Journal of Information Security and Applications 79 (2023) 103647

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

VAASI: Crafting valid and abnormal adversarial samples for anomaly detection systems in industrial scenarios

Angel Luis Perales Gómez [a,*], Lorenzo Fernández Maimó [a], Alberto Huertas Celdrán [b], Félix J. García Clemente [a]

[a] Departamento de Ingeniería y Tecnología de Computadores, University of Murcia, Espinardo, Murcia, Spain
[b] Communication Systems Group CSG, Department of Informatics IfI, University of Zurich, CH 8050 Zürich, Switzerland

ARTICLE INFO

Keywords:
Adversarial attacks
Anomaly detection
Deep learning
Explainable artificial intelligence
Industrial control systems

ABSTRACT

In the realm of industrial anomaly detection, machine and deep learning models face a critical vulnerability to adversarial attacks. In this context, existing attack methodologies primarily target continuous features, often in the context of images, making them unsuitable for the categorical or discrete features prevalent in industrial systems. To fortify the cybersecurity of industrial environments, this paper introduces a groundbreaking adversarial attack approach tailored to the unique demands of these settings. Our novel technique enables the creation of targeted adversarial samples that are valid within the framework of supervised cyberattack detection models in industrial scenarios, preserving the consistency of discrete values and correcting cases where an adversarial sample transitions into a normal one. Our approach leverages the SHAP interpretability method to identify the most salient features for each sample. Subsequently, the Projected Gradient Descent technique is employed to perturb continuous features, ensuring adversarial sample generation. To handle categorical features for a specific adversarial sample, our method scrutinizes the closest sample within the normal training dataset and replicates its categorical feature values. Additionally, Decision Trees trained within a Random Forest are utilized to ensure that the resulting adversarial samples maintain the essential abnormal behavior required for detection. The validation of our proposal was conducted using the WADI dataset obtained from a water distribution plant, providing a realistic industrial context. During validation, we assessed the mean error and the total number of adversarial samples generated by our approach, comparing it with the original Projected Gradient Descent method and the Carlini & Wagner attack across various parameter configurations. Remarkably, our proposal consistently achieved the best trade-off between mean error and the number of generated adversarial samples, showcasing its superiority in safeguarding industrial systems.

1. Introduction

Currently, industrial activities account for a large percentage of the economy of countries. In order to increase productivity and, thus, gain a competitive advantage, many industries have engaged in a race to automate their production processes. This race brought with it the Industry 4.0 paradigm [1], which advocated the introduction of devices with processing capacity together with advanced techniques such as Big Data, the Internet of Things (IoT), or Artificial Intelligence (AI) [2]. Most of the new devices introduced are categorized as Cyber-Physical Systems (CPS) since they can process information in the cyber layer and exert an impact on the physical world. However, in the Industry 4.0 paradigm, the human activities carried out by workers were not considered and, therefore, we are currently witnessing a new paradigm shift where workers are placed at the center of production processes, and all techniques and devices have to take this into account. This new paradigm is known as Industry 5.0 [3].

In this new paradigm, not only do we have to verify that industrial systems work correctly to satisfy the production processes, but we also have to ensure that these systems pose no risk to the physical safety of workers in any scenario. In this context, Anomaly Detection (AD) systems based on Machine Learning (ML) and Deep Learning (DL) have gained importance. These systems are capable of learning the boundaries of an industrial system's typical behavior, flagging anything outside of this norm as a potential security threat [4].

Although AD systems based on ML/DL have shown adequate performance to detect cyberattacks in industrial scenarios [5,6], it is necessary to remember that all systems supported by ML and DL techniques are vulnerable to evasion attacks [7]. These attacks are a

* Corresponding author.
E-mail addresses: angelluis.perales@um.es (A.L. Perales Gómez), lfmaimo@um.es (L. Fernández Maimó), huertas@ifi.uzh.ch (A. Huertas Celdrán), fgarcia@um.es (F.J. García Clemente).

https://doi.org/10.1016/j.jisa.2023.103647

Available online 4 November 2023

subtype of adversarial attack that consists in crafting specially modified samples, called adversarial samples, that are misclassified by the AD system. In particular, those evasion attacks that convert an abnormal sample to a sample considered as normal by the AD system are of special interest since they can craft adversarial samples that reach the target device and cause an impact on the physical world.

