Wireless Networks (2024) 30:5059–5075
https://doi.org/10.1007/s11276-022-03214-3

Network intrusion detection system for DDoS attacks in ICS using deep autoencoders

Ines Ortega-Fernandez[1,3,4] · Marta Sestelo[2,3] · Juan C. Burguillo[4,5] · Camilo Piñón-Blanco[1,4]

Accepted: 13 December 2022 / Published online: 5 January 2023

Abstract

Anomaly detection in industrial control and cyber-physical systems has gained much attention over the past years due to the increasing modernisation and exposure of industrial environments. Current dangers to the connected industry include the theft of industrial intellectual property, denial of service, or the compromise of cloud components; all of which might result in a cyber-attack across the operational network. However, most scientific work employs device logs, which necessitate substantial understanding and preprocessing before they can be used in anomaly detection. In this paper, we propose a network intrusion detection system (NIDS) architecture based on a deep autoencoder trained on network flow data, which has the advantage of not requiring prior knowledge of the network topology or its underlying architecture. Experimental results show that the proposed model can detect anomalies, caused by distributed denial of service attacks, providing a high detection rate and low false alarms, outperforming the state-of-the-art and a baseline model in an unsupervised learning environment. Furthermore, the deep autoencoder model can detect abnormal behaviour in legitimate devices after an attack. We also demonstrate the suitability of the proposed NIDS in a real industrial plant from the alimentary sector, analysing the false positive rate and the viability of the data generation, filtering and preprocessing procedure for a near real time scenario. The suggested NIDS architecture is a low-cost solution that uses only fifteen network-based features, requires minimal processing, operates in unsupervised mode, and is straightforward to deploy in real-world scenarios.

Keywords Network intrusion detection system · Anomaly detection · Industrial control systems · Cyber-physical systems · Industrial cybersecurity · Deep autoencoder

1 Introduction

Modern industrial control systems (ICS) make use of advanced information technology to control cyber-physical processes, and are connected to external systems such as cloud environments. Furthermore, most ICS communication protocols are obsolete and lack inherent security, making them a prime target for cyberattacks. For example, Industrial Ethernet is a popular protocol, but it lacks fundamental security features such as encryption and authentication.

✉ Ines Ortega-Fernandez
ines.ortega@uvigo.es

Marta Sestelo
sestelo@uvigo.es

Juan C. Burguillo
j.c.burguillo@uvigo.es

Camilo Piñón-Blanco
cpinon@gradiant.org

[1] Galician Research and Development Center in Advanced Telecommunications (GRADIANT), Carretera do Vilar 56-58, 36214 Vigo, Spain

[2] Department of Statistics and O.R., SiDOR research group, Universidade de Vigo, 36310 Vigo, Spain

[3] CITMAga, 15782 Santiago de Compostela, Spain

[4] Escola de Enxeñaría de Telecomunicación, Universidade de Vigo, 36310 Vigo, Spain

[5] atlanTTic Research Center, Universidade de Vigo, 36310 Vigo, Spain

Attacks on cyber-physical systems have become a reality with the discovery of Stuxnet in 2010 [25]. Unlike attacks on information systems, cyberattacks have a physical impact on industrial devices, causing delays or stoppage in production. Reactive security methods (such as anomaly detection) should be used in addition to traditional security mechanisms currently in place on ICS, such as access control.

Specific solutions for safeguarding operational settings should handle the unique availability and real-time conditions that characterise industrial control systems. Taking this into consideration, we have developed an unsupervised framework for anomaly detection that is protocol and network agnostic. We enable the anomaly detection framework on network flow data to achieve this goal: real-time and availability limitations are harsh on field-level devices that are dedicated to executing only one activity. Furthermore, because field-level devices have limited memory and computing capability, it is frequently impossible to execute other software on them. For these reasons, deploying a host-based intrusion detection system is frequently difficult, leaving only a network-based implementation as a viable option.

A network intrusion detection system (NIDS) based on flow data can be implemented without knowledge of the network topology and without the need for any modification or manipulation of existing industrial equipment. Data can be collected, for example, by passively listening to a network switch's span port and capturing network flow data without interfering with the main operation of industrial devices. This method provides complete visibility of the industrial network, allowing real-time monitoring and detection of malicious network traffic. Furthermore, by using network flow data, instead of raw traffic captures or device logs (a frequent approach in ICS anomaly detection), the same anomaly detection framework and architecture may learn from any industrial network. Additionally, network flow data requires minimum processing before being employed in a mathematical model, and model characteristics do not require separate processing based on the network or the field-level protocols, lowering implementation costs.

The fundamental contribution of this work is a Deep Autoencoder-based NIDS framework that outperforms state-of-the-art results from alternative supervised algorithms in a more challenging learning environment. Furthermore, it outperforms the results of Isolation Forest [20], a popular approach to unsupervised anomaly detection. Despite the fact that deep autoencoders have been employed previously, we consider flow-based data from the perspective of an Industrial control system, analysing and demonstrating its suitability for a real-world deployment. Deep Autoencoders enable us to work on an unsupervised learning approach, allowing us to utilise the NIDS in real-world scenarios where labels are normally not provided during the training phase. Furthermore, without prior knowledge or significant study of the characteristics of the devices from the Industrial control system, the collection and preprocessing of network flow features is feasible in any industrial network. This enables the proposed NIDS to be deployed as a plug-and-play solution to anomaly detection in ICS. Despite the fact that network flow data contains less information than features based on individual network packet payloads, the proposed NIDS achieves excellent results in both a DDoS attack dataset and a real-world environment. We exploit the robustness, predictability, and repetitive patterns, that characterise industrial network flows, to achieve a high detection rate and a low False Positive rate. Furthermore, the NIDS can detect the effects of an attack on the behaviour of legitimate devices, allowing it to be used in critical infrastructures, where identification of malfunctioning equipment due to a cyberattack is essential.

The major contributions of this work are summarised as follows:

- This paper presents a cost-effective and scalable Network intrusion detection system for industrial control systems, based on network traffic features.
- We employ network traffic statistics to detect anomalous behaviour from distributed denial of service attacks on industrial situations, in contrast to earlier studies examined on Sect. 2, by means of deep autoencoders, an unsupervised Deep Learning approach.
- The proposed NIDS outperforms other approaches on the same dataset. A detailed comparison of the deep autoencoder results to Isolation Forest and state-of-the-art results using supervised methods, is performed, and information on the NIDS's applicability for real-world settings is provided.
- We showcase the performance of the proposed NIDS in a real scenario, analysing its suitability in terms of reported false alarms, data ingestion capabilities, and real time performance.

The remainder of the article is organised as follows: Sect. 2 presents previous works on anomaly detection in Industrial control systems using Machine Learning. Section 3 covers autoencoder neural networks and how to use for anomaly detection, introducing the chosen ensemble architecture of activation functions. Section 4 describes the proposed NIDS architecture, including the data collection and ingestion technique as well as the anomaly detection schema for real-world scenarios. The dataset used to assess the performance of the deep autoencoder is presented in Sect. 5, as well as the data filtering and feature extraction approach, a description of the Isolation Forest algorithm as the baseline model, and the model evaluation metrics.

Section 6 compares the Deep Autoencoder's performance detecting different flooding attacks in an industrial environment to Isolation Forest and state-of-the-art results, and exposes the results of the proposed framework in a real-world deployment. Finally, Sect. 7 summarises the key research contents, successes, and conclusions, as well as some future research prospects.

2 Related work

Due to the large diversity of devices and network protocols, securing ICS is a difficult undertaking. While typical security methods such as access control, encrypted communications, and firewalls can be effective in preventing basic attacks, they are not often used in industrial networks due to the use of obsolete equipment and protocols. In contrast, anomaly detection can detect ICS attacks by comparing the change to a profile of the industrial system's regular operation. This profile can be built using a variety of strategies and data sources.

The majority of research attention has been focused on detecting anomalies in industrial device logs. [22] investigated the use of process mining and compliance checking analysis to detect abnormalities in ICS, allowing the identification of long-running attacks, which gradually disrupt an industrial control system through a combination of anomalous activity sequencing. [7] identify patterns from sensor measurements and use them to categorise the status of an industrial plant using logical analysis of data (LAD), a combinatorial optimisation-based technique. [11] use plant design knowledge to generate highly connected discrete and continuous state variables. Then, using deep learning techniques, they model the dynamics of sensors and understand the functional connections between them using rule-based machine learning algorithms to detect abnormalities in a simulated water treatment facility (SWaT).

