IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 3, 2020, accepted February 29, 2020, date of publication March 16, 2020, date of current version March 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980937

Multi-Channel Deep Feature Learning for Intrusion Detection

GIUSEPPINA ANDRESINI[1], ANNALISA APPICE[1,2], NICOLA DI MAURO[1], CORRADO LOGLISCI[1], AND DONATO MALERBA[1,2], (Member, IEEE)

[1]Department of Computer Science, University of Bari Aldo Moro, I-70125 Bari, Italy
[2]Consorzio Interuniversitario Nazionale per l'Informatica – CINI, I-70125 Bari, Italy

Corresponding author: Giuseppina Andresini (giuseppina.andresini@uniba.it)

This work was supported in part by the research project "TALIsMan-Tecnologie di Assistenza personALizzata per il Miglioramento della quAlità della vitA" (ID: ARS01_01116) within "PON RI 2014-2020" funded by the Italian Ministry for Universities and Research (MIUR).

ABSTRACT Networks had an increasing impact on modern life since network cybersecurity has become an important research field. Several machine learning techniques have been developed to build network intrusion detection systems for correctly detecting unforeseen cyber-attacks at the network-level. For example, deep artificial neural network architectures have recently achieved state-of-the-art results. In this paper a novel deep neural network architecture is defined, in order to learn flexible and effective intrusion detection models, by combining an unsupervised stage for multi-channel feature learning with a supervised one exploiting feature dependencies on cross channels. The aim is to investigate whether class-specific features of the network flows could be learned and added to the original ones in order to increase the model accuracy. In particular, in the unsupervised stage, two autoencoders are separately learned on normal and attack flows, respectively. As the top layer in the decoder of these autoencoders reconstructs samples in the same space as the input one, they could be used to define two new feature vectors allowing the representation of each network flow as a multi-channel sample. In the supervised stage, a multi-channel parametric convolution is adopted, in order to learn the effect of each channel on the others. In particular, as the samples belong to two different distributions (normal and attack flows), the samples labelled as normal should be more similar to the representation reconstructed with the normal autoencoder than that of the attack one, and viceversa. This expected dependency will be exploited to better disentangle the differences between normal and attack flows. The proposed neural network architecture leads to better predictive accuracy when compared to competitive intrusion detection architectures on three benchmark datasets.

INDEX TERMS Cybersecurity, intrusion detection, machine learning, computer security.

I. INTRODUCTION

The goal of a network intrusion detection system (IDS) is to discover any unauthorised access to a computer network by analysing traffic on the network for signs of malicious activity. In particular, the network intrusion detection task is to build a predictive model capable of distinguishing between attack and normal network flows. Despite decades of developments, existing IDSs still face challenges in improving the detection accuracy by reducing the false alarm rate and detecting unknown attacks. To solve these problems, many researchers have focused on developing IDSs that capitalise on machine learning methods [1]. The category of the machine learning methods [2]–[4] has achieved satisfactory detection levels when sufficient training data are available and sophisticated hand-engineering features are constructed to achieve sufficient generality and detect both attack variants and novel attacks.

With the advent of deep learning [5], the task of hand-engineering features has been replaced with trainable multi-layer networks that have shown impressive feature representation capability for a wide range of applications. The recent research trend is recognising deep learning as a definitely relevant approach in intrusion detection [1], [6], [7], since (non-linear) multiple activation layers may actually facilitate the discovery of effective patterns that keep their effectiveness also under drifting conditions [8]. In this case, raw input data are transformed into higher representations

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

through consecutive transformations, with each transformation reaching a higher level of abstraction and complexity, which is useful for gaining accuracy in the predictive task.

Among the different approaches in deep learning, autoencoder architectures have received significant attention. An autoencoder is an artificial neural network (NN) consisting of an encoder function mapping the input to a hidden code and a decoder, producing the reconstructed input learned by minimizing a loss function. As the hidden code commonly reduces the size of data, autoencoders are mostly used for saving the output of the encoder function for dimensionality reduction [9]–[13]. In any case, there are few studies that learn autoencoders, which go beyond the dimensionality reduction purpose, e.g., considering the output of the decoder function for data denoising [14] or the loss (residual error) for the anomaly detection [15]–[17]. In particular, the loss has been recently used as a likelihood measure to assess the outlier degree of a sample in intrusion detection tasks [18].

The aim of this paper is to enrich the representation of the original flows with class-specific information, in order to facilitate the disentanglement between the classes. In particular, we consider network flows represented as raw data vectors and use autoencoders, in a novel manner, to derive a multi-channel representation of each network flow. One way to learn a new representation specific for each kind of flow is to learn two autoencoders on normal and attack flows, separately. Then they are used to restore each network flow by applying the learned encoder and decoder in cascade to the original data row representation. The restored representation obtained represents a new description of the original flow. In this way, each sample is represented as a multi-channel sample spanned on three dimensions, that is, the original raw vector, as well as the two vectors recovered through the autoencoders.

We note that our use of autoencoders is novel with respect to the common one in the literature. In fact, it leads to augmenting the representation size instead of performing dimensionality reduction. On the other hand, considering the output of the decoder function of class-specific autoencoders is not directed to operate simple data denoising. In principle, the autoencoder trained on the normal samples can contribute to recovering denoised normal samples, but it should see attacks as anomalies, and so reconstruct them badly. Viceversa, the autoencoder trained on the attacks should denoise the attacks, seeing the normal flows as anomalies and badly reconstructing them. The idea is to exploit possible patterns existing among the channels, given the class of flow. Multi-channel deep feature learning can disclose these patterns, which aids in intrusion detection.

Multi-channel feature learning has recently gained attention in both image analysis [19] and speech recognition [20], where the learned model can be improved by capturing possible correlation among multiple channels. In general, multiple channels are dealt with through feature-level fusion-based approaches [21] or decision-level fusion-based strategies [22]. However, these approaches separate the feature extraction from the dependency modelling—they may under-utilise the ability of modelling dependencies. To address this issue, we learn the intrusion detection model through a convolution neural network, adopting many convolutional filters to learn dependencies among multiple channels, i.e., learning channel-based representations. The ability of deep learning in automatic feature extraction and feature selection reduces the difficulties in computing domain-specific, hand-engineered features, and helps us to bypass the traditional feature selection phase.

In short, the main contributions of this work are the following:

1) An extensive discussion of the state-of-the-art works in deep learning for intrusion detection.
2) The definition of a novel deep learning intrusion detection approach, named MINDFUL (MultI-chanNel Deep FeatUre Learning for intrusion detection), that uses autoencoders to derive a multi-channel representation of flows, and resorts to a deep learning architecture with convolutions, in order to disclose possible patterns hidden in the adopted multi-channel representation. To the best of our knowledge, this is the first study that explores multi-channel deep feature learning in intrusion detection.
3) An extensive evaluation of the effectiveness of the proposed approach in intrusion detection for samples collected in several benchmark datasets. The empirical study investigates the ability of our approach to increase accuracy when compared to several competitors taken from the recent literature on deep learning in intrusion detection.

The paper is organised as follows. The related works are presented in Section II. The formulated machine learning methodology is describe in Section III. The findings in the evaluation of the proposed strategy are discussed in Section IV. Finally, Section V refocuses on the purpose of the research, draws conclusions and proposes future developments.

II. RELATED WORKS

The recent literature on network security proves how the advances made in machine learning have led the problem of intrusion detection to a more challenging level of study and relative computational solutions to an improved level of performance [23]. Neural network architectures, and more generally deep learning, offer a determinant contribution, thanks to the opportunity to deal with high-dimensionality and non-linearity that are typical of network-related data. They make classical intrusion detection systems perform poorly, since trainable multi-layer networks achieve higher feature representation capabilities than sophisticated hand-engineering features constructed by classical approaches. In fact, to deal with network data complexity, classical intrusion detection systems commonly perform complex data transformations to obtain highly qualified training data. One possibility is offered by the methods of feature selection and feature extraction. In the first category, we find works focused on the identification of an optimal feature subset of the original features, e.g. ranking methods based on Chi-square statistical significance test or fuzzy rough set theory. In the second category, we can collocate techniques such as principal component analysis (PCA) and linear discriminant analysis (LDA). Their main characteristic is defining a low-dimensional space that fully takes the correlation between features into consideration, although they require manual experience and data pre-processing skills [24], [25] and perform linear combinations only. We find few attempts in the literature, which adopt an opposite approach, that is, projecting data into a higher-dimensional space, where only the kernel-based methods are applicable [26].

Deep learning approaches represent a valid alternative because they have a good potential for achieving effective data representation. The superiority of the accuracy performance of deep learning in intrusion detection applications is already proved in [7], [27]. In this section, we focus on autoencoders and convolutional neural networks, as they are more related to the method proposed in this paper.