However, adversarial attacks lack the ideal characteristics to be deployed in industrial environments. The main challenge is that these attacks are mainly designed for ML/DL models focused on computer vision and, therefore, they only work with continuous data. In particular, most adversarial attacks rely on the gradient of ML/DL models; therefore, when applied to discrete data, the resulting adversarial samples contain inconsistent values, i.e., continuous value in a feature that only accepts discrete values. This makes it impossible to extrapolate these attacks to other scenarios, such as industry, where tabular data is used, including continuous and categorical data. The second challenge is that the techniques used to craft adversarial samples modify all the features of these samples. On the one hand, this causes the error between the original and adversarial samples to be larger. On the other hand, features not necessary to craft adversarial samples are also modified. Together, this makes these attacks more easily detectable by an expert.

Considering the aforementioned limitations, it becomes imperative to devise a method capable of conducting adversarial attacks using established gradient-based techniques while handling discrete features with the necessary care. Discrete features often come with value restrictions to maintain internal coherence. Moreover, our aim is to establish a mechanism for detecting instances where our adversarial attack may have been excessive, inadvertently transforming an anomalous sample back into a clearly normal one. Therefore, to overcome the previous challenges, this work presents the following contributions:

- A new targeted adversarial attack called Valid and Abnormal Adversarial Samples for Industrial Scenarios (VAASIS) specially designed to attack industrial systems. This attack generates adversarial samples for supervised models and it is based on the most efficient and powerful attack, i.e., the Projected Gradient Descent (PGD) method [8]. The proposal relies on interpretability techniques to determine the most important features of each sample and launch PGD to modify only those features. In addition, the proposed attack identifies categorical features and modifies them so that the final result is consistent. Finally, the attack also recognizes those samples that lost their abnormal behavior after applying PGD and recovers it.
- Validation of the proposed attack in a realistic industrial environment based on a water distribution plant. In particular, the WADI [9] dataset containing both continuous and categorical data is used. Compared to PGD and Carlini & Wagner (CW), our attack achieved the best balance between the number of generated adversarial samples and the error caused to craft them, resulting in samples difficult to detect by experts.

The remainder of this paper is structured as follows. Section 2 discusses the different works focused on adversarial attacks and their application in industrial scenarios. Section 3 illustrates the conditions that need to be met by the generated adversarial samples together with the requisites and information needed by an attacker to deploy the attack. In Section 4 the adversarial attack proposed to generate consistent and valid adversarial samples in industrial scenarios is detailed. Section 5 details the experiments carried out. Finally, Section 6 presents the conclusions and the future work.

2. Related work

This section reviews the different solutions in the literature to execute adversarial attacks, especially the evasion one. On the one hand, we review techniques to launch adversarial attacks, mainly focused on Computer Vision problems. On the other hand, we review proposals that rely on such techniques to deploy adversarial attacks in industrial scenarios.

To put it in context, an evasion attack involves crafting samples misclassified by the AD system. Those are called adversarial samples, and several techniques exist to craft them. Depending on the adversarial sample generated, the adversarial attacks can be classified as targeted or untargeted. If the attack is intended to generate an adversarial sample of a specific class, we consider it as targeted. Otherwise, it is an untargeted adversarial attack.

