero mean and unit variance. Batch Normalization is not used in the discriminator. The rectified linear unit ($ReLu$) [83] is used as the activation function for each layer of the generator, except for the output layer, where the sigmoid activation is used. The LeakyReLu function [85] is used for the discriminator. The generator learns to generate fake images from a 100-dim input random noise vector with uniform distribution. The gradient-based optimisation is performed using the RMSprop update rule [86]. Finally, the auxiliary

[11] The use of autoencoders for dimensionality reduction has been widely investigated in the cybersecurity literature [80, 81, 82].

[12] The original code of the ACGAN implementation described in [84] can be downloaded at `https://github.com/PacktPublishing/Advanced-Deep-Learning-with-Keras/tree/master/chapter5-improved-gan`

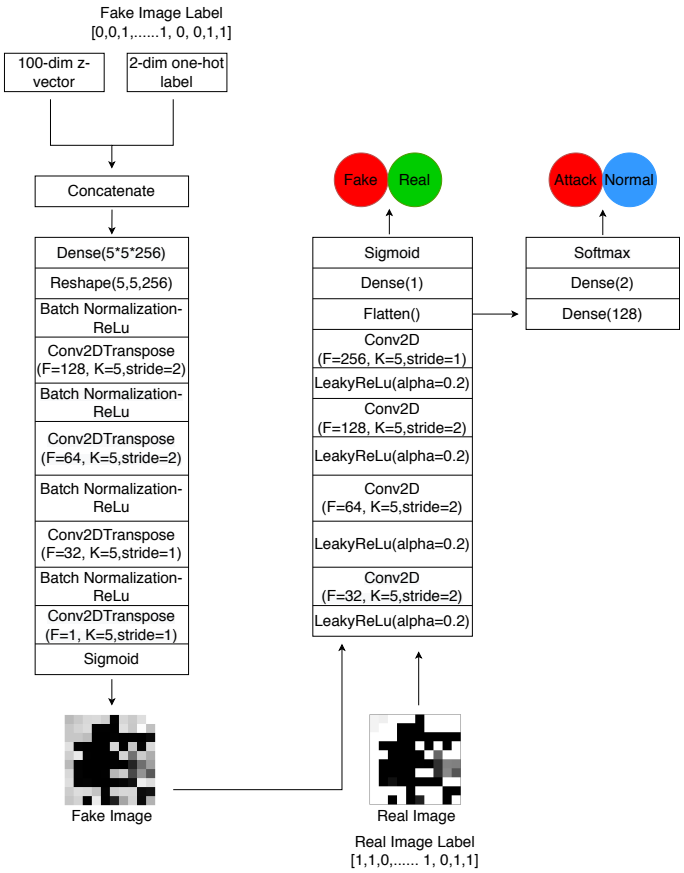


Figure 6: The ACGAN architecture of MAGNETO. For each convolutional layer, the number of filters (F), the kernel size (K) and the stride size are reported in brackets. For each dense layer, the number of neurons is reported in brackets.

module, added to perform the classification, comprises two Dense layers and a final softmax layer for the classification. This ACGAN updates two distinct loss functions: (1) the binary cross-entropy used to train the discriminator and estimate the probability that the input image is real and (2) the categorical cross-entropy predicting the class label of the image, respectively. The details of the configuration of this ACGAN architecture are reported in Figure 6.

As regards the third step, a 2D CNN architecture is implemented with three Convolutional layers, two Dropout layers and three Fully-Connected (FC) layers.

Table 2: Hyper-parameter search space for both autoencoder and 2D CNN architectures.

	autoencoders	classifier
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, 0.001]	[0.0001, 0.001]
dropout	[0,1]	[0,1]
kernel size	-	[2,4]

The network takes a 2D imagery training set as input and predicts a Bernoulli probability. The rectified linear unit ($ReLu$) [83] is selected as the activation function for each layer. This decision is motivated by the recent literature [38], which has proved that the best results are commonly achieved by using this activation function. Finally, the binary-cross entropy is used as the loss function. The details of the configuration of this 2D CNN architecture are reported in Figure 7. We note that the described 2D CNN architecture does not include any pooling operations. This decision follows the conclusions drawn in [78], which have already highlighted that including explicit pooling operations does not always improve the performance of CNNs.

Final details concern the estimation of the hyper-parameters used in both the autoencoders and the 2D CNN architectures. For each dataset, an automatic hyper-parameter optimisation is conducted using the tree-structured Parzen estimator algorithm, as implemented in the Hyperopt library [87]. This hyper-parameter optimisation is done by using 20% of the entire training as a validation set, according to the Pareto Principle [88]. In particular, we randomly select the validation set with the stratified sampling procedure [89]. Therefore, we automatically choose the hyper-parameter configuration that achieves the best validation loss. The values of the hyper-parameters automatically explored with the tree-structured Parzen estimator are reported in Table 2. In both these networks, we perform the gradient-based optimisation using the Adam update rule [90]. We initialise the weights following the Xavier scheme. Finally, we set

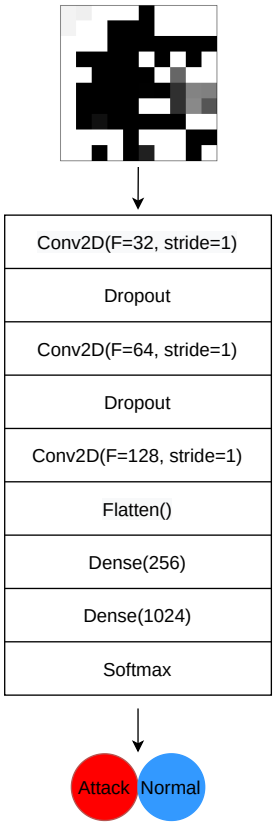


Figure 7: The 2D CNN architecture of MAGNETO. For each convolutional layer, the number of filters (F) and the stride size are reported in brackets. For each dense layer, the number of neurons is reported in brackets.

the maximum number of epochs equal to 150 by retaining the best models and using an early stopping approach that achieves the lowest loss on the validation set (the same set used for the hyper-parameter optimisation).

5. Empirical evaluation and discussion

For this empirical evaluation, we use four benchmark datasets, which are commonly considered in the network intrusion detection literature (Section 5.1). Each dataset includes both a labelled training set – processed to learn the intrusion detection model – and a testing set – considered to evaluate the intrusion

detection ability of the trained model (Section 5.2). The presentation of the empirical results is organised as follows. We start this study by analysing the effectiveness of the traffic image encoding technique, as it is coupled with the proposed 2D CNN architecture (Section 5.3). We proceed to explore the performance of the proposed imagery data augmentation step in the presence of the intrusion imbalance (Section 5.4).

We complete the study by presenting a brief discussion of additional recent evaluation results already reported in the recent intrusion detection literature, by processing the datasets also considered in this study (see Section 5.5).

5.1. Dataset description

We consider four benchmark intrusion detection datasets, that is, KDDCUP99,[13] UNSW-NB15[14], CICIDS17[15] and AAGM17.[16]

KDDCUP99 is commonly used for the evaluation of intrusion detection systems also in recent studies [91, 33, 31]. 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. We consider 10%KDDCUP99Train for the learning stage, while we use the entire testing set, denoted as KDDCUP99Test, for the evaluation stage.[17] This experimental scenario, with both 10%KDDCUP99Train and KDDCUP99Test, is often used in the literature (e.g. [36, 24]).

UNSW-NB15 is a hybrid dataset that includes the realistic modern normal activities and the synthetic contemporary attack behaviour extracted from network traffic monitored in 2015 [92]. It consists of one training set and one

[13] `http://kdd.ics.uci.edu//databases//kddcup99//kddcup99.html`

[14] `https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/`

[15] `https://www.unb.ca/cic/datasets/ids-2017.html`

[16] `https://www.unb.ca/cic/datasets/android-adware.html`

[17] 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`

testing set, which have recently been used in the evaluation of various intrusion detection methodologies [21, 93, 94, 44]. The attack records of this dataset are classified into nine families: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms.f

CICIDS17 is generally used in the evaluation of anomaly detection methodologies with the training performed on the first day [95, 96]. However, a few recent studies consider these data also in the evaluation of classification methodologies as we do in this paper [21, 97, 98, 99, 54]. The dataset comprises six attack families: Brute Force Attack, Heartbleed Attack, BotNet, DoS/DDoS Attack, Web Attack and Infiltration Attack. In our experimental study we consider the training and testing sets of CICIDS17 built according to the strategy described in [21]. 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. For the creation of both the training and testing set we use the stratified random sampling preserving 80% normal flows vs 20% attacks as in the original log.