A well-consolidated research stream has focused on the use of neural autoencoding models to lower the high dimensionality of original raw data in favour of compressed representations that exclude features prone to mis-classification [10]–[13], [28]. Stacked autoencoders are also considered in combination with traditional classifiers (e.g., SVM, K-NN, Gaussian Naive-Bayes) [29]. In addition, Zeng *et al.* [30] adopt stacked autoencoders, in which the compressed output of an autoencoder is used as the input of the autoencoder in the next layer.

Ali and Li [31] study the specific task of DDoS attack detection by combining the features produced by a hierarchy of autoencoders, which are then unified according to a weighting schema in Multiple Kernel Learning. Li *et al.* [32] try to increase the intrusion detection rate by combining pre-trained Restricted Boltzmann Machines, used as autoencoders, with a fine-tuning phase performed after the decoding operation. However, experimental results of this study prove that the Deep Belief Network model without autoencoders can perform even better than configurations integrating autoencoders. A common aspect in the above mentioned works is that complex autoencoders pay the price of several stages of training, resulting in making the whole method inefficient, without stably improving the attack detection capabilities [29]. In addition, they all train the autoencoders on the entire training set and focus on the dimensionality data reduction purpose.

Recent studies have explored the possibility to combine auotencoders, finalised to the dimensionality reduction, with non-neural methods, finalised to the construction of new features able to make inter-dependencies between class and features explicit. In [33], the authors propose a multi-perspective vectorized representation built with both a feature generalization step and a feature memorization step. The feature generalization step converts raw input features into dense vectors through a sparse autoencoder. The memorisation step accounts for feature interactions extracted through cross-product feature transformations. This is different from our method, since in [33] correlations are pre-extracted as a part of the feature extraction step, while in our method correlations are directly extracted cross channels in the classification step.

Andresini *et al.* [18] have recently investigated the use of autoencoders for both feature augmentation and anomaly detection. They propose a two-stepped intrusion detection deep learning algorithm. In the first step, a deep neural network is trained on an input feature space extended with a new attribute measuring the loss (residual error) of an autoencoder that reconstructs the training samples. In the second step, an autoencoder is used as an anomaly detector to refine classifications. In both steps, autoencoders are trained from class-normal samples only. Again this is different from our method presented here, although we also use autoencoders to synthesise new feature spaces. However, in [18], a single autoencoder is trained from normal samples only, while in our method two autoencoders are trained from normal samples and attack samples separately. In [18], one new feature is synthesised considering the loss, while in our method we consider autoencoders to define two new feature vectors arranging a multi-channel representation of the samples.

The idea of constructive new features has also been investigated in [34], where the authors try to unearth the class-discriminating ability of the original feature space by creating new feature spaces through the marginal density ratios of the class estimations. This is a direct projection of the input space, which, therefore, could inherit the data sparsity, as well as the noisy data. However, this method does not use deep learning architectures. In addition, it trains separate classifiers (SVMs) from the distinct feature spaces and merge decisions through an ensemble. This means that possible correlations a cross various feature spaces are ignored during the classifier learning in [34].

Research investigating deep learning in intrusion detection has recently focused on using Convolution Neural Networks (CNNs). These are a family of robust, popular neural networks designed to process input data stored in arrays. They are commonly considered where there is spatial or temporal ordering in input data. Therefore, to learn 2-dimensional CNNs in intrusion detection, network flows must be mapped into 2D image arrays, expressing latent characteristics of input data within a 2D data representation [35], [36]. To this aim, Li *et al.* [35] describe a quantization method to convert the value of each numeric feature into an 8-digit binary pixel. The input representation built with this method is finally processed as the input of two popular CNNs, that is, ResNet50 [37] and GoogLeNet [38]. Kim *et al.* [36] illustrate a RGB-like approach, that outperforms the one in [35]. This input representation is processed in combination with GoogLeNet Inception V3 [38]. However, these approaches, representing features as pixels, assume unconfirmed spatial

TABLE 1. List of symbols.

Symbol	Description
$\mathcal{D}$	set of training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$
$\mathbf{X}$	data matrix of single-channel samples $\mathbf{x} \in \mathbb{R}^D$
$\mathbf{X}^n$	subset of samples in $\mathbf{X}$ whose label is normal
$\mathbf{X}^a$	subset of samples in $\mathbf{X}$ whose label is attack
f	encoder network
g	decoder network
z_n	autoencoder $g_n \cdot f_n$ trained on $\mathbf{X}^n$
z_a	autoencoder $g_a \cdot f_a$ trained on $\mathbf{X}^a$
$\mathbf{x}_i$	single-channel sample $\mathbf{x}_i \in \mathbf{X}$
$\hat{\mathbf{x}}_i^n$	$\mathbf{x}_i$ reconstructed by z_n
$\hat{\mathbf{x}}_i^a$	$\mathbf{x}_i$ reconstructed by z_a
$\mathbf{x}_i'$	multi-channel sample with $\mathbf{x}_i' = [\mathbf{x}_i, \hat{\mathbf{x}}_i^n, \hat{\mathbf{x}}_i^a]$
$\mathbf{X}'$	data matrix of multi-channel samples $\mathbf{x}' \in \mathbb{R}^{D\times 3}$

relations between features that depend on the order in which they are processed.

In the other-hand, Lopez *et al.* [26] use 1-dimensional CNNs with vector-like data. However, their study computes convolutions on original data without accounting for autoencoders.

Recently, Sharma *et al.* [39] adopted a CNN architecture to process software binaries as matrix-like data, by scanning them along one dimension. They argue that this solution leads to accuracy and efficiency gain compared with existing solutions for malware detection tasks (malicious software or benign software). Their idea is similar to the one we propose. In fact we deem it unnecessary to scan the network flows along 2-dimensions, as one should do with the real images, and we read the matrix vector-by-vector from top to bottom. Contrary to our work, however, Sharma *et al.* [39] propose representing sequences of vectors (software binaries) in the form of single-channel matrices, while we rely on the combined use of raw data with class-discriminating features, which motivates the use of multi-channel CNNs.

III. THE PROPOSED METHOD

In this section we describe MINDFUL—the multi-channel deep learning method we propose to deal with the problem of network intrusion detection. It combines an unsupervised approach for multi-channel feature construction—based on two autoencoder NNs—with a supervised one exploiting cross-channel feature correlations. The list of symbols used to describe the method is reported in Table 1.

A. AUTOENCODERS

Differently from classical multi layer perceptrons (MLPs), an autoencoder is a particular NN trained to attempt to copy its input to its output [40]. In particular, it can be viewed as being composed of two functions: an encoder f—mapping the input vector $\mathbf{x}$ to a hidden representation $\mathbf{h}$ via a deterministic mapping $\mathbf{h} = f(\mathbf{x})$, parameterized by θ_f—and a decoder g—mapping back the resulting hidden representation $\mathbf{h}$ to a reconstructed vector in the input space $\hat{\mathbf{x}} = g(\mathbf{h})$, parameterized by θ_g.

Usually the functions g and f correspond to two different NNs combined in a single one, whose parameters $\{\theta_f, \theta_g\}$ are simultaneously learned by minimizing a loss function $\mathcal{L}(\mathbf{x}, g(f(\mathbf{x})) = \mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$, penalising $\mathbf{x}$ for being dissimilar from $\hat{\mathbf{x}}$ such that $\mathcal{L}_{\mathrm{se}}(\mathbf{x}, \hat{\mathbf{x}}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2$.

In this paper, we use class-specific autoencoders for feature learning, transforming a single-channel sample into a multi-channel one.

Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ be a set of N training samples, where each $\mathbf{x}_i \in \mathbb{R}^D$ is a row vector corresponding to an input sample defined over D features, and y_i is the corresponding binary label denoting a *normal* or an *attack* sample. Furthermore, let $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top \in \mathbb{R}^{N\times D}$ denote the data matrix of N D-dimensional random variables $\mathbf{x} \in \mathbb{R}^D$.

We denote with $\mathbf{X}^n = \mathbf{X}_{|y_i=n}$, resp. $\mathbf{X}^a = \mathbf{X}_{|y_i=a}$, the subset of samples in $\mathbf{X}$, whose label is normal, resp. attack. The samples in $\mathbf{X}^n$ and $\mathbf{X}^a$ could then be separately used to learn two independent autoencoders $g_n \cdot f_n$, denoted as z_n, and $g_a \cdot f_a$, denoted as z_a.