The most powerful techniques to craft adversarial samples are those based on the gradient of the model. The first well-known technique proposed was the Fast Gradient Sign Method (FGSM) [10]. This method computes a noise vector that is added or subtracted to original samples, depending on whether the attack is configured as targeted or untargeted. The noise vector is determined using the sign of the gradient of the cost function with respect to the target label and modulated by a perturbation parameter. There are several modifications to the original FGSM algorithm. For example, if l_1 or l_2 norm is used instead of the sign of the gradient, the attack is commonly known as the Fast Gradient Method (FGM). In general, all techniques based on the gradient will consider the original sample as the center of a l_p sphere, where l_p indicates the distance metric and the radius of the sphere will be the maximum allowed disturbance. The resulting adversarial sample will be another sample inside the sphere. From FGSM and FGM, more elaborated adversarial attacks were proposed. The natural evolution of the aforementioned adversarial attacks calculates adversarial samples iteratively. This attack is called Basic Iterative Method (BIM) [11], and, like FGSM, it can be targeted and untargeted. Another attack derived from the original FGSM is the PGD [8], which is similar to BIM with two exceptions. The first is that the PGD initializes the example to a random point inside the l_p sphere, while BIM initializes to the original sample. The second difference is that the result obtained by PGD is projected to the l_p sphere. Another popular adversarial attack is Deepfool [12], which finds the minimal perturbation needed to change the sample class. In contrast to other reviewed methods, Deepfool can only be configured as untargeted. In particular, the algorithm computes the orthogonal projection of the sample onto the separation affine hyperplane, which corresponds to the minimal perturbation. Jacobian-based Salience Map Attack (JSMA) [13] is another adversarial attack based on computing the forward derivatives to craft the adversarial samples. In particular, this attack allows an attacker with knowledge about the network architecture to construct saliency maps that identify features of the input that most significantly impact output classification. Finally, one of the most powerful adversarial attacks is CW [14]. This attack can generate high-quality adversarial samples, but the cost to generate them is also high. Although the authors suggested a series of relaxation, the procedure is still costly.

The previous techniques were developed with a special focus on Computer Vision problems; therefore, they deal with continuous data. However, without any modification, those techniques fail when applied to environments that produce discrete data, such as industrial settings. Besides, those techniques modify all the features, and therefore, the adversarial sample is more easily detectable by an expert.

To solve those problems, several authors proposed different solutions. For example, the authors of [15] proposed a new adversarial attack based on BIM that can determine, through a mask, the specific features that can be modified. The attack was designed with industrial environments in mind, and it also dealt with categorical data. However, the approach employed was limited and consisted in selecting features to modify, launching an adversarial attack, and approximating the modified categorical features to the nearest integer. Besides, the authors did not ensure that the samples generated are abnormal; therefore, many adversarial samples generated can lose their abnormal behavior. The authors validated their solution using the Electra dataset, which was collected from an Electric Traction Substation. Furthermore, the authors demonstrated the importance of crafting valid adversarial samples in data networks, especially in the industrial setting. In particular, the authors stated that if an adversarial attack modifies a categorical field crucial to routing the packet, the attack could not be successfully accomplished since it would not reach its destination.

The authors of [16] presented another solution focused on IoT scenarios where they proposed a rule-based mechanism to craft adversarial samples whose features contained valid values. However, the solution was not focused on time-series data, and the adversarial attack did not ensure that the samples generated were abnormal. Besides, the valid values are selected without considering the distribution of the normal samples. Another solution focusing on IoT is presented in [17]. The authors conducted tests involving various adversarial attacks (FGSM, BIM, and PGD) and their impact on several DL models. However, similar to the original FGSM, BIM, and PGD attacks, the methods introduced in this study struggled with handling discrete features or time-series data. Furthermore, these attacks failed to guarantee the generation of valid samples. A more sophisticated approach is presented in [18], where the authors developed a white-box adversarial attack within the context of a smart home. Specifically, the authors generated adversarial samples for smart meters installed in residential homes. The solution proposed by the authors is based on the backpropagation algorithm. For each iteration, the method involved extracting the gradient, normalizing it, and adding it to the adversarial sample that was being generated. The authors concluded that their approach yielded statistically indistinguishable adversarial samples from the original ones. Nonetheless, this technique also lacks support for time series data, and was unable to generate valid samples when dealing with categorical features, potentially causing the generated samples to lose their anomalous behavior.