Deep autoencoders have recently been investigated in [10]. The authors proposed training an autoencoder utilising device data collected during normal operation, enabling for high detection rates of single and multi-point attacks. Although such implementation does not necessitate prior knowledge of the infrastructure, it does necessitate some understanding of how components are arranged in order to reduce training complexity (improving the accuracy of the model). [23] conducted a similar study using a dataset gathered from a gas pipeline. The dataset includes network information (such as IP addresses or packet length), MODBUS protocol command payloads (as the MODBUS function code), and data measurements from industrial equipment. They contrast the deployment of an autoencoder with other common machine learning (ML) models, emphasising the importance of appropriately establishing the decision threshold on important infrastructures to avoid False Negatives. Finally, in [30], an LSTM Autoencoder is constructed, which learns a pattern of the industrial control system by forecasting and reconstructing the input data and taking into account the temporal correlations between the various components.

Most research on the use of features generated from network traffic to detect anomalies focuses on specific protocols such as MODBUS/TCP or Profinet. The papers [12, 16, 34] utilise network traffic to establish a pattern of conventional MODBUS/TCP communications. [12] implement a deterministic finite automaton (DFA) for each communication between a human–machine interface and different programmable logic controllers (PLCs). [16] seeks to identify bogus MODBUS orders by learning and storing a pattern of typical traffic, taking advantage of ICS's resilience and stability. [31] recently proposed memory augment based on generative adversarial network (MeAEG-Net) where they propose an abnormal Industrial IoT traffic detection model based on generative adversarial networks to make the anomaly detection framework more resilient against noise in the training dataset. [32] propose a supervised NIDS based on sparse autoencoders and random forest to achieve high detection accuracy. First, the sparse autoencoder is used to extract the features to train the supervised random forest algorithm, employing an ANASYN over-sampling technique to address the class imbalance in the NDS-KDD dataset.

Similar methodologies were examined on non-industrial traffic in terms of current trends in anomaly detection on information systems: [21] employ an autoencoder ensemble architecture. From individual network packets, they compute 23 statistical features. These features are computed for different time windows, totalling 115 features. Finally, they execute hierarchical agglomerative clustering to construct distinct sub-instances, utilising correlation as the distance measure in the clustering process to verify that the grouped features capture normal behaviour. The ensemble architecture is then created by training an autoencoder for each cluster of features. [35] recently described a similar strategy employing Variational Autoencoders to detect anomalies on a traditional information system, using the CICIDS2017 dataset. They support their research with network flow data, attaining AUC-ROC values near 0.8 for specific attack scenarios. Our NIDS obtains AUC-ROC values more than 0.96, reinforcing the use of network flow data and Deep Autoencoders as an appropriate approach to anomaly detection.

To summarise, the previous approaches have various shortcomings, and either:

- Make use of a supervised environment.
- Expert understanding of the network's specific infrastructure is required.

- Make use of field-level protocol expert knowledge and/or the precise actions performed by the devices.
- Employs complex features that necessitate considerable processing during data gathering.
- Lacks a standardised approach that can be used across various industrial networks.
- Are protocol-specific and do not generalise to other networks or ICS.
- Do not analyse the performance of the proposed approach in a real-world scenario.

According to [1], the main challenges of modern NIDS are related to lower detection accuracy due to imbalance datasets, low performance or non-demonstrated performance in real-world environments, the use of expensive computation methods that require a lot of processing and computing resources, and the design of NIDS that fit the specific needs of industrial and Internet of Things environments.

To tackle these challenges, we propose an NIDS architecture based on network flow data that uses Deep Autoencoder to learn the normal behaviour of an Industrial Control System. Extraction of network flow features is available in any network with minimal processing, allowing the deployment of a probe for feature extraction in real-time, lowering the cost and complexity of NIDS implementation. Furthermore, we can apply the same approach to different ICSs without previous knowledge about the network architecture or the intrinsic field-level protocols. Regardless of how simple the model characteristics are, our NIDS can learn enough about the behaviour of industrial devices to improve state-of-the-art results. Furthermore, our method outperforms the results obtained with isolation Forest on the same dataset. We also showcase the performance of the proposed NIDS in a real industrial environment from the alimentary sector, achieving a very low rate of False positives, showing the suitability of the proposed NIDS for real-world scenarios.

3 Anomaly detection using deep autoencoders

Anomalies are *“data points which do not conform to the well-defined distribution of the normal behaviour”* [6]. Given ICS’s repeated and consistent behaviour, we can detect anomalies by developing a strong model of normal operation and utilising deep learning (DL) to create a representation of multi-dimensional data.

Deep learning, as a branch of machine learning, attempts to learn data representations by stacking many non-linear processing layers. DL focuses on learning from data structure, capturing important statistical aspects. These characteristics can be applied to classification, regression, and other problems. In this paper, we will use Deep Autoencoders in an unsupervised learning framework, which are a type of feed-forward neural network with fully connected hidden layers [3].

3.1 Autoencoder architecture

Autoencoders are made up of an encoder and a decoder that have been trained to reproduce their input while minimising the reconstruction error. The encoder extracts features from the input vector, while the decoder attempts to recreate the same input using the learnt features, which makes them suitable for anomaly detection by using fully connected layers and introducing some constraints on the learning process, which will be discussed below. Other applications of autoencoders include sequence learning by introducing long short-term memory (LSTM) neurons in the autoencoder architecture, dimensionality reduction, data generation through statistical inference (Variational Autoencoders), or image denoising (Convolutional Autoencoders).

Given an input vector $x_i^{m \times n}$ with m observations and n columns or features, being n the number of neurons on the input and output layer, and $l < n$ the dimension of the learned features (number of hidden units). It should be noted that there may be several hidden layers between the encoder and decoder layers. If there are multiple hidden layers, it is referred to as a Deep Autoencoder. The encoder transforms the input vector x into a hidden representation $h_j (j = 1, \ldots, l)$ through a non-linear mapping as follows:

$$h_j \ = \ \sigma\left(\sum_{i=1}^{n} W_{ij} \times x_i + b_j\right), \tag{1}$$

where σ is a non-linear activation function (AF), typically *sigmoid* or *tanh*, W_{ij} is the weight between the i neuron in the input and the j neuron in the output, and b_j is the bias of the j neuron in the hidden layer.

Then, the decoder maps the hidden (latent) representation back to the original representation:

$$\widetilde{x}_i \ = \ \sigma\left(\sum_{j=1}^{l} \widehat{W}_{ij} \times h_j + \hat{b_j}\right). \tag{2}$$

The autoencoder tries to optimise the hyper-parameters to minimise the reconstruction loss between x and its reconstruction $\widetilde{x}$, commonly using the average reconstruction error computed as the mean squared error (MSE) between the original and reconstructed input, such that

$$minimise \ \mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} ||x_i - \widetilde{x}_i||^2 \quad w.r.t. \ \theta, \tag{3}$$

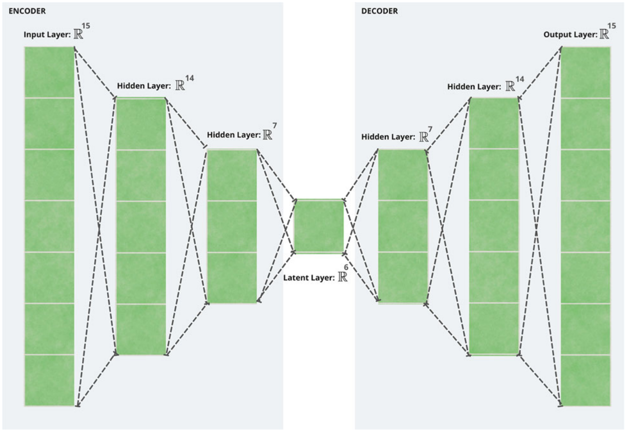
ENCODER
Input Layer: $\mathbb{R}^{15}$
Hidden Layer: $\mathbb{R}^{14}$
Hidden Layer: $\mathbb{R}^{7}$
Latent Layer: $\mathbb{R}^{6}$
DECODER
Hidden Layer: $\mathbb{R}^{7}$
Hidden Layer: $\mathbb{R}^{14}$
Output Layer: $\mathbb{R}^{15}$

Fig. 1 Resultant deep autoencoder architecture from the grid-search algorithm, showing the number of neurons per layer, and number of hidden layers

being θ the set of hyper-parameters $\theta = \{W,\ b\}$.