AAGM17 contains traffic data captured from Android applications – malware, adware and benign apps. The traffic data are obtained by installing Android apps on real (NEXUS 5) smartphones in a semi-automated manner [100]. After running the apps, the generated traffic is captured with CICFlowMeter and transformed into samples divided into two classes (malicious and normal). 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 the original dataset, the number of normal apps was higher than the number of malicious apps (80% normal vs 20%) attack. This distribution is preserved in the sample training and testing sets prepared for this study.

A summary of the features of the datasets considered in this experimental investigation is reported in Table 3. We note that the malicious traffic is imbalanced in both CICIDS17 and AAGM17 datasets that both consist of 80% normal flows and 20% attacks. KDDCUP99 and UNSW-NB15 include cate-

gorical features that require numeric transformation with the one-hot-encoding technique. In these datasets the autoencoder architecture, described in Section 4, is used to reduce the high dimensionality and the data sparseness possibly introduced with one-hot-encoding.

In addition, KDDCUP99, UNSW-NB15 and CICIDS17 collect various families of attacks occurring in the flow based between hosts, while AAGM17 focuses on malicious families targeting Android platforms. In this study we investigate the binary classification problem. So, all the datasets are divided into two classes: normal and attack. All the attack categories contained in each dataset are grouped under a unique class label. However, few studies in cybersecurity have recently investigated how to classify the attack family. The classification of the attack families is investigated in [101, 102, 103, 104] for KDDCUP99, [105, 106, 107, 108] for UNSW-NB15 and [109] for CICIDS17. Moreover, some studies focus on detecting only one type of attack, i.e. botnet attacks [110, 111, 112]. Although our study investigates a simpler problem (discriminating attacks from normal data flows and postponing the attack family classification to future investigation), we have analyzed the performance of the proposed methodology on datasets comprising different typologies and number of attack families.

5.2. Experimental setting

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 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 flows tested. All these metrics are computed on the testing set of

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

		Dataset			
		KDDCUP99	**UNSW-NB15**	**CICIDS2017**	**AAGM17**
Features	Total	**42**	**43**	**79**	**80**
	Binary	6	2	18	3
	Categorical	3	3	-	-
	Numerical	32	37	60	76
	Class	1	1	1	1
Attack families	Total	**4**	**9**	**6**	**10**
Training set	Total	**494021**	**82332**	**100000**	**100000**
	Normal flows	97278 (19.7%)	37000 (44.9%)	80000 (80%)	80000 (80%)
	Intrusions	396743 (80.3%)	45332 (55.1%)	20000 (20%)	20000 (20%)
Testing set	Total	**311029**	**175341**	**900000**	**100000**
	Normal flows	60593 (19.5%)	56000 (31.9%)	720000 (80%)	80000 (80%)
	Intrusions	250436 (80.5%)	119341 (68.1%)	180000 (20%)	20000 (20%)

the considered dataset. In every experiment each testing sample is represented, similarly to a training sample, such as a 2D image. This representation is computed by applying the image encode that has been learned in the training stage.

The efficiency performance is evaluated with the computation time spent training the intrusion detection model. The computation time is collected on a Linux machine with an Intel(R) Core(TM) i7-9700F CPU @ 3.00GHz and 32GB RAM. All the experiments are executed on a single GeForce RTX 2080. The computation Time is measured in minutes.

5.3. Image encoding and 2D CNN analysis

The study of the effectiveness of both the image encoding step and the 2D CNN architecture of MAGNETO is conducted on the four datasets illustrated

in Section 5.1. It is performed by considering MAGNETO^{-GAN} – the configuration of MAGNETO that performs the training stage, leaving-out the imagery data augmentation step. As MAGNETO^{-GAN} roughly reproduces the pipeline of DeepInsight [18] by introducing: (i) a new Mutual Info-based intra-pixel feature collision technique and (ii) a different light 2D CNN architecture, this study focuses on assessing the viability of these two novel components. In addition, we evaluate the sensitivity of the performance of MAGNETO^{-GAN} to the imaging size.

5.3.1. Intra-pixel feature collision

We start this study evaluating the accuracy of the 2D CNN architecture illustrated in Section 3 with respect to the intra-pixel collision technique implemented in the image encoding step. To this aim, we compare the performance of:

- MAGNETO^{-GAN} which manages each group of traffic features that collide at the same pixel, by selecting the one with the highest Mutual Info. This configuration assigns only the selected feature to the collision pixel filtering out the un-selected features (see details in Section 3).
- MAGNETO$^{-GAN}_{+AVG}$ which integrates the intra-pixel collision technique of DeepInsight described in [18]. This configuration replaces each group of traffic features that collide on a pixel with an artificial, new feature that is built by applying the average operator. It assigns the new artificial feature – computed as the average of the features that enter into collision on the same pixel frame – to the colliding pixel in place of the features that collide on the pixel, while all the intra-pixel colliding features are filtered-out.

Table 4 reports the overall accuracy performance of both MAGNETO^{-GAN} and MAGNETO$^{-GAN}_{+AVG}$. Since these configurations train the light 2D CNN architecture proposed in this paper (see the implementation details in Section 4), this comparative analysis allows us to study properly the consequence of the

intra-pixel collision technique in the proposed pipeline. Collected results confirm that MAGNETO^{-GAN} commonly gains accuracy by using Mutual Info to manage intra-pixel collisions. This conclusion can be equally drawn from the analysis of both OA and F1 in all the datasets. However, we note that F1 is lower than OA in both CICIDS17 and AAGM17. This happens since the attack class (the positive class of our evaluation) is imbalanced in these two datasets. However, in this imbalanced scenario, F1 – combining both precision and recall for the minority attack class – is a better performance measure to expose the "best" class distribution [113]. Additional considerations concern the difference of the performance achieved by the tested methods on the analysed datasets. The accuracy performance is higher in KDDCUP99 and CICIDS17 than in UNSW-NB15 and AAGM17. This conclusion is equally drawn by analysing both OA and F1 and it is independent of the adopted collision management technique. We believe that this behaviour may depend on the variety of attack families in the considered datasets. In fact, KDDCUP99 and CICIDS17 collect attacks from 4 families and 6 families, respectively. UNSW-NB15 and AAGM17 collect attacks from 9 families and 10 families, respectively. The more varied the attack profiles, the more complex the intrusion detection task. This analysis contributes to highlighting that a crucial future direction of this work is the systematic investigation of how to account for the attack family information in a multi-class setting, in order to improve the stability of the performance of the proposed approach on the variety of attack families. In any case, this analysis already provides the evidence of the worth of our idea that opting for a supervised approach, that accounts for the class information, may improve the quality of the encoded traffic images and, consequently, the robustness of the intrusion detection model finally trained with these images.

Further considerations are formulated to explain the fact that UNSW-NB15 is more sensitive than the remaining datasets to the adopted feature collision management mechanism. Similar to KDDCUP99, this dataset contains categorical characteristics (e.g. protocol and service) managed during the preprocessing through the combination of one-hot-encoding and the autoencoder.

On the other hand, UNSW-NB15 comprises attacks from 9 families, while KDDCUP99 comprises attacks from 4 families only. So, we maintain that the numeric transformation of categorical characteristics introduces a bias in the image representation (as the data mapped to the image are different from the original ones). The greater the variety of attacks, the greater the bias (as observed by comparing KDDCUP99 to UNSW-NB15). On the other hand, the bias is partially overcome by using the proposed supervised mechanism to manage collisions. This consideration requires further investigations in the future. For example, we may explore algorithms for supervised embedding to account for (multi-)class information (on the attack families) and improve the numeric transformation of the categorical data [114].

Final comments concern the low F1 that MAGNETO$^{-\text{GAN}}$ achieves on AAGM17 (even if it does better than MAGNETO$^{-\text{GAN}}_{+\text{AVG}}$). The classification task is very complex in this dataset due to the presence of the imbalance attack condition, coupled with the large variety of attack families. However, the scenario of this dataset is quite different from the other datasets. AAGM17 collects malicious families of attacks targeting Android platforms, while KDDCUP99, UNSW-NB15 and CICIDS17 collect families of attacks occurring in the flows between hosts.

5.3.2. 2D CNN

We proceed in this study by comparing the performance of the two 2D CNN architectures implemented in MAGNETO$^{-\text{GAN}}$ and DeepInsight [18], respectively. To this aim, we compare:

- The 2D CNN architecture of MAGNETO$^{-\text{GAN}}$, that is described in Section 4 with three convolution layers and three dense layers.
- The 2D CNN of DeepInsight [18], that is a more complex architecture that consists of two parallel CNN architectures, where each one comprises four convolutional layers.