The authors of [19] studied how adversarial samples generated by JSMA impact supervised models in industrial settings. The authors proposed a gray-box approach where the attacker has no knowledge of the model but has access to the full dataset and knowledge about the features. Due to the transferability property [20] of adversarial samples, the authors trained two substitute models and crafted adversarial samples using such models. In particular, the authors trained Random Forest (RF) and J48 models. Using the JSMA attack, the attacker modified only specific features instead of introducing perturbation to all the features. However, the approach adopted by the authors did not consider crafting valid categorical features and did not ensure that adversarial samples retained their abnormal behavior. The authors validated their proposal in an authentic power system and concluded that the performance of RF and J48 decreased by 6 and 11 percentage points, respectively, when adversarial samples were present. Another example of an adversarial attack in industrial scenarios is presented in [21], where the authors proposed a novel attack focused on autoregressive models. In particular, the attack is based on two steps. The first one is called L_0 Prediction Attack and is in charge of predicting the next values for discrete values using the model under attack. The second one, called the L_0 Optimization Attack, is in charge of modifying continuous features using an optimization algorithm to craft samples that the autoregressive model will misclassify. Finally, the adversarial sample is formed by joining the discrete features predicted before and the continuous features crafted previously. Although this attack considers the problem when crafting categorical features, it does not assure that generated adversarial samples retained their abnormal behavior. Another example of an adversarial attack applied to industrial scenarios is presented in [22]. In this solution, the authors presented a Universal Adversarial Attack, which they defined as a procedure to find a special imperceptible noise to fool regression models. The authors tested their approach using the NASA turbofan engine dataset, demonstrating that adding universal adversarial perturbation to any input data instance increased the error in the output predicted by the model. However, although this approach can be applied to time-series data, it cannot be used when the dataset contains categorical data. Furthermore, the author's approach did not consistently generate valid samples in industrial scenarios. In [23] another solution suitable for industrial scenarios is presented. Specifically, the authors proposed two strategies for generating adversarial time-series samples that degrade the detection performance of AD models. The first method is iterative and relies on the sign of the gradient of the model. The second method is similar but involves determining which features to modify based on their importance. However, this importance is determined in a straightforward manner, i.e., by slightly altering each feature to determine which one induces the greater perturbation in the model's output. Furthermore, akin to most of the solutions reviewed in this section, the method did not account for the presence of categorical features, and the generation process could potentially result in samples losing their abnormal behavior.

Table 1 summarizes the solutions reviewed in this section. To the best of our knowledge, our solution is the only one that crafts valid samples in industrial scenarios, and, in addition, it is capable of evaluating whether the sample remains abnormal. Furthermore, if a sample loses its abnormal behavior, our proposal can recover it. Other solutions mentioned above also deal with discrete data but adopt a naive approach or do not consider that a sample can lose its abnormal behavior after applying an adversarial attack.

3. Problem statement

In this section, we detail the required conditions to generate valid adversarial samples in the industrial environment, together with the information and requisites needed by an attacker.

3.1. Conditions to generate useful adversarial samples

Unlike other fields, such as computer vision, adversarial samples generated in industrial environments must not only be misclassified by the AD system but also be valid samples. Furthermore, the adversarial samples generated in these scenarios must also preserve their abnormal behavior. However, adversarial attacks in the literature were originally intended for computer vision; therefore, they cannot generate adversarial samples that meet the two conditions mentioned above.

First, and before the AD system comes into play, the goal is to generate valid samples in an industrial environment. If these samples are not generated correctly, there is a risk that the sample will not reach its target industrial device and, therefore, will not end up impacting the physical world. For example, imagine an industrial water treatment plant with sensors and actuators. Among them, we can find pumps and valves that control the entry and outlet of water to and from storage tanks. These devices can only be in two states: open and closed for valves, and on and off for pumps. Any other value is invalid and can be easily detected by a rule system prior to the anomaly detector.

Second, after applying an adversarial attack, some of the generated samples had actually become normal samples, being now harmless to the industrial system. Fig. 1 illustrates a simplified example in an industrial setting where an adversarial attack is applied on a binary classification model. In particular, Fig. 1 shows two probability density functions (p.d.f), the boundary of the AD model, and three samples where an adversarial attack was applied. In this example, we assume that samples represented by a circle shape belong to the abnormal class and can cause an impact on the physical world. In contrast, samples represented by an x belong to the normal class and are harmless for the industrial system. After applying the adversarial attack and the AD system has detected them as normal, three cases may happen. The first case is illustrated by sample a, which retains its abnormal behavior because it belongs to the abnormal distribution. The second case is illustrated by sample b, which does not belong to the normal or abnormal distribution. Finally, the last case is shown by sample c, which belongs to the normal distribution and, therefore, it is an actual normal sample. Taking this into account, the only valid adversarial sample that impacts the physical world is sample a because it retains its

Table 1
Comparison of related work focused on adversarial attacks.