An autoencoder can detect anomalies if the model is trained with (ideally) only normal data or data with a low presence of attacks or abnormalities. The autoencoder will learn a latent representation of the normal data, and the reconstruction error $||x - \widetilde{x}||^2 \approx 0$.

We can calculate the Mean Squared Error between both vectors to compute the difference between the reconstructed output and the input, and then use the result as an anomaly score. We can use the Deep Autoencoder to perform binary classification if we set a threshold *varphi* based on the reconstruction error obtained over a normal dataset: values with a higher reconstruction error than the threshold will be classified as *anomalous*, while values with a lower reconstruction error will be classified as *norma*. It is important to note that the Deep Autoencoder training parameters must be tailored to each dataset or ICS. In this work, we used a *grid search* strategy to set the model hyper-parameters, such as the learning rate, the number of hidden layers and neurons per layer. Figure 1 shows the resultant Deep Autoencoder layer and neuron architecture obtained using the grid-search algorithm.

3.1.1 Activation functions

The choice of the activation function to utilise is essential and has an impact on the training process. As previously stated, Activation Functions are used to translate an input to an output using gradient processing and a weighted sum of the input and biases. Each AF has unique properties, and depending on the input data and the neural network's goal, some functions are better suited for specific tasks. The use of this AF will aid the objective function's convergence, hence speeding up the learning process. We employ an ensemble design to mitigate each function's downsides, such as the dying neuron and vanishing gradients issues. First, we use *tanh* as the AF in the first layer to take advantage of its 0-centred output. For the hidden layers, we add a sequence of *relu* layers to handle both the vanishing gradient and dying neurons problems: [29] demonstrated that a combination of both activation functions has a higher performance than models with only *tanh* or *relu*, since *tanh* provides values on $[-1, 1]$ range that help to address the dying problem of the subsequent *relu* layers. At last, the output layer is *tanh*, since *relu* is not recommended as AF in the output layer.

4 Network IDS architecture

Because we wish to assess the suitability of an NIDS using network flow data, we must first develop a data collection technique that works for both static files (like those in a public dataset) and online data gathering for real-world scenarios.

According to [5], a network flow is a sequence of packets from a source to a destination that gives statistical information that can be used to characterise the properties of network communications. In this paper, we suggest the usage of Argus [24], a strong open-source tool that operates as an superset of numerous flow data technologies (such as Netflow, Jflow, Qflow, or IPFIX) and generates network flow data regardless of the Layer 3 protocol employed. Argus allows us to either transform static *pcap* files from a certain dataset or capture network flow data in real time by directly listening to a network port.

For static network datasets, we can build network flows from a list of individual packets using the Argus toolkit. */etc/argus.conf* can be used to configure the set of features to extract from the *pcap* file:

pcap file:

```
> argus -r INPUT.PCAP -w OUTPUT.argus
> ra -r OUTPUT.argus > OUTPUT.csv
```

In a real-world environment, data network flows must be generated continuously in order for the anomaly detection framework to utilise them in training or prediction mode (in our case, a Deep Autoencoder). Figure 2 depicts a conceptual schema of our proposed NIDS architecture for real-world deployment, in which the input data for the anomaly detection framework is generated at the *Argus flow extractor* and collected and outsourced in streaming mode with Fluentd and Apache Kafka (*Data ingestion*). We recommended to place the *argus* probe as low in the Purdue Reference Architecture [33] as possible, at levels 2 or 3, to ensure full visibility of the industrial network traffic before it reaches the IT architecture. It is worth to mention

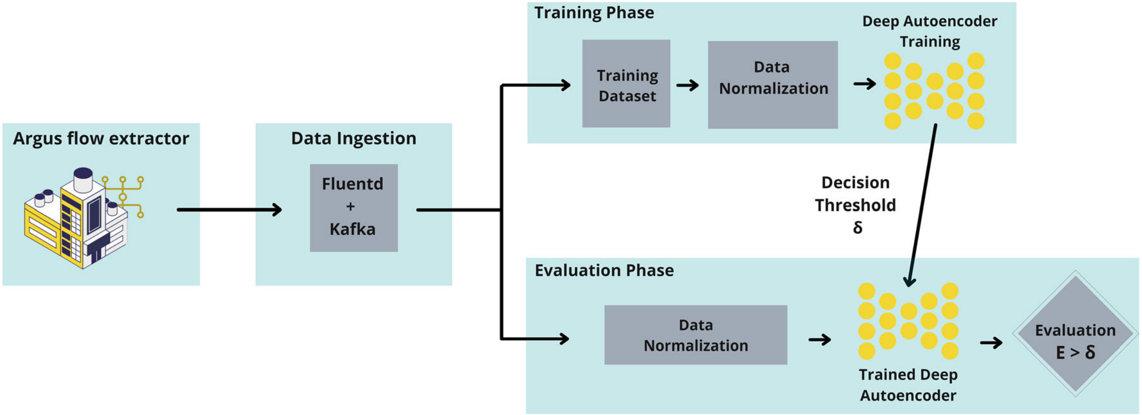

Fig. 2 Proposed NIDS conceptual diagram, detailing the network flow extraction, data ingestion and training and evaluation procedure

Table 1 Feature set description and data types

Feature name	Description	Data type
srcip	Source IP address of the flow	IP
smac	Source MAC address of the flow	MAC
dstip	Destination IP address of the flow	IP
dmac	Destination MAC address of the flow	MAC
proto	Protocol used in the communication	Integer
dport	Destination port of the flow	Integer
dpkts	Destination to source packets sent	Integer
dbytes	Destination to source bytes sent	Integer
spkts	Source to destination packets sent	Integer
sbytes	Source to destination bytes sent	Integer
pkts	Total packets sent	Integer
bytes	Total bytes sent	Integer
dur	Flow duration (in seconds)	Float
sintpkt	Source to destination inter-packet arrival time	Float
dintpkt	Destination to source inter-packet arrival time	Float

that a single *argus* probe can be configured to listen to multiple network interfaces, for example, to capture data from multiple Virtual LANs.

Once data is generated, it can be kept in a long-term data storage solution or ingested directly from Kafka into the anomaly detection framework in an online manner. To begin, we must train the Deep Autoencoder to recognise the pattern of regular communications in the Industrial Control System (*Training Phase*). Depending on the peculiarities of the ICS, the amount of data collected must be adjusted on a case-by-case basis. However, techniques such as Early Stop Callbacks (included in most deep learning libraries) can be used to immediately stop training after the network has learned the pattern of normal ICS communications, preventing overfitting. When the neural network training is completed, the decision threshold (*varphi*) is automatically calculated as the 95th quantile of the reconstruction error obtained over a new set of normal data. The anomaly detection system is now ready to evaluate incoming network flows (*Evaluation Phase*): an alarm will be raised if the reconstruction error of a specific input vector exceeds the decision threshold.

Table 1 describes the features chosen for the proposed NIDS. Our feature set contains the source and destination IP and MAC addresses of each network flow, the network protocol identifier according to the IANA specification [14], the communication's destination port, the number of packets and bytes sent, the mean inter-packet arrival times in a network flow, and the overall flow length. IP and MAC addresses are included to be able to train a single Deep Autoencoder for an entire network: otherwise, we would need to train a different neural network to learn the legitimate behaviour of individual devices. In this way, we are able to learn the behaviour of all the devices in the network with a single model, which simplifies model maintenance and deployment. In addition, it might allow the NIDS to detect other types of cyberattacks besides DDoS, such as IP/MAC spoofing attacks.

Because neural networks require numerical characteristics, we must transform all non-numerical values in the input vector to numeric equivalents and prepare them for use in a mathematical model (*Data Normalisation*). In Python, for example, the *ip* parameter of the *IPAddress* object provides the numeric equivalent of an IP address. For OSI Layer 3 protocols (such as ICMP) which do not have a *dport* value, *argus* automatically assigns a fixed value. In the case of ICMP flooding attacks, where the attacker sends ICMP echo request packets, *argus* assigns to the *dport* variable a fixed value.