Both these 2D CNN architectures are trained using the tree-structured Parzen

Table 4: Intra-pixel feature collision: overall accuracy performance of MAGNETO$^{-\text{GAN}}$ and MAGNETO$^{-\text{GAN}}_{+\text{AVG}}$ measured with OA and F1 on the testing sets of KDDCUP99, UNSW-NB15, CICIDS17 and AAGM. The best results are in bold.

Dataset	#Collisions	Algorithm	OA	F1
KDDCUP99Test	11	MAGNETO$^{-\text{GAN}}$	**93.29**	**95.66**
		MAGNETO$^{-\text{GAN}}_{+\text{AVG}}$	92.84	95.35
UNSW-NB15Test	7	MAGNETO$^{-\text{GAN}}$	**89.73**	**91.97**
		MAGNETO$^{-\text{GAN}}_{+\text{AVG}}$	73.89	83.87
CICIDS17Test	21	MAGNETO$^{-\text{GAN}}$	**98.49**	**96.28**
		MAGNETO$^{-\text{GAN}}_{+\text{AVG}}$	98.13	95.45
AAGM17Test	55	MAGNETO$^{-\text{GAN}}$	**88.03**	**66.79**
		MAGNETO$^{-\text{GAN}}_{+\text{AVG}}$	87.21	61.18

estimator algorithm for the automatic hyper-parameter optimisation. According to the description reported in [18], the 2D CNN architecture of DeepInsight is coupled with the average-based intra-pixel collision technique.

Table 5 reports the accuracy performance of both MAGNETO$^{-\text{GAN}}$ and DeepInsight. In addition, Figure 8 compares the computation times spent in minutes training the 2D CNNs of both MAGNETO$^{-\text{GAN}}$ and DeepInsight. The analysis of the accuracy performance reveals that the light 2D CNN architecture, designed for MAGNETO, learns a more accurate intrusion detection model than that implemented in DeepInsight. This confirms the conclusions drawn by Kwon et al. [35], who have verified that the addition of new layers to a CNN architecture does not guarantee gain in intrusion detection accuracy. On the other hand, the analysis of the computation times highlights that MAGNETO$^{-\text{GAN}}$ learns a more accurate intrusion detection model than DeepInsight, spending less time in training the 2D CNN architecture. This computation time performance depends on the high cost of a convolution layer, that is $\mathbf{O}(n_{l-1}s_l^2 n_l m_l^2)$ [115] with l – the index of a convolution layer, n_l – the number of filters at layer

l, s_l^2 – the size of the 2D filters at layer l and m_l^2 – the size of the output feature map at layer l. The 2D CNN trained in MAGNETO^{-GAN} is a light architecture with three convolution layers, while 2D CNN trained with DeepInsight [18] is a more complex architecture implementing two parallel CNNs with four convolutional layers per CNN. Although DeepInsight uses parallelisation to speed-up the training of parallel networks, it is slower than MAGNETO^{-GAN}. Finally, we note that the considerations reported in Section 5.3.1 on the complexity of the intrusion detection task in AAGM17 may explain the lower F1 measured by both MAGNETO^{-GAN} and DeepInsight on AAGM17 also in this experiment. In particular, our idea is that the lower F1 achieved in this dataset may depend on the high variety of attack families (i.e. 10 different attack families in AAGM) in the processed data. We believe that this condition increases the complexity of the task of detecting attacks without taking advantage of a multi-class characterisation of the attack families.

Table 5: Baseline analysis: overall accuracy performance of MAGNETO^{-GAN} measured with OA and F1 on the testing sets of KDDCUP99, UNSW-NB15, CICIDS17 and AAGM. The best results are in bold.

Dataset	Algorithm	OA	F1
KDDCUP99Test	MAGNETO^{-GAN}	**93.29**	**95.66**
	DeepInsight [18]	92.80	95.32
UNSW-NB15Test	MAGNETO^{-GAN}	**89.73**	**91.97**
	DeepInsight [18]	68.29	81.10
CICIDS17Test	MAGNETO^{-GAN}	**98.49**	**96.28**
	DeepInsight [18]	97.56	94.11
AAGM17Test	MAGNETO^{-GAN}	**88.03**	**66.79**
	DeepInsight [18]	83.90	56.24

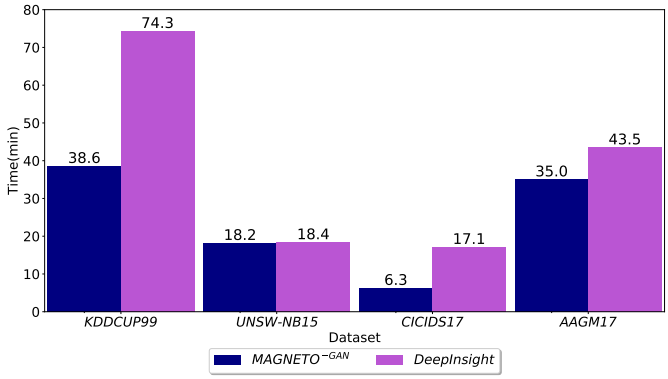
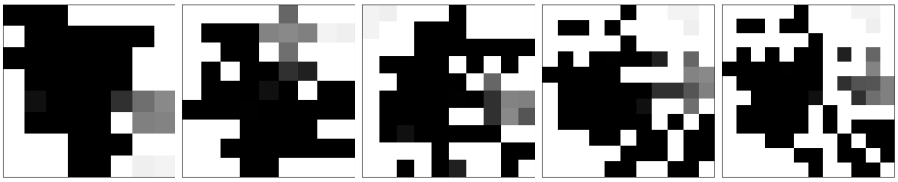


Figure 8: Computation time spent in minutes by both MAGNET0$^{-\text{GAN}}$ and DeepInsight training the intrusion detection models from the training sets of KDDCUP99, UNSW-NB15, CICIDS17 and AAGM.

5.3.3. Image encoding size

We complete the study of the effectiveness of the intrusion detection model trained through MAGNETO$^{-\text{GAN}}$, by exploring the sensitivity of its performance to the size of the image encoding. We perform this study using CICIDS17 and processing squared images of size varying among 8×8, 9×9, 10×10 (baseline), 11×11 and 12×12. The image size 9×9 delimits the minimum grid area that encloses a number of pixel frames that is greater than the number of traffic characteristics to map in the image grid. So, the baseline size 10×10 is the minimum size that guarantees the availability of an overhead of pixel frames in the grid, in order to reduce the collision risk. Figure 9 shows an example of an attack network flow encoded using the various tested image sizes. Table 6 reports the overall accuracy achieved by testing the trained intrusion detection models, as well as the time spent in minutes completing the training phase. As expected, the accuracy decreases with the image size 8×8, as this configuration does not guarantee the availability of a number of pixel frames greater than (or equal to) the number of traffic characteristics to map into the grid. On the other hand, the larger the size of the images, the lower the number of collisions, but the more time spent completing the training phase. In any case, decreasing the



(a) 8×8 (b) 9×9 (c) 10×10 (d) 11×11 (e) 12×12

Figure 9: An attack of CICIDS17Train encoded with the image size set equal to 8×8 (Figure 9a), 9×9 (Figure 9b), 10×10 (Figure 9c), 11×11 (Figure 9d) and 12×12 (Figure 9e).

number of collisions and, consequently, increasing the number of features finally selected for the deep learning does not lead to a gain in accuracy. This is a further confirmation of the effectiveness of the feature selection ability coupled with our proposal of image encoding of network flows. In general, the best accuracy is achieved with the baseline, that is, when the image size is set equal to 10×10.

Table 6: Image encoding size analysis: overall accuracy performance measured with OA and F1 and training computation time (in minutes) of MAGNETO$^{-\text{GAN}}$ by varying the size of the images in CICIDS17.