Solution	Focused on	Target model	Deals with discrete values?	Preserve abnormal behavior?	Validated with time-series data?
[10]	Computer Vision	models based on gradient	×	×	×
[11]	Computer Vision	models based on gradient	×	×	×
[8]	Computer Vision	models based on gradient	×	×	×
[12]	Computer Vision	models based on gradient	×	×	×
[13]	Computer Vision	models based on gradient	×	×	×
[14]	Computer Vision	models based on gradient	×	×	×
[15]	Industrial Scenarios	Classification models	✓	×	×
[16]	IoT Scenarios	Classification models	✓	×	×
[17]	IoT Scenarios	Classification models	×	×	×
[18]	IoT Scenarios	Classification models	×	×	×
[19]	Industrial Scenarios	models based on gradient	×	×	×
[21]	Industrial Scenarios	Autoregressive models	✓	×	✓
[22]	Industrial Scenarios	models based on gradient	×	×	✓
[23]	Industrial Scenarios	models based on gradient	×	×	✓
Ours	Industrial Scenarios	Classification models	✓	✓	✓

abnormal behavior, while samples b and c do not belong to abnormal samples distribution and, therefore, they lose their abnormal behavior.

Without considering the previous conditions, the adversarial samples generated by an attack will not impact the industrial scenarios. For example, if the sample contains invalid categorical features, it will not reach its target industrial device. Conversely, if the sample is converted into a normal one after applying the adversarial attack, it will not cause an impact on the physical world.

3.2. Requisites needed by an attacker

Before carrying out the attack, the attacker must access certain information. First of all, the attacker needs to have access to the supervised AD model. In particular, since the attack we propose is based on the PGD method, which uses the gradient of the models, the attacker needs access to an AD gradient-based model. The required AD model needs to have a proper performance on evaluation. Otherwise, the generated adversarial samples will be meaningless and not valid. To measure the AD model performance, it is common to use metrics such as precision, recall, and F1-score, which are defined based on True Positive, True Negative, False Positive, and False Negative:

- True Positive (TP): Shows the number of anomalies properly detected by the model.
- True Negative (TN): Shows the number of non-anomalies properly classified.
- False Positive (FP): Shows the number of non-anomalies wrongly classified as anomalies.
- False Negative (FN): Shows the number of anomalies wrongly classified as non-anomalies.

Then, precision, recall, and F1-score are defined as follows:

- Precision: Indicates what portion of the detected anomalies is real anomalies.
$$Precision = \frac{TP}{TP + FP}$$
- Recall: Indicates what portion of the real anomalies is detected.
$$Recall = \frac{TP}{TP + FN}$$
- F1-score: Is the harmonic mean between recall and precision and is a trade-off between precision and recall.
$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In addition, since the attacker will modify specific features during the attack, he or she also needs access to the features evaluated by the supervised model. In the worst case, the attacker can access datasets used to train and test the supervised model. The dataset partitioning process is the responsibility of the administrators, and it must be carried out in such a way that the temporal coherence of the data generated in industrial environments is preserved, as recommended by [24]. The training dataset must be used to train the AD system, while the test dataset must be used to test the AD model performance and generate adversarial samples. In the best case, the attacker must gather the samples by himself or herself, i.e., sniffing network traffic and extracting the features evaluated by the supervised model.

4. Implementation of VAASIS attack

This section introduces the steps needed to perform the proposed adversarial attack. The first step is identifying the most important features to classify the sample as the desired class. In this step, we also need to determine if those features are categorical or continuous because they