Once the dataset has been transformed to its numerical counterpart, data normalisation is required to ensure that all characteristics (with varying sizes or value ranges) contribute equally to the model. Furthermore, feature scaling helps the model's converge to an optimal value by removing the mean and scaling each feature according to the interquartile range, as well as normalising each feature to be in the $[0, 1]$ range.[1]

5 Experimental setup

This section introduces the dataset used to assess the Deep Autoencoder's feasibility for anomaly detection. We also present the data preprocessing and filtering procedure used to obtain the necessary network flow features. Furthermore, the performance metrics used to evaluate the autoencoder's performance on each of the scenarios are introduced, with a focus on their suitability for performing an exact evaluation of the generated results.

5.1 Dataset description

The Cyber-security ICS dataset from the University of Coimbra [9] is used to assess the viability of our Deep Autoencoder. The dataset was created on a small-scale process automation scenario employing MODBUS/TCP equipment for research on the application of machine learning approaches to cyber-security in Industrial Control Systems. The dataset consists of normal operating traffic as well as a variety of Distributed Denial of Service (DDoS) attacks launched against the testbed.

The primary goal of Denial of Service attacks is to degrade the network's quality or availability. If the attack is launched from numerous machines at the same time, it is considered distributed. Depending on the network and the underlying device type, these attacks might disrupt or delay production lines, affecting the regular behaviour of industrial devices. There are three forms of DDoS attacks in this dataset: MODBUS query flood, TCP SYN flood, and Ping flood:

- MODBUS query flooding occurs when an attacker attempts to interrupt communications between legitimate devices by delivering fraudulent requests, in this case, false MODBUS query packets, to the victims. This exploit can disrupt the device's normal operation if the number of bogus MODBUS packets overwhelms legitimate packets [4].
- TCP SYN flooding: exploits the TCP protocol's three-way handshake: the victim gets a SYN message to establish a TCP connection and replies with an SYN-ACK, keeping an open port open and waiting for the confirmation ACK packet. The attacker, however, never responds with an ACK message, forcing the victim to keep a port open until the connection ends [13]. The attacker sends millions of SYN packets to the victim in an attempt to exhaust all available ports, preventing genuine devices from connecting to the target server.
- Ping flooding: Similar to MODBUS query flooding, the attacker's goal is to interrupt normal communication by sending bogus Internet Control Message Protocol (ICMP) ping packets to the victim [4].

5.2 Dataset preparation

The Cyber-security ICS dataset [9] contains both normal operating data and a set of DDoS attacks. The dataset is made up of several traffic captures (*pcap* files). It includes a group of files in which no attacks were carried out and just regular operation data was gathered. Furthermore, the authors carried out each of the three flooding attacks for varying lengths (1, 5 or 15 min) over traffic captures of 30 min and 1 h. It represents a total of eighteen attack scenarios, each with a particular mix of malicious and legitimate behaviour, and the class imbalance factor becomes more aggressive as the attack length increases (see Table 2).

Our Deep Autoencoder model is trained unsupervised, using only the set of normal operation data without the presence of an attacker, but without training labels. The training data contains 158 686 network flows and spans 7 h and 30 min of regular operation. We will choose a random split of 80% of the normal data set to utilise as the *training set*. The remaining 20% will be used to establish the decision threshold, making predictions over this subset of normal operation data: if the neural network is well trained, the reconstruction error should be low, and the decision threshold for detecting anomalies can be set as the 95th quantile of the reconstruction error.

To evaluate the performance of the Deep Autoencoder, we will use the DDoS attack dataset (which contains both

[1] It is worth to mention that the same scaling parameters must be used to normalise the training dataset and the new data to be evaluated.

Table 2 Characteristics of the different attack scenarios considered in the dataset: attack duration, capture duration and contribution of each class to the total number of samples per experiment

Attack	Attack Dur. (min)	Capture Dur.	Attack %	Normal %
MODBUS query flooding	1	30 min	56	44
	5		88	12
	15		97	3
	1	1 h	79	21
	5		95	5
	15		98	2
TCP SYN flooding	1	30 min	12	88
	5		39	61
	15		62	38
	1	1 h	7	93
	5		23	77
	15		46	54
Ping flooding	1	30 min	51	49
	5		84	16
	15		94	6
	1	1 h	78	22
	5		94	6
	15		98	2

normal operation and attack data) as our *test set*: first, we filtered the dataset to separate the actual attack packets from the normal operation data based on the MAC address of the attacker and/or the period when the attack was taking place.

5.3 Isolation forest for anomaly detection

In this paper, we will employ Isolation Forest [20] as a baseline model to test performance and discuss the Deep Autoencoder results. Isolation Forest (IF) is an unsupervised machine learning approach based on decision trees and random forests theory. Because of its ability to handle high-dimensional data, it is one of the most often employed models in anomaly detection situations. The key idea behind IF is that anomalous samples should be scarce and significantly different from the rest of the data, making them susceptible to isolation.

Although the authors of IF propose $t = 100$ and $psi = 256$ as the default configuration for the method, a grid search over several subsets of the data was performed to determine the best acceptable hyper-parameters for this case. Based on our findings, $t = 50$ and $psi = 8192$ were chosen as the settings for all tests to be performed. The underlying theoretical reason for setting this threshold to 0.5 is that the average path length for a data point implies an anomaly score of 0.5, so abnormal values (shorter path lengths) are expected to result in higher anomaly scores than 0.5 [2, 28], whereas normal samples are expected to result in lower anomaly scores than 0.5 [20].

5.4 Model evaluation metrics

The performance of the Deep Autoencoder implemented for anomaly detection is determined by how well the model performs on new data. True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN) are the four categories of model evaluation results for a particular threshold. The algorithm's output is correct if it correctly classifies an attack as such (TP) or a normal operation as such (TN), whereas wrong classifications are an attack classified as a normal operation (FN) or a normal operation classified as an attack (FP), based on a specific decision threshold.

From these values, we can derive additional metrics such as the *accuracy* (amount of instances, both positive and negative, classified correctly), *precision* (how many instances predicted as positive were attacks or the ratio of instances correctly classified as attacks), or *recall* (how many instances, out of all the true attack instances we have classified as positive). In imbalanced datasets, however, it is easy to achieve a high accuracy score by assigning all the samples to the majority class (in our case, as attacks). Unlike accuracy, the rates of TP (sensitivity, TPR) and TN (specificity, TNR) evaluate performance for each class independently, and class imbalance has no effect on them. TPR and TNR provide a measure of how both classes are correctly classified. They do not, however, take into account the proportion of examples assigned to a specific class. This component can be assessed using precision and recall, however those metrics are only applicable to the positive class [15].

If we wish to evaluate the model using a single metric, we can combine the rate of TP and TN, as well as precision and recall. The Geometric Mean (G-mean) takes into account the relative balance of the classifier's performance on both classes and is defined as a function of the rate of TP and TN [18]. On the other side, the F-Score, which computes the weighted harmonic mean of both values, can be used to combine precision with recall. We can use metrics that compute how effectively a classifier performs at separating classes to analyse the model regardless of the decision threshold. These metrics make no assumptions about the distribution of classes. One example is the Receiver Operating Characteristics (ROC), which calculates the True Positive Rate and the False Positive Rate for all possible thresholds and plots the potential couples in a curve [17] (the ROC Curve). We have a single metric to evaluate how the model performs for each threshold if we compute the Area Under the Curve (AUC). AUC of 1 denotes a perfect classifier, whereas 0.5 represents the lowest possible model (a model with no skill).

ROC analysis evaluates what can happen in various class imbalance circumstances, allowing us to choose the model with the best True Positive Rate and False Positive Rate. The precision-recall curve is an alternative to ROC curves: it computes the precision and recall values for a series of thresholds, measuring the trade-off between correctly classified positive values and the number of miss-classified negative instances, but it is again focused on the positive class and affected by class imbalance. AUC is an efficient way to assess a classifier's overall performance and to compare it to other algorithms. This is especially true in the case of extremely imbalanced datasets, where accuracy is skewed toward the positive class [15].

We are dealing with various levels of data imbalance in our experiments (as discussed in Sect. 5.2), therefore the results of each metric must be evaluated properly. Furthermore, we cannot use a single metric to analyse all scenarios, hence a more extensive study based on multiple metrics is required.

6 Results and discussion

Using the *training set*, we trained a single Deep Autoencoder model to learn the regular pattern of ICS communications. We trained and evaluated the neural network using a Lenovo Thinkbook with 16GB of RAM and an Intel(R) Core(TM) i7-10510U CPU running at 1.80GHz-2.30GHz. The model was implemented using the Keras[2] and Tensorflow[3] libraries for Python. The training set has a sample size of 126949, and the neural network training took 7.1 min with 50 epochs and a batch size of 128. The Deep Autoencoder performs very well, thanks to its simple architecture with a restricted number of neurons and hidden layers, as well as the usage of few features (see Fig. 1). In Sect. 6.1, we compare the Deep Autoencoder's performance when identifying different DDoS attacks from the *test set* to Isolation Forest and state-of-the-art results. Then, on Sect. 6.2, we show the performance of the suggested NIDS in a real industrial environment from the alimentary sector.