Size	# Collisions	OA	F1	Time
8×8	38	97.94	95.05	3.50
9×9	31	98.38	96.02	6.10
10×10	21	98.49	96.28	10.10
11×11	15	98.47	96.21	10.84
12×12	9	98.04	95.27	11.47

5.4. Data augmentation analysis

The study of the effectiveness of the imagery data augmentation step is performed by considering CICIDS17 and AAGM17, where the original traffic data are imbalanced. For this analysis, we consider the full configuration of

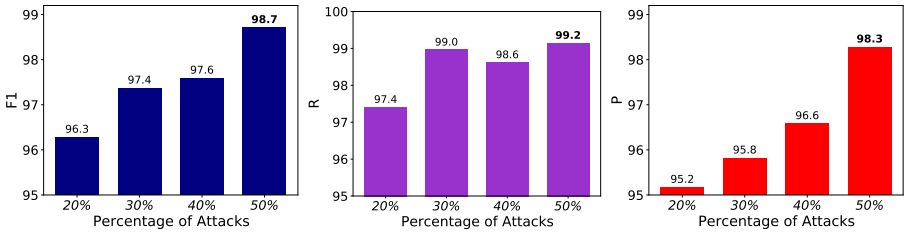
MAGNETO that cascades traffic data imagery encoding, GAN-based imagery data augmentation and 2D CNN. Our goal is to:

- Explore the sensitivity of the accuracy of MAGNETO along the balance size of the training set that can be achieved by producing new training images of artificial attacks.
- Analyse the sensitivity of the accuracy of MAGNETO along the imbalance size of the training set.
- Prove the effectiveness of our GAN-based imagery data augmentation step compared to that of state-of-the-art artificial data augmentation techniques, such as SMOTE [13] and ADASYN [14], that neglect the imagery format.
- Study the effectiveness of our idea of training a 2D CNN from the GAN-augmented training set instead of using the GAN discriminator for the final data flow classification.

5.4.1. Varying balance size

We analyse the performance of the GAN-based imagery data augmentation step of MAGNETO by varying the balance size. In particular, we increase the percentage of attacks from 20% (as in the original dataset) to 30%, 40% and 50% of the entire training set.

Figures 10a and 11a show the F1-score of MAGNETO, measured by varying the training balance size in CICIDS17 and AAGM17, respectively. In both datasets, the trend of the F1-score highlights that the intrusion detection model takes advantage of the imagery data augmentation step. In particular, the learned intrusion detection model progressively gains accuracy, as new images of artificial attacks are injected into the training set to achieve a balanced condition. This confirms the conclusions already drawn in [11] according to which the performance of a deep neural network can be improved by completing its training with balanced data.



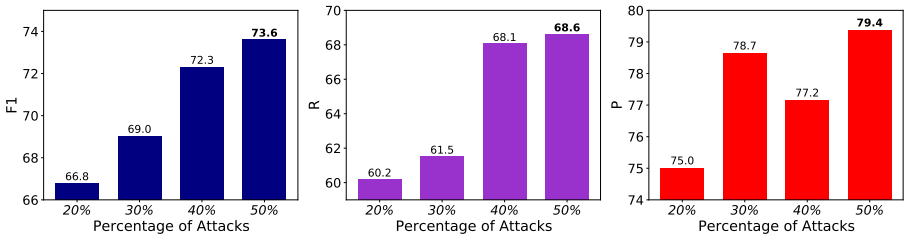
(a) F1 (b) R (c) P

Figure 10: Accuracy performance (axis Y), measured with F1-score – F1 – (Figure 11a), Recall – R – (Figure 10b) and Precision – P (Figure 10c) – of MAGNETO on CICIDS17Test, by varying the size of the balance (axis X) – the percentage of attacks on the entire training set – between 20% (original dataset) to 30%, 40% and 50% (augmented datasets).

To study this performance in depth, we also explore the trend of Recall (Figures 10b and 11b) and Precision (Figures 10c and 11c) in both the datasets. We note that the overall improvement detected with the F1-score is always coupled with the improvement of Recall – the ability to find all new attacks improves as the balance size increases within the training stage. In addition, the increase in Recall is commonly coupled with the increase in Precision – the ability to detect attacks correctly. The only exception is observed on AAGM17, when the balance size is set equal to 40%. In fact, this configuration of MAGNETO achieves a precision that is lower than that achieved at the balance size of 30%, although it is higher than that achieved on the baseline when no imagery data augmentation (balance size =20 %) has been performed.

5.4.2. Varying the imbalance size

We analyse the robustness of the proposed methodology to the size of the imbalance phenomenon by varying the amount of attack flows in the training set. For these experiments, we consider both CICDS2017 and AAGM17 datasets, which consist of 80% normal flows and 20% attacks. To stress the initial imbalance condition of these datasets, we consider all the training normal flows, and a sample of the attack flows randomly extracted from the training set. In this way, we generate three new training set trails consisting of: (1) 85% nor-



(a) F1 (b) R (c) P

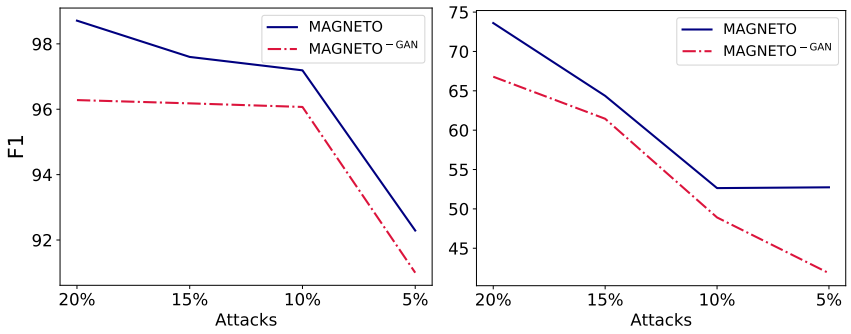
Figure 11: Accuracy performance (axis Y), measured with F1-score – F1 – (Figure 11a), Recall – R – (Figure 11b) and Precision – P – (Figure 11c) – of MAGNETO on AAGM2017Test, by varying the size of the balance in the final training set (axis X) – the percentage of attacks on the entire training set – between 20% (original dataset) to 30%, 40% and 50% (augmented datasets).

mal flows and 15% attacks, (2) 90% normal flows and 10% attack and (3) 95% normal flows and 5% attacks.

Figures 12a and 12b plot the F1-score measured when both MAGNETO and MAGNETO^{-GAN} are trained by varying the initial size of the attack imbalance in the training sets and tested on the original testing sets, CICDS2017Test and AAGM17Test, respectively. MAGNETO is run to increase the percentage of attacks to 50% of the entire training set. We note that diminishing the initial number of attacks (and consequently stressing the imbalance condition) leads to a decrease in the F1-score in both algorithms. However, MAGNETO continues outperforming MAGNETO^{-GAN} independently of the initial balance degree in the training set. In short, these results show that, although the data generation process loses effectiveness as less attack data are available for supervising both the generator and discriminator of the GAN architecture, the GAN-based imagery data augmentation step still gains accuracy, even when it starts from an extremely imbalance condition.

5.4.3. GAN vs SMOTE and ADASYS

We compare the performance of the GAN-based imagery data augmentation step implemented in MAGNETO to that of the state-of-the-art data augmenta-



(a) CICIDS17Test (b) AAGM17Test

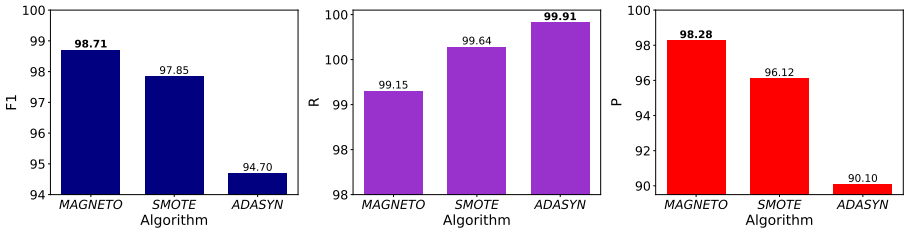
Figure 12: Accuracy (axis Y) measured by F1-score – F1 – of MAGNETO and MAGNETO^{-GAN} by varying the percentage of initial attacks in the training set (axis X). The F1-score is computed on CICIDS2017Test (Figure 12a) and AAGM17Test (Figure 12b), respectively.

tion techniques – SMOTE [13][18] and ADASYN [14].[19] To this aim, we define two variants of MAGNETO – denoted as SMOTE and ADASYN. They adopt the same image encoding of the network traffic data and the same 2D CNN architecture (described in Sections 3 and 4) as MAGNETO, while they replace GAN-based data augmentation with SMOTE and ADAYS, respectively.