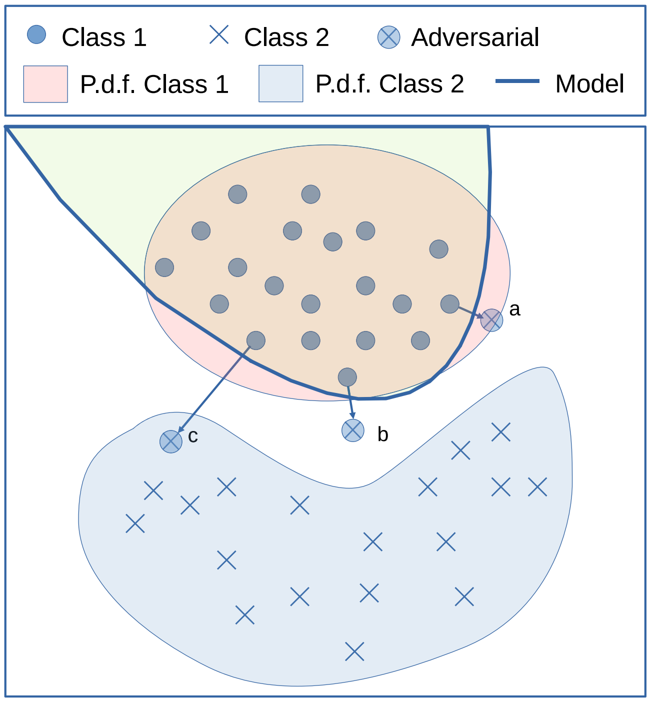

Fig. 1. An example of a binary classification problem where three different cases can happen in industrial scenarios after applying an adversarial attack. A given sample from class 1 can go to different zones when altered adversarially, sometimes even falling into the actual p.d.f. of class 2 if the modification is excessive.

must be treated differently. In the second step, a specific adversarial attack based on the model gradient is selected and deployed to craft the continuous features. The third step is in charge of modifying categorical features based on the similarity with respect to samples belonging to the target class. The fourth step is responsible for evaluating individual adversarial samples to ensure they remain abnormal. Otherwise, the attack tries to recover the abnormal behavior of the samples. The whole process can be observed in Algorithm 1, which will be explained in detail in the following sections. Besides, Fig. 2 shows the steps of VAASIS, including an additional validation step that determines the success of the attack.

4.1. Selection of features to modify

Most adversarial attacks in the literature modify the values of all the features without any control. Fundamentally, this is because these attacks are focused on the field of computer vision, where the input features are pixel values, and a slight modification of them is almost imperceptible to the human eye. However, modifying all the features is not an option in data network environments, particularly in industrial networks. On the one hand, some of the features may be essential for the correct functioning of the system and, therefore, cannot be modified. For example, the IP of a network packet is crucial for the packet to be routed and reach its destination. On the other hand, modifying all the features without following a clear criterion can generate adversarial samples that are noticeably different from the original samples and, therefore, easily detectable by an expert.

We propose to modify specific features to introduce the minimal required error and, therefore, to generate adversarial samples as similar as possible to the original ones from which they were created. To select the proper features to modify, we estimate the importance of each feature in classifying a sample as a particular class. Furthermore, we consider that the importance of the features may vary from sample to sample, not being constant throughout the dataset.

When evaluating the importance of the features, a common approach is to use an ML model that includes some basic interpretability mechanisms, such as the importances returned by a Random Forest (RF) or the weights of a linear regression model. In fact, these models can be used successfully in time-series scenarios [25], such as industrial ones, by adapting the shape of the dataset from 3-dimension to 2-dimension. In particular, a dataset in an industrial environment usually has the shape of (n, ts, m), where n is the number of samples in the dataset, ts is the number of timesteps considered, and m is the number of features in the dataset. However, those techniques only return global importances computed over the entire dataset. This prevents us from selecting specific features for individual samples.

To solve this problem, we propose using SHapley Additive exPlanations (SHAP) [26], a unified framework for interpreting predictions that can be used to measure feature importance. In SHAP, these importances are known as SHAP Values (SV), and they come from the Shapley values of a conditional expectation function of the original model. These SHAP Values are computed using Eq. (1); however, we use an existing library that hides the implementation details. The aim of using SHAP is to determine the importance the model gives to each feature in each sample. This information allows us to select those features that are most likely to cause the sample to change class upon receiving a minimal perturbation. Regarding the Algorithm 1, the function in charge of computing SHAP values is $get\_shap\_values()$.

$$SV(f,x)=\sum_{z'\subseteq x'}\frac{|z'|!(M-|z'|-1)!}{M!}[f_x(z')-f_x(z'\setminus i)] \quad (1)$$

Where f is the original prediction model, x' is the simplified inputs that map to the original inputs, x, through a mapping function $x = h_x(x')$, $z' \in \{0,1\}^M$, M is the number of simplified input features, $|z'|$ is the number of non-zero entries in z' and $z' \subseteq x'$ represents all z' vectors where the non-zero entries are a subset of the non-zero entries in x', and $z' \setminus i$ denote setting $z'_i = 0$.