Since the activation produced by the top layer in the decoder network corresponds to a reconstructed vector in the same input space, the idea is to consider it as new learned features. In particular, each autoencoder can be employed to build a new feature vector $\hat{\mathbf{x}} = g(f(\mathbf{x})) \in \mathbb{R}^D$ from a sample $\mathbf{x}$. These features may then be considered to transform a *single-channel* sample to a *multi-channel* one by concatenation.

Hence, each sample $\mathbf{x}_i \in \mathbb{R}^D$ could be replaced by the multi-channel sample:

$$\mathbf{x}_i' = [\mathbf{x}_i, \hat{\mathbf{x}}_i^n, \hat{\mathbf{x}}_i^a]^\top \in \mathbb{R}^{D\times 3},$$

where $\hat{\mathbf{x}}_i^n = g_n(f_n(\mathbf{x}_i))$ and $\hat{\mathbf{x}}_i^a = g_a(f_a(\mathbf{x}_i))$ are the reconstructed representations of the single-channel sample $\mathbf{x}_i$, thus leading to the extended multi-channel representation $\mathbf{X}' = [\mathbf{x}_1', \ldots, \mathbf{x}_N']^\top \in \mathbb{R}^{N\times D\times 3}$.

In this way, the sample $\mathbf{x}_i$ is enriched with information synthesised by exploiting both the normal and the attack autoencoder. When the samples belong to two different distributions, samples $\mathbf{x}_i$, labelled as normal should be more similar to the representation $\hat{\mathbf{x}}_i^n$ than that of $\hat{\mathbf{x}}_i^a$, or $||\mathbf{x}_i - \hat{\mathbf{x}}_i^n||^2 < ||\mathbf{x}_i - \hat{\mathbf{x}}_i^a||^2$, and viceversa. This conjecture has been experimentally validated in Section IV-D.1.

The aim now is to exploit, in the supervised step, the effect of one channel on each of the others, in order to better disentangle the differences between the two classes. As we will see in the next section, a solution to learn a representation among the three channels is to adopt a (1 × 1) convolutional filter—a cross-channel parametric convolution. This is also why the learned representations have been concatenated in $\mathbb{R}^{D\times 3}$—a 3-channel representation—and not in $\mathbb{R}^{3D}$, by just increasing the number of features.

B. CONVOLUTIONAL NEURAL NETWORKS

A Convolutional Neural Network (CNN) [41] is a specialised kind of NN used for processing data with grid-like topology, such as images or sequences. A CNN is able to produce a good internal representation of the world by successfully capturing (spatial and/or temporal) dependencies (e.g. edges, colours, gradient orientation) in data through the application of relevant filters [42].

Typically a CNN consists of alternating convolutional layers and spatial pooling layers. The name ''convolutional neural network'' indicates that the network employs a mathematical operation called *convolution* [40]—a specialised kind of linear operation. A convolution is a dot-product operation between a grid-structured set of weights and similar grid-structured inputs drawn from different spatial localities in the input volume [43]. A convolutional layer builds a feature map **f** by linear convolutional filters followed by a nonlinear activation function σ as:

$$f_{ijk} = \sigma(\mathbf{w}_k^\top \mathbf{x}_{ij}),$$

where (ij) is the position in the feature map, $\mathbf{x}_{ij}$ denotes the input patch—*receptive field*—centered at the point (ij), and k is the channel index in the feature map.

Usually a convolutional layer is followed by an additional layer that performs a local averaging and a sub-sampling, by both reducing the resolution of the feature map and diminishing the sensitivity of the output to shifts and distortion [44]. The purpose of this operation, called *pooling*, is to achieve spatial invariance by reducing the resolution of the feature map, preserving important information and discarding irrelevant details [45].

For two-dimensional data, such as images, the convolutions in a 2D CNN occur with multi-dimensional filters, scanning through the data dimensions from left to right and from top to bottom, in order to capture the high-level features from the input image.

Restricting the size of the filter to 1, thus using (1×1) 2D convolutions, we may obtain a dimensionality reduction/increase in the filter dimension as happens in the Google Inception module [46], as opposed to a $(k \times k)$ convolution, making a reduction in the spatial dimension. Furthermore, since they include the use of a non-linear activation, this makes them dual purpose. The same Network-in-Network approach, proposed in [47] to increase the representational power of neural networks, when applied to convolutional layers can be viewed as a (1×1) convolutional layer. In particular, a (1×1) convolution should have the effect of combining existing information in the channel dimension to obtain more abstract channel-wise information.

C. 1D CONVOLUTIONAL NEURAL NETWORKS

1D CNNs have been recently used in several domains like process mining [48], remote sensing [49], wind prediction [50], medical image processing [19] or malware detection [39]. 1D CNNs process 1-dimensional input vectors, like sequential data, and the filter in the convolution slides along one dimension only.

Restricting the size of the filter to 1 in a 1D convolutional layer, like in a (1×1) 2D convolutional layer, has the effect of a non-linear reduction/increase of the number of channels.

Now the idea is to exploit a 1D convolution, in order to increase the cross channel information. In particular, as reported in Figure 1, the 3-channel representation $\mathbf{X}'$, obtained as previously described, is input to a 1D convolutional layer with filter size equal to 1. Using a number of filters greater than three, we are able to increase the number of non-linear cross-channel dependencies.

In particular, given a sample $\mathbf{x}'_i = [\mathbf{x}_i, \hat{\mathbf{x}}_i^n, \hat{\mathbf{x}}_i^a]^\top \in \mathbb{R}^{D\times 3}$, the receptive field of the 1D convolution is represented by $\mathbf{x}'_{i,j,:}$, and each filter k computes the signal $f_{i,j,k}$ in the feature map as:

$$f_{i,j,k} = \sigma(\mathbf{w}_k^\top \mathbf{x}'_{i,j,:} + \mathbf{b}_k),$$

with weights and bias $\mathbf{w}_k, \mathbf{b}_k \in \mathbb{R}^3$, and σ a non-linear activation function. The channels of each feature in $\mathbf{x}'_{i,j,:}$ are convolved with the same shared weights and mapped to the feature map with a non-linearity. Adopting K filters in the convolutional layer leads to a transformation of a sample in $\mathbb{R}^{D\times 3}$ to a feature map in $\mathbb{R}^{D\times K}$.

An alternative solution could be to concatenate the learned features in the space $\mathbb{R}^{3D}$ instead of in $\mathbb{R}^{D\times 3}$, and then input them to a fully connected layer. However, the topology of the input is completely ignored in a fully connected layer [44]—the output of the training is not affected by the order of the input features. Hence, with this alternative, we may lose both the channel-based ordering and the possibility to learn new cross-channel features.

The output of the 1D convolutional layer is then flattened and dealt with as the input to two stacked fully-connected layers (FC), computing the output classification probabilities using a final softmax layer. The overall architecture of our proposed MINDFUL model is reported in Figure 1. The pseudocode of MINDFUL is described in Algorithm 1

IV. EMPIRICAL STUDY

We consider three benchmark datasets, described in Section IV-A, in order to evaluate the effectiveness of the intrusion detection methodology implemented in MINDFUL. 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. Specifically, we analyse the performance of MINDFUL along the various accuracy metrics presented in Section IV-C. These metrics are commonly considered in cybersecurity for the evaluation of intrusion detection systems. The implementation details of the deep learning architecture adopted in this experimental study are illustrated in Section IV-B, while the results, achieved on each dataset, are discussed in Section IV-D. In particular, the presentation of the results is organised as follows. First, we evaluate how the two autoencoders, computed on the normal/attack training samples separately, disclose knowledge that may contribute to separate attacks from normal flows (see Section IV-D.1).