Figures 13 and 14 show the F1-score, Recall and Precision of MAGNETO, SMOTE and ADASYN in CICIDS17 and AAGM17, respectively. The compared configurations are run with 50/50 balance in the training set. The results yielded confirm that the GAN-based imagery data augmentation step introduced in this paper outperforms the considered state-of-the-art data augmentation techniques in terms of both F1-score (Figures 13a and 14a) and Precision (Figures 13b and 14b). The behaviour of the compared configurations in terms of Recall (Figures 13b and 14b) requires additional considerations. In fact, MAGNETO, that

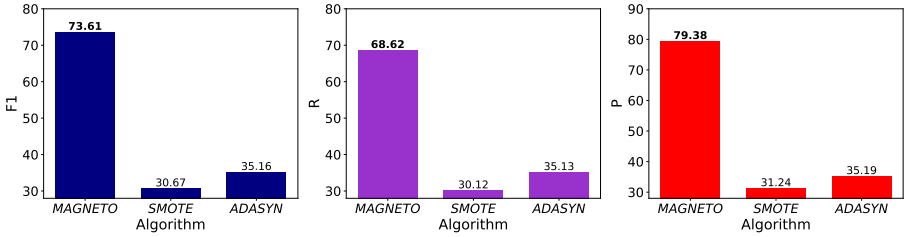
[18] We use the implementation of SMOTE in `https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html` with default parameter set-up.

[19] We use the implementation of ADASYN in `https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.ADASYN.html#imblearn.over_sampling.ADASYN` with a default parameter set-up.



(a) F1 (b) R (c) P

Figure 13: Accuracy (axis Y) – measured with F1-score – F1 – (Figure 13a), Recall – R – (Figure 13b) and Precision – P – (Figure 13c) – MAGNETO, SMOTE and ADASYS on CICIDS2017Test with 50/50 balance in the training set.

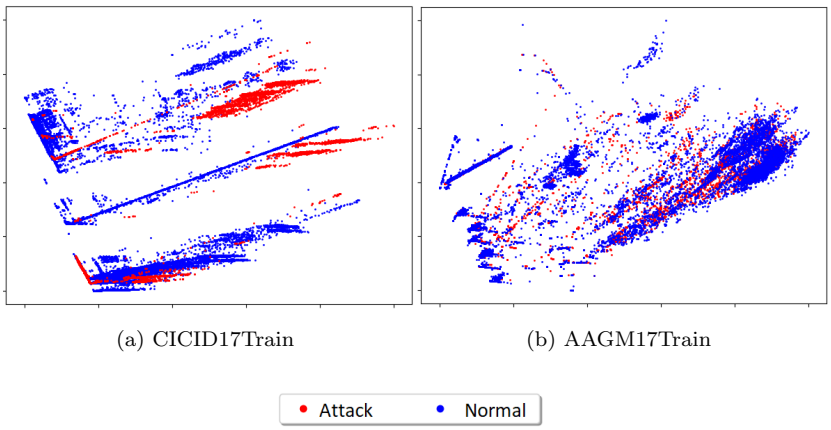


(a) F1 (b) R (c) P

Figure 14: Accuracy (axis Y) – measured with F1-score – F1 – (Figure 14a), Recall – R – (Figure 14b) and Precision – P – (Figure 14c) – of MAGNETO, SMOTE and ADASYS on AAGM2017Test with 50/50 balance in the training set.

achieves the highest Recall in AAGM17, exhibits a Recall drop in CICIDS17. However, examining in depth the overall performance of the three intrusion detection models learned on CICIDS17, we note that the observed Recall drop is negligible – it diminishes from 99.91 (ADASYN) and 99.64 (SMOTE) to 99.15 (MAGNETO). On the other hand, the Precision gain is significant – it increases from 90.10 (ADASYN) and 96.12 (SMOTE) to 98.12 (MAGNETO).

Additional considerations can be formulated to explain the poor performance of the data augmentation in SMOTE and ADASYS in AAGM17. To this purpose, we analyze the distribution of the training samples of both CICIDS17 and AAGM17, respectively. They are plotted in Figures 15a and 15b, respec-



(a) CICID17Train (b) AAGM17Train

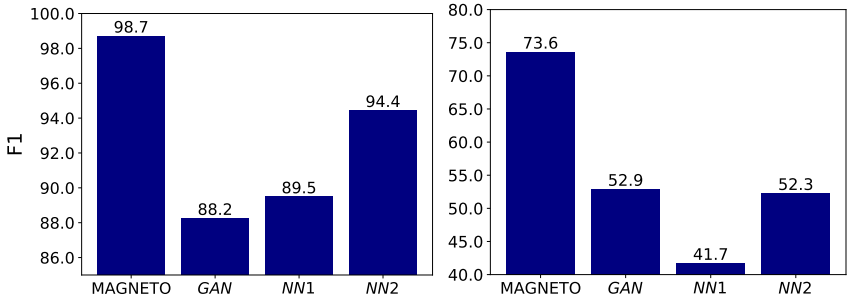
Figure 15: Training samples in CICIDS17Train (Figure 15a) and AAGM17 Test (Figure 15b). The top two principal components of the samples are plotted on the axes X and Y, respectively.

tively, by projecting the top two principal components of the training samples onto a 2D Cartesian plane. Figure 15a depicts dense clusters of attacks in CICIDS17Train. Figure 15b highlights salt and pepper attacks in AAGM17Train. By considering that both SMOTE and ADASYS base the data augmentation on neighbour samples, neighbourhoods of salt and pepper attacks can reasonably lead to the construction of non-representative artificial attacks in AAGM17.

5.4.4. GAN + 2D CNN analysis

To prove the effectiveness of cascading the GAN and the 2D CNN architectures, we compare the accuracy of MAGNETO to that of three baselines that use the GAN-trained model as a classifier. This analysis is possible as we account for the class supervision during the learning step for data augmentation by training a supervised GAN architecture (ACGAN). Specifically, we compare:

- MAGNETO—the data flow classification is performed with the 2D CNN model trained on the GAN-augmented training set.
- GAN— the data flow classification is performed with the discriminator function that has been trained within the GAN architecture, including also a softmax layer to classify the samples.



(a) CICID17Test (b) AAGM17Test

Figure 16: Accuracy measured by F1-score – F1 – of MAGNETO, GAN, NN1 and NN2 on CICIDS2017Test (Figure 16a) and AAGM17Test (Figure 16b), respectively

- NN1— the data flow classification is performed by using a deep neural network that replicates the architecture of the GAN discriminator and uses initial weights randomly initialized, following the Xavier scheme.

- NN2— the data flow classification is performed by training a deep neural network that replicates the architecture of the GAN discriminator and uses the weights learned with the GAN discriminator as initial weights of the model.

Figures 16a and 16b plot the F1-score achieved by the compared configurations on CICDS2017Test and AAGM17Test, respectively. The results confirm that the combination of GAN + 2D CNN adopted in MAGNETO is able to gain accuracy in both datasets compared to the architectures using only the GAN model. On the other hand, the poor performance obtained by considering the GAN architecture for the classification task can be attributed to the different task addressed when training the discriminator. In fact, when we train an AC-GAN architecture, we are able to use the class information also for learning a data flow classification function within the discriminator. However, in this context, the learning focus mainly rotates on learning a discriminator that achieves the best separation between the real samples and the generated ones [116].

5.5. Competitor analysis

We compare the accuracy performance achieved by MAGNETO to that of several competitors selected from the recent state-of-the-art literature. In particular, we consider the following competitors:

- 1D Convolutional Neural Network-based competitors (1D CNN): CNN-1D [44], CNN4 [38],MINDFUL [36] and THEODORA [54].
- 2D Convolutional Neural Network-based competitors (2D CNN): Grey-scale [20], [21] and RGB [21].
- Long Short-Term Memory Neural Network-based competitor (LSTM): BLSTM [29].
- Recurrent Neural Network-based competitor (RNN): BRNN [29].
- Deep Neural Network-based competitors (DNN): DNN 4 Layers [117], DNN3 [118], DNN4 [38], DBN [24] , A+DBN [24], MLP [44], WnD [25], MLP [25], SAE [25] and AIDA [26].
- GAN Network-based competitors (GAN): AnoGAN [62] [32], Efficient GAN [31], ALAD [32] and MAD-GAN [33].

Both the 2D CNN-based competitors and the GAN-based competitors are the nearest related to MAGNETO. In fact, the 2D CNN-based competitors, similarly to MAGNETO, experiment various 2D image encoding technique to transform network flows into imagery data and to train a 2D CNN architecture. In any case, they all use image encoding techniques that transform 1D vector data into 2D imagery data without accounting for patterns of inter-feature spatial continuity. On the contrary, MAGNETO adapts an image encoding technique, originally formulated for gene data, to transform network traffic data into images, depicting contiguity in data intensity across neighbour pixels. This allows MAGNETO to train deep learning architectures that are effectively able to take advantage of convolutions on continuous and contiguous data pixels. On the other hand, the GAN-based competitors use GAN architectures, but with a goal

that is different from data balance. In fact, they mainly perform generative adversarial learning for anomaly detection (i.e. to detect the samples of the minority class as anomalies). Finally, THEODORA is the other competitor that, similarly to MAGNETO, tries to increase the number of attacks in the training set. However, MAGNETO produces new artificial attacks, which augment the training set, while THEODORA assigns the label "attack" to normal training samples that are close to attacking ones.