Typically, to apply SHAP, a so-called background dataset is needed for integrating features. To determine the impact of a feature, that feature is set to missing and the change in the model output is observed. This dataset serves as a kind of training dataset for SHAP technique and it is used to simulate missing values by replacing the feature with the values it takes in this dataset. The SHAP documentation recommends selecting between 100 and 1 000 random background samples.

Once the SV for each feature are obtained, they are ordered descendingly. In this way, we can select the highest values based on the percentile statistic that will be established experimentally. Finally, due to the limitations that adversarial attacks have dealing with categorical features, continuous and categorical features will be modified separately to generate valid adversarial samples in industrial environments. The pseudocode of this process is showed in lines 1 to 3 of Algorithm 1.

4.2. Generation of continuous features

In this step, we select the adversarial attack that modifies the previously selected continuous features. In this case, we opted for a targeted adversarial attack that is powerful and not excessively costly in computing time. In particular, we use PGD to generate continuous features. This attack calculates in each iteration the gradient of the loss function with respect to the input and modulates the gradient sign by a perturbation parameter to finally subtract it from the original sample. The formal definition of this attack is shown in Eq. (2).

$$X'_0 = X; X'_{n+1} = Clip_{X,\varepsilon}(X'_n - \varepsilon_{step} \cdot sign(\nabla_x J(X'_n, y_{true}))) \quad (2)$$

In this case, X is the original sample, and X'_0 is the first iteration where the original samples are considered. Similarly, X'_{n+1} are the successive iterations where the same gradient step as in FGSM is applied. Furthermore, y_{true} represents the target class of the adversarial attack, $J(X'_n, y_{true}))$ is the loss function with respect to the input, and ∇_x

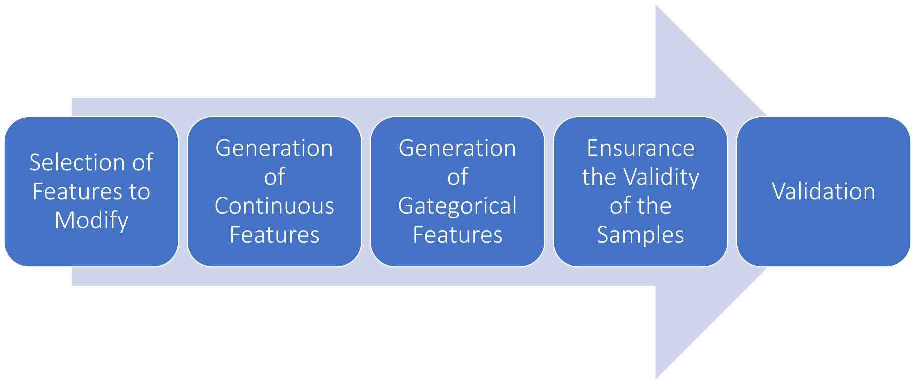
Selection of Features to Modify
Generation of Continuous Features
Generation of Gategorical Features
Ensurance the Validity of the Samples
Validation

Fig. 2. The five main steps of VAASIS attack.