During the Deep Autoencoder model training, we automatically compute the decision threshold value as the 95th quantile of the reconstruction error, which in our experiments corresponds to $\varphi = 2.973 \cdot 10^{-4}$. We compute multiple threshold-based metrics, as well as the AUC of both the ROC and PR curves, for each evaluation experiment with the test set. As previously stated, the dataset is made up of eighteen experimental setups with varying degrees of class imbalance.

6.1 Experimental results

Two factors must be considered in Critical Infrastructures and Industrial Control Systems. First, because the attacks may influence the physical operation of the devices and hence have real-world effects, False Negatives must be avoided. Second, it is critical to identify not only the presence of an attacker, but also the repercussions of the attack on legitimate devices that may have an impact on the physical environment. This is the case with MODBUS query flooding attacks, which disrupt the operation of legitimate devices.

Table 3 shows the main metrics related to the performance of the Deep Autoencoder and the Isolation Forest baseline model across all the considered DDoS attack scenarios. A detailed view of the results achieved by the Deep Autoencoder and Isolation Forest (including other metrics of interest) can be found on Appendix We can see how the proposed NIDS can detect not only malicious behaviour from the attacker, but also anomalous behaviour of legitimate devices as a result of the attack, obtaining a high True Positives Rate (TPR) of over 0.988 across all scenarios. When compared to the IF performance (where the lower TPR is 0.303), we see that this method is not as capable as the Deep Autoencoder in detecting deviations

[2] https://keras.io/.

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

Table 3 Summarised results achieved with deep autoencoder and isolation forest for different DDoS flooding attacks

			Deep autoencoder				Isolation forest			
Attack	Attack Dur. (min)	Capture Dur.	TPR	FNR	AUC-ROC	AUC-PR	TPR	FNR	AUC-ROC	AUC-PR
MODBUS query flooding	1	30 min	**0.988**	0.012	0.983	0.988	0.762	0.238	**0.993**	0.951
	5		0.988	0.012	0.981	0.998	0.788	**0.212**	0.993	0.990
	15		0.985	**0.015**	0.976	1	0.786	0.214	0.994	0.999
	1	1 h	0.992	0.008	0.965	0.991	0.784	0.216	0.996	0.955
	5		0.988	0.013	0.964	0.998	0.766	0.234	0.995	0.992
	15		0.99	0.010	0.965	1	0.764	0.236	0.996	0.997
TCP SYN flooding	1	30 min	1	0	0.986	0.896	1	0	0.999	0.749
	5		1	0	0.980	0.967	1	0	0.999	0.836
	15		1	0	0.990	0.993	1	0	0.999	0.808
	1	1 h	1	0	0.989	**0.849**	**0.303**	0.697	0.999	0.769
	5		1	0	0.984	0.947	1	0	1	0.819
	15		1	0	0.989	0.988	1	0	1	0.664
Ping flooding	1	30 min	1	0	0.988	0.988	1	0	0.999	0.289
	5		1	0	0.986	0.997	1	0	0.999	0.731
	15		1	0	0.986	0.999	1	0	0.999	0.777
	1	1 h	1	0	0.966	0.990	1	0	1	**0.154**
	5		1	0	0.970	0.998	1	0	1	0.546
	15		1	0	**0.962**	0.999	1	0	1	0.77

The values in bold highlight the most relevant value per metric that are discussed in this section

from regular device operation, specially for modbus query flooding attacks. The Deep Autoencoder also performs exceptionally well in terms of avoiding False Negatives, with a maximum false negative rate (FNR) of 0.015 and perfect detection of TCP SYN and ping flooding attacks. The low FNR illustrates the viability of the suggested method in a critical infrastructure scenario where avoiding False Negatives is crucial: the maximum FNR across all scenarios for the Deep Autoencoder is 0.015. This criteria, on the other hand, is not met by IF in the MODBUS query flooding attacks, where IF achieves a minimum rate of FN of 0.212.

The Deep Autoencoder's low rate of False Negatives validates the algorithm used to calculate the decision threshold as a factor of reconstruction error: an appropriate strategy for anomaly detection on Industrial Control Systems is to determine the threshold by generating predictions on a set of normal operating data. While focused on recognising the attack behaviour, an automatic and unsupervised threshold was able to effectively differentiate the attack from regular behaviour (at the fall-out of raising some false alarms). However, one of the key disadvantages of IF is the lack of flexibility that adopting a constant default threshold may imply, given that it may not be the best solution in all situations. Furthermore, in the case of larger datasets, setting this threshold dynamically is an expensive operation that is still under investigation [19].

We can use the AUC-ROC to study how well the Deep Autoencoder performs in terms of False Positives and True Positives: for all instances, the Deep Autoencoder achieves very high values (over 0.962), demonstrating that it can perform very well in terms of True Positives and False Positives. The minimum value of this metric for IF is 0.993: IF performs well identifying the *attack* class, but at the expense of reporting a lot of False Negatives.

The analysis of precision-recall values (and thus the AUC-PR metrics) yields the following results: the Deep Autoencoder achieves AUC-PR values above 0.849. The minimum value for this metric is achieved on the 1-min TCP SYN flooding attack across a 1-h traffic capture, which has only 7% of positive samples, and therefore the precision metric is impacted. When we look at the results

Table 4 Summarised benefits and shortcomings of each analysed algorithm, including our implementations and the proposed classifiers from [8]

Algorithm	Unsupervised	High detection rate	Limited FP	Appropriate for real scenario	Affected by class imbalance	Requires large dataset
Deep autoencoder	•	•	•	•		•
Isolation forest	•	•		•	•	
kNN					•	
DT		•		•	•	
RF		•	•		•	
SVM					•	

obtained by IF, we see a similar pattern, the AUC-PR measure is influenced more severely by the imbalance towards the negative class, with a minimum AUC-PR value of 0.154. As a result, we can consider that the Deep Autoencoder is more resistant to flooding attacks, with very high values in the AUC-PR metric for all the scenarios evaluated.

At last, we can compare the performance of the proposed Deep Autoencoder to other approaches Using the same dataset. The authors of the Cyber-security ICS dataset [9] performed a comparative analysis in a supervised training framework with four different classifiers: k-Nearest Neighbours (kNN), Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF), using a total of 68 features extracted from the raw *pcap* files, such as packet timestamps, inter-packet arrival times, binary features defining which protocols were involved, and every field of the Ethernet, ARP, IP, ICMP, UDP, TCP and MODBUS over TCP headers [8].

They trained the classifiers to learn from both normal and attack behaviour (70 percent of the entire data was used for training) and evaluated the classifiers' ability to recognise such patterns. For each scenario, several training and testing experiments were carried out to examine how the increase in capture size and attack time influences the models. However, a supervised training environment difficulties the use of their approach in a real-world environment, where labels are usually not available. It is important to note the differences between the two approaches: we trained our system once in an unsupervised fashion, using only normal operation data, then tested it against each attack scenario independently. As a result, the training environment is different, as is the class imbalance factor. Furthermore, the dataset authors trained several classifiers with up to 68 features (including all of the protocol-level variables collected from the *pcap* files), but we only employed fifteen network-based features.

The supervised classifiers reached accuracy values of over 0.6 in certain scenarios, but as discussed in Sect. 5.4, this is a misleading result due to the dataset's imbalance factor increasing as attack time increases. In [8] a similar situation can be seen with the F1-Score: the F1-Score value decreases for the 5 min attack scenarios (from close to 1.0 to around 0.7, depending on the algorithm and attack) and then increases to close to 0.9 for the 15min attack experiments, because the classifiers have more data for training. The performance deterioration on the MODBUS query flooding attacks is not as aggressive, but an investigation by the authors determined that the improved results were caused by a feature with limited variance during the attacks, allowing all the methods to detect them quickly. This field is not in our feature set and has no effect on our experiments.

Our Deep Autoencoder performs better in this aspect, an is not as affected by class imbalance. The F1-Score values remains more stable with class imbalance: the bottom value for this measure is 0.8 on certain scenarios with significant class imbalance, and remains close to 0.9 on others, and does not suffer from performance degradation as attack time increases.