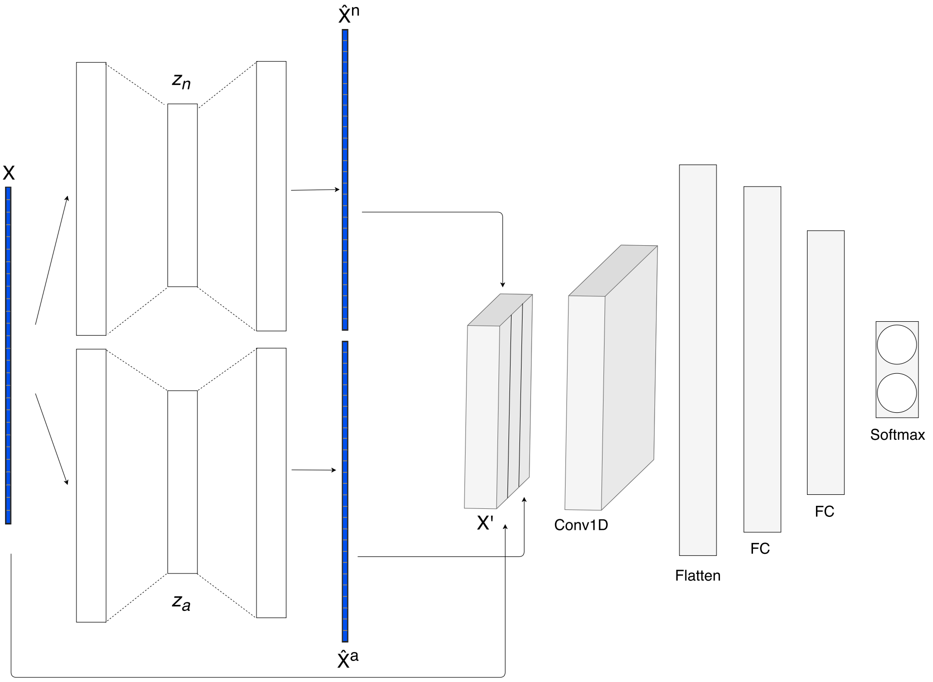


FIGURE 1. Architecture of MINDFUL. The architecture takes as training samples **X** as input. The normal samples are used to learn the first autoencoder z_n (top-left) and the attack samples are used to learn the second autoencoder z_a (bottom-left). The two autoencoders are used for each training sample **x**, both normal and attack samples — in order to return the reconstructed features $\hat{\mathbf{x}}^n$ and for $\hat{\mathbf{x}}^a$. These new features are used to build a new augmented dataset that is used as input to a 1-CNN neural network (right).

Algorithm 1 Multi-Channel Deep Learning Method for Network Intrusion Detection

```
Data:
    $\mathcal{D}$: set of training samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with
    $y_i \in \{attack, normal\}$
    $\mathbf{X}$: data matrix of single-channel samples $\mathbf{x} \in \mathbb{R}^D$
Result:
  $(z_n, z_a, model)$ : the learned intrusion detection model
begin
    /* Autoencoder training */
    $z_n \leftarrow$ trainAutoencoder($\mathbf{X}^n$)
    $z_a \leftarrow$ trainAutoencoder($\mathbf{X}^a$)
    /* Compute reconstructed vectors using the autoencoders $z_n$ and $z_a$ */
    foreach $\mathbf{x}_\in \mathbf{X}$ do
        $\hat{\mathbf{x}}_i^n \leftarrow z_n(\mathbf{x}_i)$
        $\hat{\mathbf{x}}_i^a \leftarrow z_a(\mathbf{x}_i)$
        $\mathbf{x}_i' = [\mathbf{x}_i, \hat{\mathbf{x}}_i^n, \hat{\mathbf{x}}_i^a]^\top$
    /* CNN training */
    $model \leftarrow$ train1DCNN($\mathbf{X}'$)
    return $z_n, z_a, model$
```

Second, we analyse the effectiveness of MINDFUL along the components of the adopted deep learning architecture (see Section IV-D.2). Finally, we discuss the evaluation results that are reported in the recent intrusion detection literature and which have been achieved by processing the datasets also considered in our study (see Section IV-D.5).

A. DATASET DESCRIPTION

A summary of the characteristics of the datasets considered in this evaluation is reported in Table 2.

- KDDCUP99[1] was introduced in the KDD Tools Competition organised in 1999. This is a benchmark dataset that is commonly used for the evaluation of intrusion detection systems also in recent studies [51]–[53]. It contains network flows simulated in a military network environment and recorded as vectors of 42 attributes (6 binary, 3 categorical and 32 numerical input attributes, as well as 1 class attribute). The original dataset comprised a training set of 4.898.431 samples and a testing set of 311.027 samples. As reported in [54], the testing set collects network flows belonging to 14 attack families, for which no sample is available in the training set. We note that this simulates a *zero-day* attack condition. To keep the cost of the learning stage under control, the original dataset comprises a reduced training set, denoted as 10%KDDCUP99Train, that contains 10% of the training data taken from the original dataset. In this study, we consider 10%KDDCUP99Train for the learning stage, while we use the entire testing set, denoted as KDDCUP99Test, for the evaluation stage.[2] We note

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

[2] 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

TABLE 2. Dataset description. For each dataset we report: the number of attributes, the total number of network flow samples collected in the dataset, the number of normal network flows (and their percentage on the total size) and the number of attacking flows (and their percentage on the total size).

Dataset	Attributes	Total	Normal(%)	Attacks(%)
10%KDDCUP99Train	42	494021	97278 (19.7%)	396743 (80.3%)
KDDCUP99Test		311029	60593 (19.5%)	250436 (80.5%)
UNSW-NB15Train	43	82332	37000 (44.9%)	45332 (55.1%)
UNSW-NB15Test		175341	56000 (31.9%)	119341 (68.1%)
CICIDS2017Train	79	100000	80000 (80%)	20000 (20%)
CICIDS2017Test		900000	720000 (80%)	180000 (20%)

that this experimental scenario, with both 10%KDDCUP99Train and KDDCUP99Test, is commonly used in the literature (e.g. [32], [55], [56]). In addition, the entire dataset is imbalanced in both the training and testing set, where the percentage of attacks is higher than that of normal flows (80.3% vs 19.7% in the training set and 80.5% vs 19.5% in the testing set).

- UNSW-NB15 dataset[3] was created by the IXIA PerfectStorm tool[4] in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). This is a hybrid dataset that includes the realistic modern normal activities and the synthetic contemporary attack behaviour extracted from network traffic monitored in 2015 [57]. The dataset has recently been used in the evaluation of various intrusion detection approaches [26], [36], [55], [58]. It consists of one training set and one testing set, which comprise network flow samples, stored as the vectors of 43 attributes (2 binary, 3 categorical and 37 numerical input attributes and 1 class attribute). The dataset, that is quite balanced in the training set, is a little imbalanced in the testing set, where the percentage of attacks is slightly higher than the percentage of normal flows (68.1% vs 31.9%).
- CICIDS2017[5] was collected by the Canadian Institute for Cybersecurity in 2017. This dataset contains normal flows and the most up-to-date common attacks, which resemble the true real-world data (PCAPs). It also comprises the results of the network traffic analysis, performed using CICFlowMeter with the labelled flows based on the timestamp, source and destination IPs, source and destination ports, protocols and attack. The original dataset was a 5-day log collected from Monday July 3, 2017 to Friday July 7, 2017 [59]. The first day (Monday) contained only benign traffic, while the other days contained various types of attack, in addition to normal network flows. Every network flow sample is spanned over 79 attributes (18 binary and 60 categorical input attributes and 1 class attribute) [59]. We note that this dataset is commonly used in the evaluation of anomaly detection approaches with the learning stage performed on the first day [60], [61]. However, a few recent studies have considered these data also in the evaluation of classification approaches, as we do in this paper [36], [56], [62]. In our experimental study, we consider the training and testing sets built according to the strategy described in [36]. So, 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 have used the stratified random sampling, in order to select 80% of normal flows and 20% of attacks, as in the original log. This dataset is imbalanced in both the learning stage and the evaluation stage. In fact, the number of normal network flows is significantly higher than the number of attacks (80% vs 20%). We note that this resembles the common set-up of an anomaly detection learning task that often occurs in a network.

B. IMPLEMENTATION DETAILS

The proposed methodology has been implemented in Python 3.7 using the Keras 2.3[6] library with TensorFlow[7] as backend. The source code is available online.[8]

For each dataset, we have conducted a hyper-parameter optimization using the tree-structured Parzen estimator algorithm as implemented in the Hyperopt library [63], by using 20% of the training set as the validation set. We chose the configuration of the parameter that achieved the best validation loss. The hyper-parameters and their corresponding possible values are reported in Table 3. Data have been scaled using the Min-Max scaler.