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 collected results are reported in Table 7 for all the datasets. The results of MAGNETO^{-GAN} are collected on all the datasets. The results with MAGNETO are collected in CICIDS17 and AAGM17, as both these datasets present an imbalance condition. These results show that both MAGNETO and MAGNETO^{-GAN} outperform their competitors, including both the 2D CNN-based competitors (Grey-scale and RGB evaluated on UNSW-NB15 and CICIDS2017 in the reference studies), the GAN-based competitors (AnoGAN, Efficient GAN, ALAD and MAD-GAN [33], evaluated on KDDCUP99 in the reference study) and THEODORA (evaluated in all the datasets). The only exceptions are observed with UNSW-NB15. In this dataset, MAGNETO^{-GAN} is outperformed by WnD, MINDFUL, THEODORA and AIDA. However, we note that WnD uses embedding, while we use the one-hot-encoding of categorical data and subsequently the autoencoder. So, the difference in the performance may also be due to the pre-processing stage. On the other hand, MINDFUL, THEODORA and AIDA, that are at any rate outperformed by both MAGNETO and MAGNETO^{-GAN} in the remaining datasets, use richer information than MAGNETO. In particular, they enrich the vector of the original network traffic characteristics with new information, that is synthesised through autoencoders trained on normal data and attack data, separately. This result paves the way for future developments of MAGNETO, which aim at investigating how the quality of the training data can be improved by synthesising normal-based and/or attack-based information and injecting this information into the image encoding

Table 7: Competitor analysis: The accuracy metrics of the competitors are collected from the reference papers. "-" denotes that no value is reported in the reference paper.

Dataset	Category	Description	OA	F1
KDDCUP99	MAGNETO^{-GAN}	2D CNN	**93.29**	**95.66**
	MINDFUL [36]	1D CNN	92.49	95.13
	CNN4 [38]	1D CNN	92.47	-
	DNN4Layers [117]	DNN + Text-based encoding	93.00	95.50
	DNN-3 [118]	DNN	93.00	95.50
	DNN4 [38]	DNN	92.88	-
	DBN [24]	DNN	91.40	-
	A+DBN [24]	Autoencoder + DBN	92.10	-
	BLSTM [29]	LSTM	-	93.27
	BRNN [29]	RNN	-	91.82
	THEODORA [54]	1D CNN + Label re-assignment	92.97	95.46
	AIDA [26]	Autoencoder + MLP	92.36	95.04
	AnoGAN [62] [32]	GAN	-	88.65
	Efficient GAN [31]	GAN		93.72
	ALAD [32]	GAN	-	95.01
	MAD-GAN [33]	GAN	-	90.00
UNSW-NB15	MAGNETO^{-GAN}	2D-CNN	89.73	91.97
	MINDFUL [36]	1D CNN	**93.40**	**95.29**
	CNN-1D [44]	1D CNN	89.80	91.30
	Grey-scale[20][21]	2D CNN	80.00	84.00
	RGB [21]	2D CNN	83.00	86.50
	DNN4Layers [117]	DNN + Text-based encoding	76.50	90.10
	MLP [44]	DNN	86.60	88.90
	WnD [25]	DNN + Embedding	91.20	-
	MLP [25]	MLP	86.70	-
	SAE [25]	Autoencoder	88.20	-
	THEODORA [54]	1D CNN + Label re-assignment	92.47	94.75
	AIDA [26]	Autoencoder + MLP	90.54	92.71
CICIDS17	MAGNETO^{-GAN}	2D-CNN	98.49	96.28
	MAGNETO	2D-CNN+ GAN	**99.48**	**98.71**
	MINDFUL [36]	1D CNN	97.90	94.93
	Grey-scale [20][21]	2D-CNN	-	82.00
	RGB [21]	2D-CNN	-	89.00
	THEODORA [54]	1D CNN + Label re-assignment	98.03	95.25
	AIDA [26]	Autoencoder + MLP	94.50	85.80
AAGM17	MAGNETO^{-GAN}	2D-CNN	88.03	66.79
	MAGNETO	2D-CNN + GAN	**90.16**	**73.61**
	MINDFUL [36]	1D CNN	86.15	51.62
	THEODORA [54]	1D CNN + Label re-assignment	87.62	65.92
	AIDA [26]	Autoencoder + MLP	86.06	57.78

step.

6. Conclusions

Machine learning methods, formulated in the intrusion detection literature, are often not effective in dealing with the imbalance condition of attacks in

historical network traffic. In this study we address the network intrusion detection problem by proposing a multi-stage methodology, denoted as MAGNETO, that combines image encoding, GAN-based data augmentation and 2D CNN. In particular, we use an image encoding technique that accounts for patterns of inter-feature spatial continuity to map the 1D feature vector representation of the network flows into a 2D imaging representation of these data. 2D data are used to train both an ACGAN and a 2D CNN architecture. ACGAN is used to produce new images of artificial attacks dealing with imbalance and possibly simulating unforeseen network attacks. 2D CNN is trained on the augmented imagery training set as a CNN-based intrusion detection model.

We evaluate the effectiveness of the proposed methodology using four benchmark datasets that contain network flows collected in different years. The experimental analysis proves the viability of the multiple steps of the proposed methodology. In addition, it proves that MAGNETO gains accuracy compared to several, recently defined, state-of-the-art competitors, using deep learning and, in a few cases, image encoding of data, 2D CNNs and/or GANs.

As future work we plan to explore the effectiveness of the used image encoding method, by carrying out experiments using common CNN architectures like those based on ResNet, Inception or LeNet, as well as new variants of GAN models, such as Wasserstein Generative Adversarial Networks (WGAN)[119] or Bi-Directional Big GANs (BigBiGAN) [120]. In addition, we plan to explore how the RGB mapping, that was already explored in [21], can be used in combination with the image encoding technique used here as a base for an enhanced imaging representation of the network traffic. We also intend to explore the possibility of gaining accuracy by injecting additional information, expressing patterns of normal and/or attacking behaviour, into the image encoding step.

In addition, motivated by the increasing interest in intrusion detection systems able to recognise the attack family, we plan to extend this investigation to the classification of the intrusion families. Indeed, the evaluation of the performance of the proposed binary classification methodology, that we have conducted on datasets comprising different types and number of attack families,

has assessed that the proposed methodology can outperform state-of-the-art binary competitors. However, it has also achieved different results, depending on the variety of attacks in each dataset. In general, the more varied the attack profiles, the more complex the intrusion detection task and, hence, the lower the ability of a binary pattern to cover all attack variants simultaneously. To address this issue, we consider the possibility of formulating a multi-class extension of the proposed binary classification methodology, in order to classify the intrusion families. This would require the exploration of the imbalance issue also in relation to the presence of rare attack families within malicious traffic. In principle, the multi-class classification would be possible if we explored a combination strategy (e.g. one-versus-all or one-versus-one) [121] to frame the proposed binary methodology in the multi-class scenario. Alternatively, the multi-class classification could be performed by replacing the binary cross-entropy loss with the categorical cross-entropy loss in the deep neural network architectures. However, both solutions require further investigation.

Finally, the proposed methodology does not provide detailed information on the structure and characteristics of different attacks. Therefore, explainable artificial intelligence techniques (e.g. Grad-CAM [122] or Activation Maximization [123]) may be an additional research direction here. To this end, we plan to explore how the information provided by the explanation of image classifications may aid in identifying a description of the attack signature, by highlighting the traffic characteristics that are the most relevant for the classification of each attack family.

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), funding scheme PON RI 2014-2020, 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 wish to thank Lynn Rudd for her help in reading the manuscript.

References

[1] J. Jang-Jaccard, S. Nepal, A survey of emerging threats in cybersecurity, Journal of Computer and System Sciences 80 (5) (2014) 973 – 993, special Issue on Dependable and Secure Computing.

[2] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, W.-Y. Lin, Intrusion detection by machine learning: A review, Expert Systems with Applications 36 (10) (2009) 11994 – 12000.

[3] Akashdeep, I. Manzoor, N. Kumar, A feature reduced intrusion detection system using ann classifier, Expert Systems with Applications 88 (2017) 249 – 257.

[4] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, M. Marchetti, On the effectiveness of machine and deep learning for cyber security, in: 10th International Conference on Cyber Conflict, CyCon 2018, Tallinn, Estonia, May 29 - June 1, 2018, IEEE, 2018, pp. 371–390.