Table 4 summarises the benefits and shortcomings of each analysed algorithm: our Deep Autoencoder outperforms all unsupervised (IF) and supervised approaches on both accuracy and F1-Score (as shown on the detailed results in Appendix), and has a high detection rate while maintaining a very low rate of False Positives, which makes it adequate for real scenarios given a large enough dataset. However, more metrics would be required to provide a deeper comparison between our Deep Autoencoder and the supervised classifiers, since the accuracy and F1-Score are insufficient to evaluate the performance of the classifiers in all cases. For example, the precision and recall numbers achieved by the Deep Autoencoder (which have a higher variability with attack duration since they are exclusively focused on the positive class) cannot be compared to the state-of-the-art results. However, the Deep Autoencoder is proven to be more resilient than IF and the supervised classifiers against class imbalance, since it is trained on the normal class only. To the best of our knowledge, there are no other unsupervised approaches using the Cyber-security ICS dataset from the University of Coimbra that we can compare with our proposed approach.

6.2 Application to a real industrial scenario

The obtained results using a simulated dataset support the viability of a Network Intrusion Detection System based on a Deep Autoencoder on network flows for its use in a real-world environment: the anomaly detection framework reported extremely few False Negatives and False Positives in the experimental scenario. In this section we show how the proposed NIDS can be successfully applied to a real industrial scenario from the alimentary sector. To do so, we deployed the data acquisition, processing and anomaly detection infrastructure detailed on Sect. 4 in the operational network to collect network flows in real time. This will allow us to evaluate the performance of the anomaly detection framework, and to evaluate the suitability of the data acquisition, preprocesing and filtering procedure in a real time setting. However, since we are evaluating the NIDS in a real environment, we were not able to execute specific cyberattacks that might impact the operation of the real devices, and we are not aware of any anomalous behaviour or cyber-attack happening during the evaluation of the NIDS. Therefore, we will focus on analysing the ability of the anomaly detection framework to learn the normal behaviour of the industrial devices, studying the alarms reported to analyse the amount of False Positives.

First of all, *argus* flow extractor was connected to the span port of the main network switch, on the operational network of the industrial plant, to capture network flows in near real time. The designed network probe using *argus*, Fluentd and Apache Kafka was able to handle the amount of data generated by the industrial environment, producing 4000 network flows per minute, and redirecting the data into the anomaly detection framework by reading the network flows in near real time from Apache Kafka, and they were successfully retrieved without disrupting the ICS's normal operation. The network flows were also stored in OpenDistro[4] for latter analysis. It is worth to mention that this OT network only required the deployment of one *argus* probe, but if needed *argus* can be configured to listen to multiple network interfaces at the same time (for example, to capture traffic from different Virtual LANs), or multiple *argus* probes can be deployed in large networks to capture network flows in an scalable and distributed way.

Table 5 False positives reported on two different scenarios of the real industrial plant

Scenario	No. of samples	No. of FP	FPR (%)
ICMP UDP TCP	6230517	70686	1.1345
MODBUS S7Comm	2106415	10	0.0005

Focusing on the anomaly detection framework, we trained the Deep Autoencoder with 3 h of data collected in near real time from the network probe, generating a total sample of size $n_t = 226365$, which conforms our training set X_{train}. Since we aim to minimise the amount of False Positives, we set the decision threshold as the 99th of the reconstruction error obtained over a 10% random sample of X_{train}, which corresponds to $\varphi = 0.0670714903655881$.

Once the Deep Autoencoder was trained, we deployed it to perform predictions for two consecutive days. Table 5 summarises the results achieved in this scenario. We will focus on the amount of reported alarms (False Positives) to analyse the suitability of the proposed NIDS in a real-world industrial scenario. We will analyse two different environments: first, we will analyse all the TCP, UDP and ICMP network traffic to determine the amount of reported anomalies. Second, we will focus on the most common industrial protocols running on the network (see Fig. 3), by filtering the network traffic to those network flows with destination port 102 and 502, which corresponds to industrial protocols S7Comm, a Siemens proprietary protocol used by PLCs of the Siemens S7-300/400 family [26], and MODBUS TCP, which is commonly used for supervision and control of automation equipment, allowing the use of MODBUS messaging in an 'Intranet' or 'Internet' environment using the TCP/IP protocols [27].

In the first scenario, the NIDS reported a total of 70686 positives over 2 days, representing a 1.135% of False Positives. This is a very positive result, which validates the use of our proposed methodology for anomaly detection in a real scenario. The percentage of False Positives close to 1% is the expected one when using a threshold calculated at the 99th quantile of the reconstruction error, since we are allowing the NIDS to consider the remaining 1% as possible outliers in the training set. The False Positive rate is slightly higher than the achieved on the DDoS dataset, but this is expected in real-world scenarios. Studying the reported false alarms, we found out that most of them were related to information layer traffic. Figure 3 shows the number of network flows per destination port in the training set, and the number of anomalies detected per network protocol and destination port.

Most of the False Positives (91.99%) are reported on UDP traffic (protocol number 17), whereas less than 8% of

[4] https://opendistro.github.io/for-elasticsearch/.

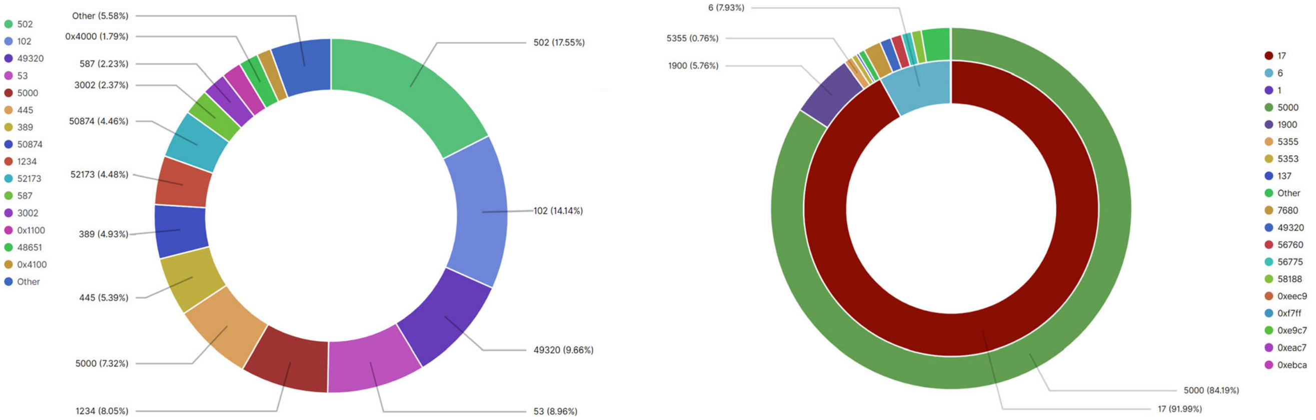

Fig. 3 Percentage of network flows per destination port in the training set (left) and percentage of detected anomalies per protocol and destination port in the test set (right)

the FP are related to TCP traffic (protocol number 6), and less than 1% to ICMP (protocol number 1). Focusing on the UDP anomalies, we can observe how 84.19% of the reported False Positives are related to UDP port 5000, which in this case it corresponds to VMWare related traffic. The next bucket corresponds to UDP port 1900 (5.76%) which is generated by SSDP multi-cast queries to the IP address 239.255.255.250 executed by Windows devices (such as SCADAs or HMIs). In the case of anomalous TCP traffic, most of it (19.4%) is on port 7680 which is related to Windows Update Delivery Optimization queries. Therefore, most of the False Positives is network traffic related to IT services, and can be filtered from the NIDS scope. This leads to our second analysis scenario, were we focus on the two main industrial protocols used in the network: Modbus and S7comm protocols. In this case, the amount of reported False Positives is much lower, reporting only 10 false alarms which represent a 0.0005% of the total network traffic.

These results validate the use of the proposed NIDS for anomaly detection on real-world ICSs: we can train an anomaly detector on the entire network traffic, and use it to detect DDoS cyberattacks on the operational network with high accuracy and a very limited amount of false alarms.

7 Conclusions

This study describes a network intrusion detection system (NIDS) architecture, that is supported by a deep autoencoder for anomaly detection, and is trained using only normal data and network flow features. Our Deep Autoencoder model has a high performance in many DDoS attack scenarios, improving the results from the state-of-the-art, and a baseline model, in a more challenging scenario. Our approach also has a low rate of both, False Negatives and False Positives, indicating that it is suitable for application in real-world scenarios.