The autoencoders have been defined with three layers. These layers include 40 × 10 × 40 neurons, in KDDCUP99 and UNSW-NB15 datasets, and 50 × 10 × 50 neurons in CICIDS2017. A dropout layer is placed before the decode layer, in order to perform data regularisation and prevent the overfitting. For the autoencoders, the mean squared error

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

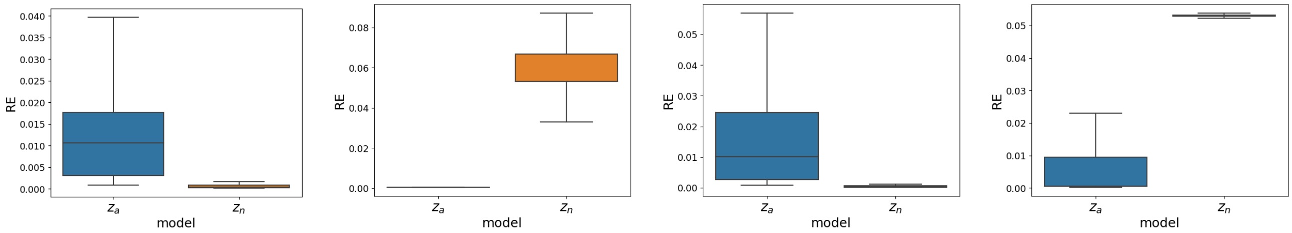
[4] https://www.ixiacom.com/products/perfectstorm

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

[6] https://keras.io/

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

[8] https://github.com/gsndr/MINDFUL



(a) Training normal flows (b) Training attacks (c) Testing normal flows (d) Testing attacks

FIGURE 2. Reconstruction error analysis (KDDCUP99 dataset): normal and attack flows reconstructed with the autoencoders trained on the normal flows (z_n) and the attacks (z_a), respectively. Figures 2(a) and 2(b) show the box plots of the reconstruction errors (*RE*) of the training samples, while Figures 2(c) and 2(d) show the box plots of the reconstruction errors of the testing samples.

TABLE 3. Hyperparameter search space for both the autoencoders and classifier.

	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, 0.01]	[0.0001, 0.01]
dropout	[0,1]	[0,1]

(*mse*) has been used as the loss function. The classical rectified linear unit (*ReLu*) [64] has been selected as the activation function for each hidden layer, while for the last layer the *Linear* activation function has been used.

As regards the classifier, the architecture consists of a 1D convolutional layer and three fully-connected layers. The network takes samples of size ($D \times 3$) as input and predicts a Bernoulli probability. The input sample is transformed by the 1D convolutional layer with 64 filters into a feature map of size ($D \times 64$), that is processed by the following fully-connected layer of size 320, 160 and 2, respectively. The output probabilities are obtained using the softmax activation function in the last layer. The ReLu activation function has been used in all the other layers. In order to perform data regularisation, a dropout layer follows each layer in the network. For this architecture, weights are optimised by minimising the binary-cross entropy as the loss function.

The networks are trained with mini-batches by backpropagation, and the gradient-based optimisation is performed using the Adam update rule [65]. The weights are initialized following the Xavier scheme. Furthermore, a maximum number of epochs equal to 150 has been set, retaining the best models, using an early stopping approach that achieves the lowest loss on a validation set.

C. EVALUATION METRICS

The overall performance of the proposed approach is measured by analysing both the accuracy and F-score of the intrusion detection models learned. While the accuracy is the ratio of flows correctly labelled, the F-score is the harmonic mean of precision and recall, where the precision measures the ability of an intrusion detection system to identify only the attacks, while the recall can be thought as of a system's ability to find all the attacks. Mathematically, the precision is calculated as the ratio of the attacking flows correctly labelled by the intrusion detection algorithm to all attacking flows labelled at each independent configuration parameter set. The recall is calculated as the ratio of the attacking flows correctly labelled by the algorithm to all flows which are actually attacking. The higher the F-score, the better the balance between precision and recall achieved by the algorithm. On the contrary, the F-score is not so high when one measure is improved at the expense of the other.

D. RESULTS

The performance of the proposed approach is measured reporting the residual error of autoencoders, as well as the accuracy and the F-score of intrusion detection models.

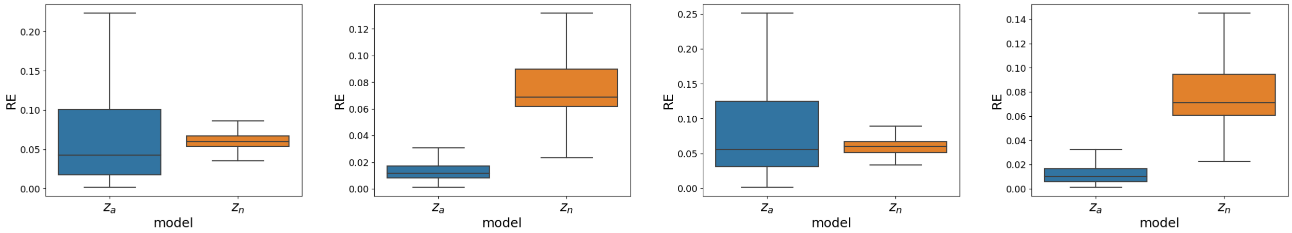
1) AUTOENCODER ANALYSIS

We start by investigating how the autoencoders can actually disclose knowledge that contributes to separating attacking flows from normal ones. To this aim, we explore how the autoencoders z_n (trained on normal samples) and z_a (trained on attack samples) can accurately reconstruct samples from both classes.

Figures 2, 3 and 4 show the box plots of the reconstruction errors, computed as $||\mathbf{x}-\hat{\mathbf{x}}||^2$, when both autoencoders z_n and z_a are used to reconstruct both normal and attack flows $\mathbf{x}$ as $\hat{\mathbf{x}}$ in each considered dataset.

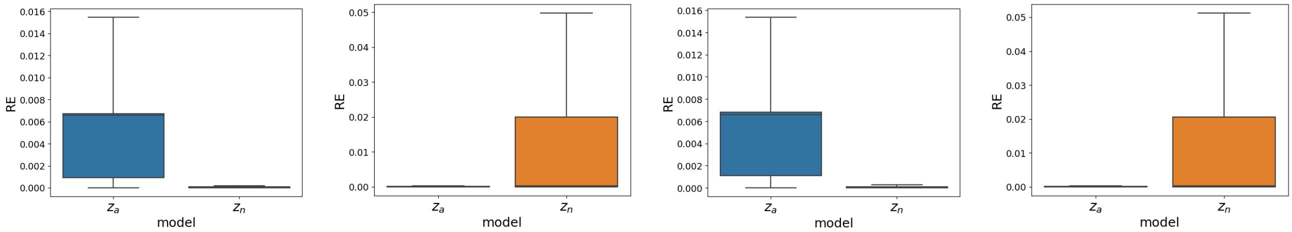
These plots confirm that the autoencoder z_n is, in general, more accurate in reconstructing the normal samples than in reconstructing the attack ones. On the other hand, the opposite behaviour is evident for z_a. This behaviour, that was expected on the training samples (as shown in Figures 2(a), 2(b), 3(a), 3(b), 4(a) and 4(b)), has also been evident on testing samples (as shown in Figures 2(c), 2(d), 3(c), 3(d), 4(c) and 4(d)), although the testing samples comprise network flows unseen at training time.

In general, the result of this analysis supports the idea that the knowledge enclosed in the autoencoders, separately trained from the normal and attack samples, allows us to introduce information that can contribute to separating the two classes. Furthermore, this result also suggests that



(a) Training normal flows (b) Training attacks (c) Testing normal flows (d) Testing attacks

FIGURE 3. Reconstruction error analysis (UNSW-NB15 dataset): normal and attack flows reconstructed with the autoencoders trained on the normal flows (z_n) and the attacks (z_a), respectively. Figures 3(a) and 3(b) show the box plots of the reconstruction errors (*RE*) of the training samples, while Figures 3(c) and 3(d) show the box plots of the reconstruction errors of the testing samples.



(a) Training normal samples (b) Training attacks (c) Testing normal samples (d) Testing attacks

FIGURE 4. Reconstruction error analysis (CICIDS2017 dataset): normal and attack flows reconstructed with the autoencoders trained on the normal flows (z_n) and the attacks (z_a), respectively. Figures 4(a) and 4(b) show the box plots of the reconstruction errors (*RE*) of the training samples, while Figures 4(c) and 4(d) show the box plots of the reconstruction errors of the testing samples.

our methodology can better benefit from autoencoder-based number of samples in both classes.