[5] D. S. Berman, A. L. Buczak, J. S. Chavis, C. L. Corbett, A survey of deep learning methods for cyber security, Information 10 (4) (2019) 1–35.

[6] D. Sovilj, P. Budnarain, S. Sanner, G. Salmon, M. Rao, A comparative evaluation of unsupervised deep architectures for intrusion detection in sequential data streams, Expert Systems with Applications 159 (2020) 113577.

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

[8] B. Dong, X. Wang, Comparison deep learning method to traditional methods using for network intrusion detection, in: 2016 8th IEEE International

Conference on Communication Software and Networks (ICCSN), IEEE, 2016, pp. 581–585.

[9] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, P. J. Kennedy, Training deep neural networks on imbalanced data sets, in: 2016 International Joint Conference on Neural Networks (IJCNN), 2016, pp. 4368–4374.

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

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

[12] D. A. Cieslak, N. V. Chawla, A. Striegel, Combating imbalance in network intrusion datasets, in: 2006 IEEE International Conference on Granular Computing, 2006, pp. 732–737.

[13] N. Chawla, K. Bowyer, L. Hall, W. Kegelmeyer, Smote: Synthetic minority over-sampling technique, J. Artif. Intell. Res. (JAIR) 16 (2002) 321–357.

[14] Haibo He, Yang Bai, E. A. Garcia, Shutao Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 2008, pp. 1322–1328.

[15] Z. Xu, D. Shen, T. Nie, Y. Kou, A hybrid sampling algorithm combining m-smote and enn based on random forest for medical imbalanced data, Journal of Biomedical Informatics (2020) 103465.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, MIT Press, Cambridge, MA, USA, 2014, p. 2672–2680.

[17] C. C. Aggarwal, Neural Networks and Deep Learning - A Textbook, Springer, 2018.

[18] A. Sharma, E. Vans, D. Shigemizu, K. Boroevich, T. Tsunoda, Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture, Scientific Reports 9 (11399) (2019) 1–7.

[19] J. Jam, C. Kendrick, K. Walker, V. Drouard, J. G.-S. Hsu, M. H. Yap, A comprehensive review of past and present image inpainting methods, Computer Vision and Image Understanding 203 (2021) 103147.

[20] Z. Li, Z. Qin, K. Huang, X. Yang, S. Ye, Intrusion detection using convolutional neural networks for representation learning, in: ICONIP, Springer International Publishing, 2017, pp. 858–866.

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

[22] H. Zhang, X. Yu, P. Ren, C. Luo, G. Min, Deep adversarial learning in intrusion detection: A data augmentation enhanced framework (2019). arXiv:1901.07949.

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

[24] Y. Li, R. Ma, R. Jiao, A hybrid malicious code detection method based on deep learning, in: International journal of security and its applications, Vol. 9, 2015, pp. 205–216.

[25] J. Yan, D. Jin, C. W. Lee, P. Liu, A comparative study of off-line deep learning based network intrusion detection, in: 10th International Conference on Ubiquitous and Future Networks, 2018, pp. 299–304.

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

[27] S. A. Althubiti, E. M. Jones, K. Roy, Lstm for anomaly-based network intrusion detection, in: 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), IEEE Computer Society, 2018, pp. 1–3.

[28] R. Vinayakumar, K. Soman, P. Poornachandran, Evaluation of recurrent neural network and its variants for intrusion detection system ids, Int. J. Inf. Syst. Model. Des. 8 (3) (2017) 43–63.

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

[30] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, IEEE Access 5 (2017) 21954–21961.

[31] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, V. R. Chandrasekhar, Efficient gan-based anomaly detection, CoRR abs/1802.06222 (2018) 1–13.

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

[33] L. Dan, C. Dacheng, J. Baihong, S. Lei, G. Jonathan, N. See-Kiong, Madgan: Multivariate anomaly detection for time series data with generative adversarial networks, in: Artificial Neural Networks and Machine Learning, 2019, pp. 703–716.

[34] J. Yang, T. Li, G. Liang, W. He, Y. Zhao, A simple recurrent unit model based intrusion detection system with DCGAN, IEEE Access 7 (2019) 83286–83296.

[35] D. Kwon, K. Natarajan, S. C. Suh, H. Kim, J. Kim, An empirical study on network anomaly detection using convolutional neural networks, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2018, pp. 1595–1598.

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

[37] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, L. Cui, Robust detection for network intrusion of industrial iot based on multi-cnn fusion, in: Measurement, Vol. 154, 2020, p. 107450.

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

[39] W. Xuan, G. You, Detection and diagnosis of pancreatic tumor using deep learning-based hierarchical convolutional neural network on the internet of medical things platform, Future Generation Computer Systems 111 (2020) 132 – 142.

[40] S. Gai, Z. Bao, New image denoising algorithm via improved deep convolutional neural network with perceptive loss, Expert Systems with Applications 138 (2019) 112815.

[41] S. Abdoli, P. Cardinal, A. L. Koerich], End-to-end environmental sound classification using a 1d convolutional neural network, Expert Systems with Applications 136 (2019) 252 – 263.

[42] D. Stephens, A. Smith, T. Redfern, A. Talbot, A. Lessnoff, K. Dempsey, Using three dimensional convolutional neural networks for denoising

echosounder point cloud data, Applied Computing and Geosciences 5 (2020) 100016.

[43] S. Alonso-Monsalve, A. L. Suárez-Cetrulo, A. Cervantes, D. Quintana, Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators, Expert Systems with Applications 149 (2020) 113250.

[44] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Shallow neural network with kernel approximation for prediction problems in highly demanding data networks, Expert Systems with Applications 124 (2019) 196 – 208.

[45] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, 2016, pp. 770–778.

[46] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2015, pp. 1–9.

[47] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, F. Iqbal, Malware classification with deep convolutional neural networks, in: 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1–5.

[48] R. Burks, K. A. Islam, Y. Lu, J. Li, Data augmentation with generative models for improved malware detection: A comparative study*, in: 2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), 2019, pp. 0660–0665.

[49] H. Lee, M. Park, J. Kim, Plankton classification on imbalanced large scale database via convolutional neural networks with transfer learning,

in: 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3713–3717.

[50] S. Pouyanfar, Y. Tao, A. Mohan, H. Tian, A. S. Kaseb, K. Gauen, R. Dailey, S. Aghajanzadeh, Y. Lu, S. Chen, M. Shyu, Dynamic sampling in convolutional neural networks for imbalanced data classification, in: 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), 2018, pp. 112–117.

[51] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, Li Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[52] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (2) (2020) 318–327.

[53] H. Zhang, L. Huang, C. Q. Wu, Z. Li, An effective convolutional neural network based on smote and gaussian mixture model for intrusion detection in imbalanced dataset, Computer Networks 177 (2020) 107315.

[54] G. Andresini, A. Appice, F. Caforio, D. Malerba, Improving cyber-threat detection by moving the boundary around the normal samples, in: Machine Intelligence and Big Data Analytics For Cybersecurity Applications, Studies in Computational Intelligence, 2021, pp. 105–127.

[55] H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, C. G. Schwarz, M. L. Senjem, J. L. Gunter, K. P. Andriole, M. Michalski, Medical image synthesis for data augmentation and anonymization using generative adversarial networks, in: Simulation and Synthesis in Medical Imaging, Springer International Publishing, Cham, 2018, pp. 1–11.

[56] M. Frid-Adar, I. Diamant, E. Klang, M. Amitai, J. Goldberger, H. Greenspan, Gan-based synthetic medical image augmentation for in-

creased cnn performance in liver lesion classification, Neurocomputing 321 (2018) 321 – 331.

[57] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, H. Greenspan, Synthetic data augmentation using gan for improved liver lesion classification, in: 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), 2018, pp. 289–293.

[58] Y. Yiming, Z. Tan, N. Su, A data augmentation strategy based on simulated samples for ship detection in rgb remote sensing images, ISPRS International Journal of Geo-Information 8 (2019) 276.

[59] Z. Cui, M. Zhang, Z. Cao, C. Cao, Image data augmentation for sar sensor via generative adversarial nets, IEEE Access 7 (2019) 42255–42268.

[60] S.-Y. Shin, Y.-W. Kang, Y.-G. Kim, Android-gan: Defending against android pattern attacks using multi-modal generative network as anomaly detector, Expert Systems with Applications 141 (2020) 112964.

[61] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, S.-K. Ng, Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks, in: Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series, Springer International Publishing, Cham, 2019, pp. 703–716.

[62] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, G. Langs, Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, in: Information Processing in Medical Imaging, Springer International Publishing, Cham, 2017, pp. 146–157.