We have derived fifteen network flow features that can be easily extracted from any network with minimum data processing. The most common approach in the state-of-the-art is the use of device logs, but they require extensive processing and background knowledge of the industrial scenario. We have described and evaluated a NIDS, capable of detecting both DDoS attacks and their effects on the behaviour of the industrial devices, by only using network flow features. This lowers the requirements to process and combine data sources with varying characteristics, such as device logs and/or network protocol fields, which can change across industrial networks and scenarios; making our proposal suitable for a production environment.

Unlike other works in the literature, we show the suitability of the proposed NIDS in a real industrial plant from the alimentary sector. We have designed, described and applied a data gathering, filtering and preprocessing strategy, that makes use of open-source tools, enabling a passive data collection approach, suitable for different network topologies; without interfering with the functioning of its industrial devices, allowing complete visibility of the ICS network in a scalable and distributed manner. Our proposed NIDS achieved a very low rate of False Positives, enabling its use for cyberattack and anomaly detection on real-world ICSs.

Future work includes the correlation and grouping of similar alerts provided by different devices in order to avoid multiple alerting, which could overwhelm an operator receiving the alarms. Furthermore, we want to explore the explainability of the model, concerning the results and the attack classification, once an anomaly has been detected.

Appendix

See Tables 6 and 7.

Table 6 Detailed experimental results achieved with Deep Autoencoder for different DDoS flooding attacks

Attack	Attack Dur. (min)	Capture Dur.	TN	FP	TP	FN	Precision	Recall	TPR	TNR	FPR	G-Mean	F1-Score	AUC-ROC	AUC-PR
MODBUS query flooding	1	30 min	11159	257	14267	179	0.982	0.988	0.988	0.977	0.023	0.983	0.985	0.983	0.988
	5		8738	227	71954	879	0.997	0.988	0.988	0.975	0.025	0.981	0.992	0.981	0.998
	15		2464	84	219933	3308	1	0.985	0.985	0.967	0.033	0.976	0.992	0.976	1
	1	1 h	3456	226	13555	115	0.984	0.992	0.992	0.939	0.061	0.965	0.988	0.965	0.991
	5		3084	197	71978	909	0.997	0.988	0.988	0.940	0.060	0.963	0.992	0.964	0.998
	15		2643	170	213755	2192	0.999	0.990	0.990	0.940	0.060	0.964	0.995	0.965	1
TCP SYN flooding	1	30 min	11320	315	1200	0	0.792	1	1	0.97	0.03	0.986	0.884	0.986	0.896
	5		10151	420	5996	0	0.935	1	1	0.96	0.04	0.980	0.966	0.980	0.967
	15		11151	238	17957	0	0.987	1	1	0.98	0.02	0.989	0.993	0.990	0.993
	1	1 h	22844	520	1197	0	0.697	1	1	0.98	0.02	0.989	0.822	0.989	0.849
	5		21920	705	5985	0	0.895	1	1	0.97	0.03	0.984	0.944	0.984	0.947
	15		21212	457	17948	0	0.975	1	1	0.98	0.02	0.989	0.987	0.989	0.988
Ping flooding	1	30 min	11626	283	11937	0	1	1	1	0.976	0.024	0.988	0.988	0.988	0.988
	5		11556	337	58762	0	1	1	1	0.972	0.028	0.986	0.997	0.986	0.997
	15		11504	324	176653	0	1	1	1	0.973	0.027	0.986	0.999	0.986	0.999
	1	1 h	3497	253	11937	0	1	1	1	0.933	0.067	0.966	0.990	0.966	0.990
	5		3535	228	59704	0	0.996	1	1	0.939	0.061	0.969	0.998	0.970	0.998
	15		3457	288	179152	0	1	1	1	0.923	0.077	0.961	0.999	0.962	0.999

Table 7 Detailed experimental results achieved with Isolation forest for different DDoS flooding attacks

Attack	Attack Dur. (min)	Capture Dur.	TN	FP	TP	FN	Precision	Recall	TPR	TNR	FPR	G-Mean	F1-Score	AUC-ROC	AUC-PR
MODBUS query flooding	1	30 min	9110	1436	10988	3433	0.884	0.762	0.762	0.864	0.238	0.322	0.784	0.993	0.951
	5		7451	768	57271	15434	0.987	0.788	0.788	0.907	0.212	0.271	0.821	0.993	0.99
	15		1664	636	175068	47702	0.996	0.786	0.786	0.723	0.214	0.466	0.821	0.994	0.999
	1	1 h	905	1955	10681	2950	0.845	0.784	0.784	0.316	0.216	0.732	0.795	0.996	0.955
	5		1592	933	55702	17057	0.984	0.766	0.766	0.630	0.234	0.532	0.801	0.995	0.992
	15		1399	774	164571	50955	0.995	0.764	0.764	0.644	0.236	0.522	0.801	0.996	0.997
TCP SYN flooding	1	30 min	8339	2428	1198	0	0.330	1	1	0.774	0	0.475	0.712	0.999	0.749
	5		8800	998	5987	0	0.857	1	1	0.898	0	0.319	0.968	0.999	0.836
	15		9633	929	17930	0	0.951	1	1	0.912	0	0.297	0.990	0.999	0.808
	1	1 h	18757	2880	362	832	0.112	0.303	0.303	0.867	0.697	0.201	0.226	0.999	0.769
	5		18460	2450	5977	0	0.709	1	1	0.883	0	0.342	0.924	1	0.819
	15		16858	3077	17919	0	0.853	1	1	0.846	0	0.393	0.967	1	0.664
Ping flooding	1	30 min	10068	952	11935	0	0.926	1	1	0.914	0	0.294	0.984	0.999	0.289
	5		9197	1763	58754	0	0.971	1	1	0.839	0	0.401	0.994	0.999	0.731
	15		9652	1134	176633	0	0.994	1	1	0.895	0	0.324	0.999	0.999	0.777
	1	1 h	1694	1083	11934	0	0.917	1	1	0.610	0	0.624	0.982	1	0.154
	5		1801	982	59692	0	0.984	1	1	0.647	0	0.594	0.997	1	0.546
	15		1385	1260	179131	0	0.993	1	1	0.524	0	0.690	0.999	1	0.77

Acknowledgements The authors would also want to thank Tiago Cruz from the Centre of Informatics and Systems, Department of Informatics Engineering (University of Coimbra, Coimbra, Portugal) for his help and comments on how to use and filter the Cyber-security ICS dataset.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was supported by the program Industrias del Futuro 4.0 of the Agencia Gallega de Innovación (IN854A 2019/15); the Ayudas Cervera para Centros Tecnológicos grant of the Spanish Centre for the Development of Industrial Technology (CDTI) under the project ÉGIDA (CER-20191012); the Xunta de Galicia (Centro singular de investigación de Galicia accreditation 2019-2022) and the European Union (European Regional Development Fund - ERDF); and the Grant MTM2017-89422-P (MINECO/AEI/FEDER, UE). Funding for open access charge: Universidade de Vigo/CISUG.