2) ABLATION STUDY

We proceed with the analysis by studying how a) the additional information synthesised through the autoencoders, b) the multi-channel input representation and c) the convolutions can jointly contribute to gain accuracy in the intrusion detection model learned by MINDFUL. To this purpose, we consider four architecture configurations as baselines. These are in turn defined by removing the autoencoders' information, the multi-channel input representation, or the convolutions from the whole architecture of MINDFUL. In particular, we consider the following baseline architectures:

- NN: it consists of the last 3 fully-connected layers of the MINDFUL architecture, and it processes as input samples $\mathbf{x}_i \in \mathbb{R}^D$, represented in the original feature space, i.e., $\mathbf{X}^{(D)} \rightarrow$ FC(320) $\rightarrow$ FC(160) $\rightarrow$ FC(2);
- ANN: it accounts for the autoencoders' information. The architecture is the same as NN, but taking as input samples $\mathbf{x}_i \oplus \hat{\mathbf{x}}_i^n \oplus \hat{\mathbf{x}}_i^a \in \mathbb{R}^{3D}$, where the output of the autoencoders z_n and z_a has been row-concatenated, i.e., $\mathbf{X}^{(3D)} \rightarrow$ FC(320) $\rightarrow$ FC(160) $\rightarrow$ FC(2). In particular, we want to see how much is added by considering the autoencoders' information wrt NN;
- CNN: it works like NN, taking as input samples $\mathbf{x}_i \in \mathbb{R}^D$, but adding a 1D convolutional layer before the fully-connected layers, i.e., $\mathbf{X}^{(D)} \rightarrow$ Conv1D(64) $\rightarrow$ FC(320) $\rightarrow$ FC(160) $\rightarrow$ FC(2);
- ACNN: it is like CNN, but the same as ANN it takes as input the row-concatenated autoencoders' information, i.e., $\mathbf{X}^{(3D)} \rightarrow$ Conv1D(64) $\rightarrow$ FC(320) $\rightarrow$ FC(160) $\rightarrow$ FC(2). In particular, it is like MINDFUL, but the outputs of the autoencoders z_n and z_a are not concatenated to obtain a multi-channel representation.

The baselines listed above have been run with the same parameter set-up (e.g. activation function, number of neurons and loss function) adopted to run MINDFUL (see the description in Section IV-B).

We evaluate the performance of MINDFUL, NN, ANN, CNN and ACNN in terms of accuracy and F-score. The results, reported in Table 4 for all the datasets, show that MINDFUL outperforms all its baselines. This confirms the effectiveness of combining autoencoders, convolutions and multi-channel input, in order to gain accuracy in an intrusion detection task. In particular, we note that autoencoders decoupled from convolutions cannot guarantee an overall improvement of the performance. On the other hand, putting autoencoders aside, a convolution dense layer commonly improves the intrusion detection accuracy. In any case, the highest overall accuracy and the highest F-score are commonly achieved when convolutions are applied to data enriched with autoencoders. Concerning that point, our analysis highlights that the superiority of MINDFUL due to convolutions depends on the ability of computing convolutions on multiple channels (instead of a single channel built by concatenation), looking for features across the original variables and their autoencoder-based counterparts.

TABLE 4. Accuracy and F-score measured on KDDCUP99Test, UNSW-NB15Test and CICIDS2017Test: MINDFUL, NN, ANN, CNN and ACNN. The best results are in bold.

Dataset		Architecture				
		MINDFUL	NN	ANN	CNN	ACNN
KDDCUP99Test	Accuracy	**92.49**	92.06	91.96	92.18	92.11
	F-score	**95.13**	94.83	94.75	94.92	94.87
UNSW-NB15Test	Accuracy	**93.40**	85.84	79.28	87.71	90.68
	F-score	**95.29**	90.51	83.06	91.72	93.49
CICIDS2017Test	Accuracy	**97.90**	95.89	96.71	95.71	96.54
	F-score	**94.93**	89.50	91.72	89.35	91.32

TABLE 5. Number of parameters (weights) learned with each neural network on KDDCUP99Train, UNSW-NB15Train and CICIDS2017Train: MINDFUL, NN, ANN, CNN and ACNN.

Dataset	Architecture				
	MINDFUL	NN	ANN	CNN	ACNN
KDDCUP99Train	891.938	65.122	91.362	891.810	2.571.170
UNSW-NB15Train	912.418	65.442	92.322	912.290	2.632.610
CICIDS2017Train	1.649.698	76.962	126.882	1.649.570	4.844.450

In addition, we analyse the number of parameters estimated with MINDFUL, NN, ANN, CNN and ACNN. The results reported in Table 5 show that the higher accuracy of MINDFUL is commonly achieved at the expenses of the higher number of parameters to estimate. In any case, we note that ACNN, which is the runner-up in the accuracy analysis reported in Table 4 for KDDCUP99 and UNSW-NB15, requires a higher number of parameters than MINDFUL.

3) VARYING THE NUMBER OF FILTERS

To explore in depth the advantages of using convolutions on multi-channel input, we have performed an additional experiment, where MINDFUL has been run by varying the number of filters in the convolutional layer. Let us consider that filters can be seen as pattern detectors, as they multiply the number of channels. Our point of view is that the higher the number of filters, the better the pattern detected by augmenting the data volume along the channel dimension through the convolutions and, consequently, the higher the accuracy of the intrusion detection model. To validate this point of view, we analyse the F-score by varying the number of filters in the convolutional layer. Figure 5 plots the F-score of a simplified version of the MINDFUL architecture, as it is measured on both the training set and testing set of UNSW-NB15, by varying the number of filters of the convolutional layer among 3, 5, 7, 9 and 11. After the flatten layer, we used a single fully-connected layer, of varying size, before the last layer, in order to have architectures with the same number of parameter. When three filters were considered, 256 neurons were used in the fully-connected layer, arriving at 68 neurons in the case of 11 filters. This plot confirms our point of view, as it shows that the F-score computed on the training set, jointly with the F-score computed on the testing set, increases with the number of filters.

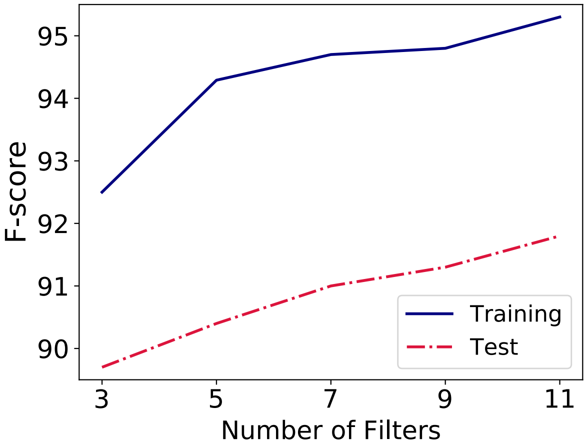


FIGURE 5. F-score of MINDFUL measured on both the training set and testing set of UNSW-NB15 by varying the number of filters among 3, 5, 7, 9 and 11.

4) IMBALANCED SCENARIO

We analyse the robustness of the proposed approach to the imbalance phenomenon. For this analysis, we consider the CICIDS2017 dataset, where data have already been collected in an imbalanced scenario (the expected one in many real world networks), consisting of 80% normal flows

TABLE 6. Accuracy and F-score measured on KDDCUP99Test, UNSW-NB15Test and CICIDS2017Test: MINDFUL is compared to several competitors reported in the recent literature. The accuracy metrics of the competitors are collected from the reference papers. The best results are in bold. "-" denotes that no value is reported in the reference paper for the considered metric.

Dataset	Algorithm	Description	Accuracy	F-score
KDDCUP99Test	MINDFUL	Autoencoder + 1D CNN	92.49	95.13
	DNN 4 Layers [55]	DNN + Text representation methods	**93.00**	**95.50**
	DBN [32]	Deep Belief Network	91.40	-
	A+DBN [32]	Autoencoder + Deep Belief Network	92.10	-
	AIDA [18]	Autoencoder + MLP	92.36	95.04
UNSW-NB15Test	MINDFUL	Autoencoder + 1D CNN	**93.40**	**95.29**
	CNN-1D [26]	1D CNN	89.80	91.30
	Grey-scale [36]	2D CNN	80.00	84.00
	RGB [36]	2D CNN	83.00	86.50
	DNN 4 Layers [55]	DNN + Text representation methods	76.50	90.10
	MLP [26]	DNN	86.60	88.90
	WnD [33]	DNN + Embeddings	91.20	-
	MLP [33]	MLP	86.70	-
	SAE [33]	Autoencoder	88.20	-
	AIDA [18]	Autoencoder + MLP	90.54	92.71
	MDPCA-DBN [58]	Clustering + DNN	90.18	91.49
	RBM [33]	RBM	87.10	-
CICIDS2017Test	MINDFUL	Autoencoder + 1D CNN	**97.90**	**94.93**
	Grey-scale [36]	2D-CNN	-	82.00
	RGB [36]	2D-CNN	-	89.00
	AIDA [18]	Autoencoder + MLP	94.50	85.80

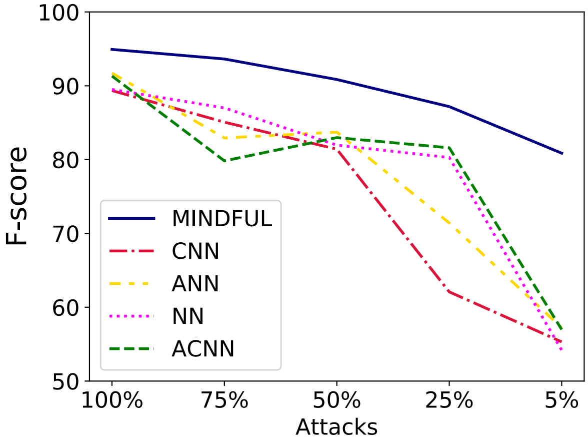


FIGURE 6. F-score of MINDFUL NN, ANN, CNN and ACNN by varying the amount of attacks in both CICIDS2017.

and 20% attacks. We stress this condition by considering new trials on this dataset, which comprise all normal flows and a sample of attacks both in the learning stage and in the evaluation stage. We consider five trials with 100% (baseline), 75%, 50%, 25% and 5% attacks, respectively. The F-score of MINDFUL, NN, ANN, CNN and ACNN, collected by diminishing the number of attacks, is plotted in Figure 6. We note that diminishing the number of attacks (and consequently stressing the imbalanced condition), the F-score of all compared algorithms decreases. In any case, MINDFUL continues outperforming its baselines independently of the balance degree in the training set.