[63] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples (2014). `arXiv:1412.6572`.

[64] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, D. Mukhopadhyay, Adversarial attacks and defences: A survey, ArXiv abs/1810.00069 (2018) 1–31.

[65] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 582–597.

[66] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, A. Swami, Practical black-box attacks against deep learning systems using adversarial examples, CoRR abs/1602.02697 (2016) 1–14.

[67] S. Shin, I. Lee, C. Choi, Anomaly dataset augmentation using the sequence generative models, in: 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 2019, pp. 1143–1148.

[68] Q. Wang, W. Guo, K. Zhang, A. G. Ororbia, X. Xing, X. Liu, C. L. Giles, Adversary resistant deep neural networks with an application to malware detection, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 1145–1153.

[69] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier GANs, in: Proceedings of the 34th International Conference on Machine Learning, Vol. 70 of Proceedings of Machine Learning Research, PMLR, International Convention Centre, Sydney, Australia, 2017, pp. 2642–2651.

[70] L. van der Maaten, G. Hinton, Viualizing data using t-sne, Journal of Machine Learning Research 9 (2008) 2579–2605.

[71] F. P. Preparata, S. J. Hong, Convex hulls of finite sets of points in two and three dimensions, Commun. ACM 20 (2) (1977) 87–93.

[72] J. Vergara, P. Estevez, A review of feature selection methods based on mutual information, Neural Computing and Applications 24 (2014) 175–186.

[73] Z. Wang, Q. She, T. E. Ward, Generative adversarial networks: A survey and taxonomy, CoRR abs/1906.01529 (2019) 1–41.

[74] M. Mirza, S. Osindero, Conditional generative adversarial nets, CoRR abs/1411.1784 (2014) 1–7. arXiv:1411.1784.

[75] K. Cheng, R. Tahir, L. K. Eric, M. Li, An analysis of generative adversarial networks and variants for image synthesis on MNIST dataset, Multim. Tools Appl. 79 (19-20) (2020) 13725–13752.

[76] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[77] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, 1998, pp. 2278–2324.

[78] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for simplicity: The all convolutional net, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings, 2015, pp. 1–14.

[79] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[80] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, A. Abuzneid, Features dimensionality reduction approaches for machine learning based network intrusion detection, Electronics 8 (2019) 322.

[81] Y. Zeng, H. Gu, W. Wei, Y. Guo, Deep-full-range : A deep learning based network encrypted traffic classification and intrusion detection framework, IEEE Access 7 (2019) 45182–45190.

[82] Y. N. Kunang, S. Nurmaini, D. Stiawan, A. Zarkasi, Firdaus, Jasmir, Automatic features extraction using autoencoder in intrusion detection system, in: 2018 International Conference on Electrical Engineering and Computer Science (ICECOS), 2018, pp. 219–224.

[83] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks., in: AISTATS, JMLR.org, 2011, pp. 315–323.

[84] R. Atienza, Advanced Deep Learning with Keras: Apply Deep Learning Techniques, Autoencoders, GANs, Variational Autoencoders, Deep Reinforcement Learning, Policy Gradients, and More, Packt Publishing, 2018.

[85] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: in ICML Workshop on Deep Learning for Audio, Speech and Language Processing, 2013, pp. 1–6.

[86] T. Tieleman, G. Hinton, Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude, COURSERA: Neural Networks for Machine Learning (2012).

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

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

[89] V. L. Parsons, Stratified sampling, in: Wiley StatsRef: Statistics Reference Online, American Cancer Society, 2017, pp. 1–11.

[90] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: ICLR, 2014, pp. 1–15.

[91] M. Labonne, A. Olivereau, B. Polve, D. Zeghlache, A cascade-structured meta-specialists approach for neural network-based intrusion detection, 16th Annual Consumer Communications & Networking Conference (2019) 1–6.

[92] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: Military Communications and Information Systems Conference, 2015, pp. 1–6.

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

[94] Y. Yang, K. Zheng, C. Wu, X. Niu, Y. Yang, Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks, Applied Sciences 9 (2) (2019) 238.

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

[96] P. Angelo, A. Costa Drummond, Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling, Security and Privacy 1 (4) (2018) 1–13.

[97] A. Binbusayyis, T. Vaiyapuri, Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach, IEEE Access 7 (2019) 106495–106513.

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

[99] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, H. Janicke, 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), IEEE, 2019, pp. 228–233.

[100] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah, A. A. Ghorbani, Towards a network-based framework for android malware detection and characterization., in: PST, IEEE Computer Society, 2017, pp. 233–234.

[101] Y. Yu, N. Bian, An intrusion detection method using few-shot learning, IEEE Access 8 (2020) 49730–49740.

[102] B. S. Bhati, C. Rai, B. Balamurugan, F. Al-Turjman, An intrusion detection scheme based on the ensemble of discriminant classifiers, Computers & Electrical Engineering 86 (2020) 106742.

[103] E. Zyad, A. Taha, B. Mohammed, Improve r2l attack detection using trimmed pca, in: 2019 International Conference on Advanced Communication Technologies and Networking (CommNet), 2019, pp. 1–5.

[104] M. Saharkhizan, A. Azmoodeh, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, A Hybrid Deep Generative Local Metric Learning Method for Intrusion Detection, Springer International Publishing, Cham, 2020, pp. 343–357.

[105] Z. Cheng, S. Chai, A cyber intrusion detection method based on focal loss neural network, in: 2020 39th Chinese Control Conference (CCC), 2020, pp. 7379–7383.

[106] K. Mambwe Sydney, Y. Sun, Performance analysis of intrusion detection systems using a feature selection method on the unsw-nb15 dataset, Journal Of Big Data 7 (2020) 1–20.

[107] G. Kaur, A comparison of two hybrid ensemble techniques for network anomaly detection in spark distributed environment, Journal of Information Security and Applications 55 (2020) 102601.

[108] W. Zong, Y.-W. Chow, W. Susilo, Interactive three-dimensional visualization of network intrusion detection data for machine learning, Future Generation Computer Systems 102 (2020) 292 – 306.

[109] Kurniabudi, D. Stiawan, Darmawijoyo, M. Y. Bin Idris, A. M. Bamhdi, R. Budiarto, Cicids-2017 dataset feature analysis with information gain for anomaly detection, IEEE Access 8 (2020) 132911–132921.

[110] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. Pham, S. K. Padannayil, K. Simran, A visualized botnet detection system based deep learning for the internet of things networks of smart cities, IEEE Transactions on Industry Applications 56 (4) (2020) 4436–4456.

[111] S. Y. Yerima, M. K. Alzaylaee, Mobile botnet detection: A deep learning approach using convolutional neural networks, in: 2020 International Con-

ference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2020, pp. 1–8.

[112] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, H. Gacanin, Hybrid deep learning for botnet attack detection in the internet of things networks, IEEE Internet of Things Journal (2020) 1–1.

[113] S. Daskalaki, I. Kopanas, N. Avouris, Evaluation of classifiers for an uneven class distribution problem, Applied Artificial Intelligence 20 (5) (2006) 381–417.

[114] D. de Ridder, O. Kouropteva, O. Okun, M. Pietikäinen, R. P. W. Duin, Supervised locally linear embedding, in: Artificial Neural Networks and Neural Information Processing — ICANN/ICONIP 2003, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 333–341.

[115] K. He, J. Sun, Convolutional neural networks at constrained time cost, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 5353–5360.

[116] X. Mao, Z. Su, P. S. Tan, J. K. Chow, Y.-H. Wang, Is discriminator a good feature extractor? (2020). arXiv:1912.00789.

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

[118] R. K. Vigneswaran, R. Vinayakumar, K. P. Soman, P. Poornachandran, 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), 2018, pp. 1–6.

[119] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: Proceedings of the 34th International Conference on Machine Learning, Vol. 70 of Proceedings of Machine Learning Research,

PMLR, International Convention Centre, Sydney, Australia, 2017, pp. 214–223.

[120] J. Donahue, K. Simonyan, Large scale adversarial representation learning, CoRR abs/1907.02544 (2019) 1–32.

[121] A. C. Lorena, A. C. P. de Leon Ferreira de Carvalho, J. Gama, A review on the combination of binary classifiers in multiclass problems, Artif. Intell. Rev. 30 (1-4) (2008) 19–37.

[122] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626. doi:10.1109/ICCV.2017.74.

[123] A. Mahendran, A. Vedaldi, Visualizing deep convolutional neural networks using natural pre-images, Int. J. Comput. Vision 120 (3) (2016) 233–255.