Data availability The datasets generated during and/or analysed during the current study are available in the IEEE Data Port repository, https://ieee-dataport.org/documents/cyber-security-modbus-ics-dataset

References

1. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies, 32*(1), e4150.
2. Ahmed, S., Lee, Y., Hyun, S.-H., & Koo, I. (2019). Unsupervised machine learning-based detection of covert data integrity assault in smart grid networks utilizing isolation forest. *IEEE Transactions on Information Forensics and Security, 14*(10), 2765–2777. https://doi.org/10.1109/TIFS.2019.2902822
3. Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning, 2*(1), 1–27. https://doi.org/10.1561/2200000006
4. Bhatia, S., Kush, N., Djamaludin, C., Akande, J. & Foo, E. (2014). Practical modbus flooding attack and detection. *Proceedings Of The Twelfth Australasian Information Security Conference*. AUS: Australian Computer Society, Inc.
5. Brownlee, N., Mills, C. & Ruth, G. (1999). Rfc2722: Traffic flow measurement: Architecture. USA: RFC Editor.
6. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computer Survey, 10*(1145/1541880), 1541882.
7. Das, T. K., Adepu, S., & Zhou, J. (2020). Anomaly detection in Industrial Control Systems using Logical Analysis of Data. *Computers and Security, 96*, 101935. https://doi.org/10.1016/j.cose.2020.101935
8. Frazão, I., Abreu, P. H., Cruz, T., Araújo, H., & Simões, P. (2019). Denial of service attacks: Detecting the frailties of machine learning algorithms in the classification process. In E. Luiijf, I. Žutautaitė, & B. M. Hämmerli (Eds.), *Critical information infrastructures security*. Springer International Publishing.
9. Frazão, I., Abreu, P., Cruz, T., Araújo, H. & Simões, P. (2019). Cyber-security modbus ICS dataset. IEEE Dataport. https://doi.org/10.21227/pjff-1a03
10. Gauthama Raman, M. R., Dong, W., & Mathur, A. (2020). Deep autoencoders as anomaly detectors: Method and case study in a distributed water treatment plant. *Computers and Security, 99*, 102055. https://doi.org/10.1016/j.cose.2020.102055
11. Gauthama Raman, M. R., & Mathur, A. (2022). AICrit: A unified framework for real-time anomaly detection in water treatment plants. *Journal of Information Security and Applications, 64*, 103046. https://doi.org/10.1016/j.jisa.2021.103046
12. Goldenberg, N., & Wool, A. (2013). Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection, 6*(2), 63–75. https://doi.org/10.1016/j.ijcip.2013.05.001
13. Horak, T., Strelec, P., Huraj, L., Tanuska, P., Vaclavova, A., & Kebisek, M. (2021). The vulnerability of the production line using industrial IoT systems under DDOS attack. *Electronics (Switzerland), 10*(4), 1–32. https://doi.org/10.3390/electronics10040381
14. Internet Assigned Numbers Authority (2019). Assigned internet protocol numbers. https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
15. Japkowicz, N. (2013). Assessment metrics for imbalanced learning. In: Imbalanced learning (p. 187-206). John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118646106.ch8
16. Kim, B.K., Kang, D.H., Na, J.C., Chung, T.M. & Kim, B.K. (2015). Detecting abnormal behavior in SCADA networks using normal traffic pattern learning. Lecture notes in electrical engineering. (Vol. 330, pp. 121–126). Springer. https://doi.org/10.1007/978-3-662-45402-2_18
17. Krzanowski, W. J., & Hand, D. J. (2009). *Roc curves for continuous data* (1st ed.). Chapman & Hall/CRC.
18. Kubat, M., Holte, R. & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. Machine Learning, **30**(2-3), 195–215. Copyright: Copyright 2020 Elsevier B.V., All rights reserved. https://doi.org/10.1023/a:1007452223027
19. Laskar, M. T. R., Huang, J. X., Smetana, V., Stewart, C., Pouw, K., An, A., & Liu, L. (2021). Extending isolation forest for anomaly detection in big data via k-means. *ACM Transactions on Cyber-Physical Systems*. https://doi.org/10.1145/3460976.
20. Liu, F.T., Ting, K.M. & Zhou, Z.-H. (2008). Isolation forest. Proceedings of the 2008 Uighth IEEE International Conference On Data Mining (p. 413–422). IEEE Computer Society, USA. https://doi.org/10.1109/ICDM.2008.17
21. Mirsky, Y., Doitshman, T., Elovici, Y. & Shabtai, A. (2018). Kitsune: An ensemble of autoencoders for online network intrusion detection. Network and Distributed System Security Symposium 2018 (NDSS'18).
22. Myers, D., Suriadi, S., Radke, K., & Foo, E. (2018). Anomaly detection for industrial control systems using process mining. *Computers and Security, 78*, 103–125. https://doi.org/10.1016/j.cose.2018.06.002
23. Nazir, S., Patel, S., & Patel, D. (2021). Autoencoder Based Anomaly Detection for SCADA Networks. *International Journal of Artificial Intelligence and Machine Learning, 11*(2), 83–99. https://doi.org/10.4018/IJAIML.20210701.oa6
24. QoSient (2020). Open argus. https://openargus.org/

25. Shakarian, P., Shakarian, J. & Ruef, A. (2013). Attacking Iranian nuclear facilities: Stuxnet. In: Introduction to cyber-warfare (pp. 223–239). Elsevier. https://doi.org/10.1016/b978-0-12-407814-7.00013-0
26. Siemens AG 2009-2022 (2022). What properties, advantages and special features does the s7 protocol offer¿. https://support.industry.siemens.com/cs/document/26483647/what-properties-advantages-and-special-features-does-the-s7-protocol-offer
27. Swales, A., et al. (1999). Open MODBUS/TCP specification. *Schneider Electric, 29*, 3–19.
28. Togbe, M. U., Barry, M., Boly, A., Chabchoub, Y., Chiky, R., Montiel, J., & Tran, V.-T., et al. (2020). Anomaly detection for data streams based on isolation forest using Scikit–Multiflow. In O. Gervasi (Ed.), *Computational science and its applications–ICCSA 2020*. Cham: Springer International Publishing.
29. Vu, L., & Nguyen, Q.U. (2019). An ensemble of activation functions in autoencoder applied to IoT anomaly detection. Proceedings - 2019 6th NAFOSTED Conference On Information And Computer Science, NICS 2019 (pp. 534–539). IEEE. https://doi.org/10.1109/NICS48868.2019.9023860
30. Wang, C., Wang, B., Liu, H., & Qu, H. (2020). Anomaly detection for industrial control system based on autoencoder neural network. *Wireless Communications and Mobile Computing, 2020*, 1–10. https://doi.org/10.1155/2020/8897926
31. Wang, T., Li, W., Rong, H., Yue, Z., & Zhou, J. (2022). Abnormal traffic detection-based on memory augmented generative adversarial IIoT-assisted network. *Wireless Networks, 28*(6), 2579–2595. https://doi.org/10.1007/s11276-022-02992-0
32. Wang, Z., Jiang, D., Huo, L., & Yang, W. (2021). An efficient network intrusion detection approach based on deep learning. *Wireless Networks*. https://doi.org/10.1007/s11276-021-02698-9
33. Williams, T. J. (1994). The purdue enterprise reference architecture. *Computers in Industry, 24*(2–3), 141–158.
34. Yoon, M.-K., & Ciocarlie, G. (2014). Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems. In: NDSS Workshop on Security of Emerging Networking Technologies.https://doi.org/10.14722/sent.2014.23012
35. Zavrak, S., & İskefiyeli, M. (2020). Anomaly-based intrusion detection from network flow features using variational autoencoder. *IEEE Access, 8*, 108346–108358. https://doi.org/10.1109/ACCESS.2020.3001350

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Ines Ortega-Fernandez received her B.Sc. degree in Computer Science and M.Sc. degree in Cybersecurity from Carlos III University of Madrid in 2018. She is pursuing a Ph.D. degree in Information and Communications Technology from the University of Vigo. Ms. Ortega-Fernandez is currently a Technical Manager of Data Analytics & AI at the Security & Privacy department of the Galician Research and Development Center in Advanced Telecommunications (GRADIANT). Her research focus is on the use of Artificial Intelligence to secure industrial control systems.

Marta Sestelo is Assistant Professor (accredited as Associate Professor) at the Department of Statistics and O.R. at the University of Vigo. In 2013, she received her PhD in Statistics from this university. She has taken part in 27 research projects (national and international), presented 49 conference papers, developed 6 R packages and, so far, written 20 articles either published or accepted on JCR journals. She is a regular reviewer of international journals such as Journal of Statistical Software. She has broad experience on the development of new techniques, methodologies and algorithms linked with data science and artificial intelligence.

Juan C. Burguillo received the M.Sc. degree in Telecommunication Engineering in 1995, and the Ph.D. degree in Telematics in 2001; both at the University of Vigo (Spain), where he is currently an associate professor at the Department of Telematic Engineering, and a researcher at the AtlanTTic Research Center in Telecommunications. He has directed and participated in several R&D projects in national and international calls. He has published more than one hundred papers, in international refereed journals and conference proceedings. He is a regular reviewer of several international journals, and the area editor of the journal of Simulation Modelling Practice and Theory (SIMPAT).

Camilo Piñón Blanco received his B.Sc. degree in Telecommunication Technologies Engineering (mention in Telematics Engineering) from University of Vigo (Spain) in 2021. He is currently a research engineer at GRADIANT and its pursuing a M.Sc. in Telecommunication Engineering from the University of Vigo. His main interests are Machine Learning and software engineering.