5) COMPETITOR ANALYSIS

To complete this evaluation, we compare the accuracy performance achieved by MINDFUL to that of several competitors selected from the recent state-of-the-art literature. A summary of the characteristics of the considered competitors is reported in Table 6 (column 3). For all the methods in this comparative study, we consider the accuracy and the F-score as provided in the reference studies.

The accuracy performance of the compared methods is reported in Table 6 (columns 4 and 5) for all the datasets. These results show that MINDFUL commonly outperforms its competitors (including the competitors with autoencoders and/or CNN architectures). The only exception is observed with the performance of the competitor named DNN 4 Layers on KDDCUP99Test. This competitor learns its intrusion detection model with a deep neural network along with text representation methods to capture the contextual and sequence-related information from system calls. It comprises optimisation procedures to find the optimal parameters and the optimal topology of the network. Therefore, the higher accuracy of DNN 4 Layers on KDDCUP99Test can be due to the text representation methods, as well as to the topology and parameter setting of the architecture determined. In any case, we note that the performance of DNN 4 Layers has also been evaluated on UNSW-NB15Test in [55]. In this dataset, MINDFUL significantly outperforms DNN 4 Layers. Hence, the superiority of DNN 4 Layers is restricted to the evaluation made on KDDCUP99Test, so it may also depend on the specific characteristics of this dataset. On the other hand, MINDFUL is the runner-up in KDDCUP99Test measuring accuracy and F-score, close to DNN 4 Layers.

V. CONCLUSION

In this study, we have presented a network intrusion detection methodology, named MINDFUL. This learns an intrusion detection model through a Convolution Neural Network, trained on a multi-channel representation of network flows. Two autoencoders are learned from normal and attack flows, respectively. They are used to supply the original feature vector representation of the network flows with the feature vectors built with these autoencoders. The main idea is that patterns may exist across the channels, formed by the original features and their autoencoder-based counterparts. Disclosing these patterns may aid the intrusion detection model in separating attack flows from normal ones. To select features disclosing such patterns we use a convolutional layer with a multi-channel filter, so that it is forced to learn the possible dependencies among the channels. Representations disclosed from convolutions are processed through fully-connected layers that look for further relationships among these patterns.

We evaluate the effectiveness of the proposed methodology using three benchmark datasets that contain network flows collected in different years and scenarios. The experimental analysis confirms the effectiveness of the proposed methodology. In addition, it proves that MINDFUL gains accuracy compared to several, recently defined, state-of-the-art competitors. In particular, our experiments have proved that our methodology is more robust than its baselines also for imbalanced data.

One research direction is investigating a strategy to supply new training samples, in order to reduce the gap between the majority and minority class—when this gap exists. This could also be an important task in an adversarial environment, to be considered as one of the significant directions for future work.

Another limitation of the proposed methodology is that it does not give detailed information on the structure and characteristics of the attacks. Therefore explainable artificial intelligence may be an additional research direction here.

Finally, we consider the opportunity of extending the proposed methodology, in order to classify intrusion categories (e.g. Probe, DoS, R2L or U2R). In principle, this is possible by using separate autoencoders, learned from samples belonging to distinct intrusion categories, as multiple channels of the methodology described in this paper.

ACKNOWLEDGMENT

The authors would like to thank Lynn Rudd for her help in reading the manuscript.

REFERENCES

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

[2] A. Ahmim, M. Derdour, and M. A. Ferrag, "An intrusion detection system based on combining probability predictions of a tree of classifiers," *Int. J. Commun. Syst.*, vol. 31, no. 9, Jun. 2018, Art. no. e3547.

[3] G. Shang-fu and Z. Chun-lan, "Intrusion detection system based on classification," in *Proc. IEEE Int. Conf. Intell. Control, Autom. Detection High-End Equip.*, Jul. 2012, pp. 78–83.

[4] Z. Xiaofeng and H. Xiaohong, "Research on intrusion detection based on improved combination of K-means and multi-level SVM," in *Proc. IEEE 17th Int. Conf. Commun. Technol. (ICCT)*, Oct. 2017, pp. 2042–2045.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[6] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *Proc. 8th IEEE Int. Conf. Commun. Softw. Netw. (ICCSN)*, Jun. 2016, pp. 581–585.

[7] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018.

[8] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1282–1289.

[9] N. Shone, T. Nguyen Ngoc, V. Dinh Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[10] C. Hu, X. Hou, and Y. Lu, "Improving the architecture of an autoencoder for dimension reduction," in *Proc. IEEE 11th Int. Conf. Ubiquitous Intell. Comput. IEEE 11th Int. Conf. Autonomic Trusted Comput. IEEE 14th Int. Conf. Scalable Comput. Commun. Associated Workshops*, Dec. 2014, pp. 855–858.

[11] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 496–503.

[12] Y. Wang, H. Yao, S. Zhao, and Y. Zheng, "Dimensionality reduction strategy based on auto-encoder," in *Proc. 7th Int. Conf. Internet Multimedia Comput. Service (ICIMCS)*, 2015, pp. 1–4.

[13] D. C. Ferreira, F. I. Vazquez, and T. Zseby, "Extreme dimensionality reduction for network attack visualization with autoencoders," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–10.

[14] J. Zheng and L. Peng, "An autoencoder-based image reconstruction for electrical capacitance tomography," *IEEE Sensors J.*, vol. 18, no. 13, pp. 5464–5474, Jul. 2018.

[15] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," SNU Data Mining Center, Seoul, Korea, 2015.

[16] D. Oh and I. Yun, "Residual error based anomaly detection using auto-encoder in SMD machine sound," *Sensors*, vol. 18, no. 5, p. 1308, 2018.

[17] N. Sarafijanovic-Djukic and J. Davis, "Fast distance-based anomaly detection in images using an inception-like autoencoder," in *Proc. 22nd Int. Conf. Discovery Sci.*, 2019, pp. 493–508.

[18] G. Andresini, A. Appice, N. Di Mauro, C. Loglisci, and D. Malerba, "Exploiting the auto-encoder residual error for intrusion detection," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS PW)*, Jun. 2019, pp. 281–290.

[19] Ö. Yıldırım, U. B. Baloglu, and U. R. Acharya, "A deep convolutional neural network model for automated identification of abnormal EEG signals," *Neural Comput. Appl.*, pp. 1–12, Nov. 2018. [Online]. Available: https://link.springer.com/article/10.1007%2Fs00521-018-3889-z#citeas

[20] S. Araki, T. Hayashi, M. Delcroix, M. Fujimoto, K. Takeda, and T. Nakatani, "Exploring multi-channel features for denoising-autoencoder-based speech enhancement," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 116–120.

[21] J. Chen, B. Hu, L. Xu, P. Moore, and Y. Su, "Feature-level fusion of multimodal physiological signals for emotion recognition," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Nov. 2015, pp. 395–399.

[22] M. S. Hussain, R. A. Calvo, and P. Aghaei Pour, "Hybrid fusion approach for detecting affects from multichannel physiology," in *Affective Computing and Intelligent Interaction*. Berlin, Germany: Springer, 2011, pp. 568–577.

[23] R. K. Malaiya, D. Kwon, S. C. Suh, H. Kim, I. Kim, and J. Kim, "An empirical evaluation of deep learning for network anomaly detection," *IEEE Access*, vol. 7, pp. 140806–140817, 2019.

[24] Y. Hao, Y. Sheng, and J. Wang, "Variant gated recurrent units with encoders to preprocess packets for payload-aware intrusion detection," *IEEE Access*, vol. 7, pp. 49985–49998, 2019.

[25] S. T. Ikram, "Improving accuracy of intrusion detection model using PCA and optimized SVM," *J. Comput. Inf. Technol.*, vol. 24, no. 2, pp. 133–148, Jun. 2016.

[26] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Shallow neural network with kernel approximation for prediction problems in highly demanding data networks," *Expert Syst. Appl.*, vol. 124, pp. 196–208, Jun. 2019.

[27] D. Berman, A. Buczak, J. Chavis, and C. Corbett, "A survey of deep learning methods for cyber security," *Information*, vol. 10, no. 4, pp. 1–35, 2019.

[28] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[29] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3854–3861.

[30] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "$Deep-Full-Range$: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[31] S. Ali and Y. Li, "Learning multilevel auto-encoders for DDoS attack detection in smart grid network," *IEEE Access*, vol. 7, pp. 108647–108659, 2019.

[32] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," *Int. J. Secur. Appl.*, vol. 9, no. 5, pp. 205–216, May 2015.

[33] J. Yan, D. Jin, C. W. Lee, and P. Liu, "A comparative study of off-line deep learning based network intrusion detection," in *Proc. 10th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2018, pp. 299–304.

[34] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Comput. Secur.*, vol. 86, pp. 53–62, Sep. 2019.

[35] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Proc. ICONIP*, 2017, pp. 858–866.

[36] T. Kim, S. C. Suh, H. Kim, J. Kim, and J. Kim, "An encoding technique for CNN-based network anomaly detection," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2960–2965.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[39] A. Sharma, P. Malacaria, and M. Khouzani, "Malware detection using 1-dimensional convolutional neural networks," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS PW)*, Jun. 2019, pp. 247–256.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[41] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[42] Y. LeCun, K. Kavukcuoglu, and C. Farabet, "Convolutional networks and applications in vision," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 253–256.

[43] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham, Switzerland: Springer, 2018.

[44] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[45] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *Artificial Neural Networks—ICANN*, 2010, pp. 92–101.

[46] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," Sep. 2014, *arXiv:1409.4842*. [Online]. Available: https://arxiv.org/abs/1409.4842

[47] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. 2nd Int. Conf. Learn. Represent.*, Banff, AB, Canada, Apr. 2014. [Online]. Available: http://arxiv.org/abs/1312.4400

[48] N. Di Mauro, A. Appice, and T. M. A. Basile, "Activity prediction of business process instances with inception cnn models," in *Advances in Artificial Intelligence*. Cham, Switzerland: Springer, 2019, pp. 348–361.

[49] D. Guidici and M. Clark, "One-dimensional convolutional neural network land-cover classification of multi-seasonal hyperspectral imagery in the San Francisco bay area, California," *Remote Sens.*, vol. 9, no. 6, p. 629, Jun. 2017.

[50] S. Harbola and V. Coors, "One dimensional convolutional neural network architectures for wind prediction," *Energy Convers. Manage.*, vol. 195, pp. 70–75, Sep. 2019.

[51] M. Labonne, A. Olivereau, B. Polve, and D. Zeghlache, "A cascade-structured meta-specialists approach for neural network-based intrusion detection," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–6.

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

[53] H. Zenati, C. Sheng Foo, B. Lecouat, G. Manek, and V. Ramaseshan Chandrasekhar, "Efficient GAN-based anomaly detection," 2018, *arXiv:1802.06222*. [Online]. Available: http://arxiv.org/abs/1802.06222

[54] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

[55] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[56] X. Qu, L. Yang, K. Guo, L. Ma, T. Feng, S. Ren, and M. Sun, "Statistics-enhanced direct batch growth self-organizing mapping for efficient DoS attack detection," *IEEE Access*, vol. 7, pp. 78434–78441, 2019.

[57] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[58] Y. Yang, K. Zheng, C. Wu, X. Niu, and Y. Yang, "Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks," *Appl. Sci.*, vol. 9, no. 2, p. 238, 2019.

[59] R. Abdulhammed, H. Musafer, A. Alessa, M. Faezipour, and A. Abuzneid, "Features dimensionality reduction approaches for machine learning based network intrusion detection," *Electronics*, vol. 8, no. 3, p. 322, Mar. 2019.

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

[61] P. A. A. Resende and A. C. Drummond, "Adaptive anomaly-based intrusion detection system using genetic algorithm and profiling," *Secur. Privacy*, vol. 1, no. 4, p. e36, Jul. 2018.

[62] A. Ahmim, L. Maglaras, M. A. Ferrag, M. Derdour, and H. Janicke, "A novel hierarchical intrusion detection system based on decision tree and rules-based models," in *Proc. 15th Int. Conf. Distrib. Comput. Sensor Syst. (DCOSS)*, May 2019, pp. 228–233.

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

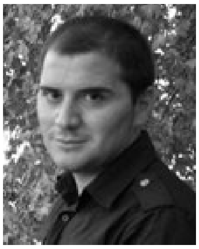
[64] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist. AISTATS*. Fort Lauderdale, FL, USA: JMLR.org, Jun. 2011, pp. 315–323.

[65] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2014, pp. 1–15.

GIUSEPPINA ANDRESINI graduated in computer science from the University of Bari Aldo Moro, Italy, in April 2018, discussing a Laurea thesis on data mining, where she is currently pursuing the Ph.D. degree with the Department of Computer Science. Her research activity mainly concerns deep learning, data mining, big data analytics, and cybersecurity. The applications include malware analysis and intrusion detection. She is involved in a research national project on intrusion detection. She is a member of the Organization Committee of the CyberChallenge.IT 2020 in the Department of Computer Science, University of Bari Aldo Moro.

ANNALISA APPICE received the Ph.D. degree in computer science from the University of Bari Aldo Moro, Italy. She was a Visiting Researcher at the University of Bristol, U.K., and at the Jozef Stefan Institute, Slovenia. She is currently an Associate Professor with the Department of Computer Science, University of Bari Aldo Moro. Her current research interests include data mining with spatio-temporal data, data streams and event logs with applications to remote sensing, process mining, and cybersecurity. On these topics, she has published more than 135 articles in international journals and conferences. She has been the Program Co-Chair of ECML-PKDD 2015 and ISMIS 2017. She is the Program Chair of the DS 2020. She has participated in the organization (as the Co-Chair) of several workshops on the topics machine learning and data mining. She is in charge of research units in two national projects (topic on remote sensing). She is serving/has served in the program committee of several international/national conferences. She is a member of the editorial board of the Data Mining and Knowledge Discovery (DAMI) and the *Journal of Intelligent Information Systems* (JIIS).

NICOLA DI MAURO received the Ph.D. degree from the University of Bari Aldo Moro, in 2005. He has been an Assistant Professor with the Department of Computer Science, University of Bari Aldo Moro, since 2005, where he is a member of the Machine Learning Group at the LACAM Laboratory. His main research interests are statistical relational learning, probabilistic deep learning, and machine learning, as well as their applications. He has participated in various European projects concerning these topics. He has published over 100 peer-reviewed technical articles published on International Journals and conference proceedings. He regularly serves on the PC (often at senior level) for several top conferences of artificial intelligence and machine learning, and is on the editorial boards of some international journals.



CORRADO LOGLISCI received the Ph.D. degree in computer science, in 2008. He is the Co-Supervisor of Ph.D. students enrolled at the Ph.D. Programme in computer science and mathematics of the University of Bari. He is currently an Assistant Professor with the Department of Computer Science, University of Bari "Aldo Moro," Italy. He has published several articles in refereed journals, international conferences, and workshops. He has participated in European and national research projects concerning Data Mining and Knowledge Discovery. His current research interests include data mining, text mining, and machine learning. He is serving/has served as a program committee member and reviewer for international and national conferences and journals. He has participated to the organization of the international workshops, winter schools. He is also a guest editor of several of special issue numbers for the *Journal of Intelligent Information Systems*.

DONATO MALERBA (Member, IEEE) received the M.Sc. degree in computer science from the University of Bari, Italy, in 1987. He is currently a Full Professor with the Department of Computer Science, University of Bari. He is also the Director of the Computer Science Department, University of Bari, and of the CINI Lab on Big Data. He has been in the Board of Directors of the Big Data Value Association and in the Partnership Board of the PPP Big Data Value. He has been responsible for the local research unit of several European and national projects, and received an IBM Faculty Award, in 2004. He was the Program (Co)-Chair of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA-AIE) 2005, the International Symposium on Methodologies for Intelligent Systems (ISMIS) 2006, SEBD 2007, the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD) 2011, and was the General Chair of the ALT/DS 2016. He is in the editorial board of several international journals. He has published more than 300 articles in international journals and conference proceedings. His research interests include machine learning, data mining, big data analytics, and their applications.