Algorithm 1: Algorithm to generate valid and consistent adversarial samples in industrial scenarios without losing their abnormal behavior

```
Input: $X_{train}$ training dataset, $X_{test}$ testing dataset, $M$ model, $\epsilon_{step}$ maximum disturbance per step, $\epsilon$ global maximum disturbance, $iter$
       number of iterations
Output: X' adversarial dataset
1  $X_B \leftarrow create\_background\_dataset(X_{train})$ /* create background dataset, $X_B$, from $X_{train}$ */
2  $SV \leftarrow get\_shap\_values(X_{test}, X_B)$ /* get SHAP values, $SV$, from $X_{test}$ using $X_B$ */
3  $F_D, F_C \leftarrow select\_most\_important\_features\_for\_class(SV, ABNORMAL\_CLASS)$ /* Select the continuous, $F_C$, and discrete, $F_D$,
       features to be modified for abnormal class */
   /* Execute the projected gradient descent over the continuous features, $F_C$, with $\epsilon_{step}$ as the maximum disturbance per
       iteration, $\epsilon$ as the maximum global disturbance allowed and $iter$ as the number of iterations */
4  $X' \leftarrow projected\_gradient\_descent(X_{test}[F_C], M, \epsilon_{step}, \epsilon, iter)$
   /* Create the discretized dataset, $X_D$, from the normal samples of $X_{train}$ */
5  $X_D[F_C] \leftarrow discretize(X_{train}[Normal][F_C])$
6  $X'_D[F_C] \leftarrow discretize(X'[F_C])$ /* Create the discretized adversarial dataset, $X'_D$, from the $X'$ */
7  for $i = 1..length(X')$ do
8      $c \leftarrow get\_similar\_samples(X'_D[i], X_D)$ /* get the similar sample $c$ for sample $i-th$ in $X_D$ */
9      $X'[i][F_D] \leftarrow c[F_D]$ /* Copy the $F_D$ from $c$ */
10 $rf \leftarrow train\_rf(X_{train})$ /* Train a Random Forest, $rf$, using the $X_{train}$ */
11 for $s$ in $X'$ /* For each sample, $s$, in X' */
12  do
13     if $predict(rf, s) := NORMAL\_CLASS$ /* If $s$ is predicted as normal by the $rf$ */
14      then
15         $selected\_paths \leftarrow []$
16         for $estimator$ in $rf.estimators$ do
17             $paths \leftarrow get\_rules(estimator, ABNORMAL\_CLASS)$ /* Get the rules to check the abnormal class */
18             $best\_path \leftarrow get\_path\_with\_least\_error(sample, paths)$
19             $append(selected\_paths, best\_path)$
20         $n\_agreements = choose\_number\_between(length(rf.estimators)/2 + 1, length(rf.estimators))$
21         for $i = 1..n\_agreements$ do
22             $path \leftarrow get\_path\_with\_least\_error(selected\_paths)$
23             $s \leftarrow modify\_sample(s, path)$
24             $remove(selected\_paths, path)$
```

is the gradient. Unlike FGSM, this iterative version requires an operator $Clip_{X,\epsilon}$ to limit the final result between $[X-\epsilon, X+\epsilon]$, and ϵ_{step} is a scalar that indicates the maximum allowed perturbance in each iteration.

For the correct operation of the attack, we must select the proper values for the parameters. In particular, we can configure three parameters that will determine the performance of the attack. The first one is ϵ_{step}. The smaller this value, the smaller the modification introduced in each iteration. Similarly, ϵ indicates how much the sample can be modified globally. Finally, the number of iterations is also a critical parameter since it indicates how many modifications will be carried out on the sample. In general, when an adversarial attack targeted at AD systems is applied to a set of samples, not all the samples are converted to the normal class, but some of them may still be classified as abnormal by the AD and, therefore, they are not considered adversarial samples. In particular, the higher the values of these parameters are, the greater the number of adversarial samples. However, this also causes the error, i.e., the difference between the original and adversarial samples, to be greater and, therefore, easier to identify by an expert. To solve this, a trade-off must be reached between the value of these parameters, the error between the original and the adversarial sample, and the number of adversarial samples generated.

Finally, once the parameters of the PGD attack are chosen, it must be launched using the dataset samples as reflected in line 4 of Algorithm 1. In particular, we recommend using the test dataset since it has not been previously seen during the training of the AD system.

4.3. Generation of categorical features

In this work, we propose a novel method to generate categorical features. In particular, after altering the selected set of continuous features of a given sample by applying the adversarial attack, we will replace the value of its selected categorical features with the corresponding values in the same categorical features of the most similar sample in the training dataset. To do so, we will apply the nearest neighbor algorithm on the training dataset using only the modified continuous features of the adversarial sample. The categorical features selected by the highest SHAP values of the adversarial sample will be filled with the corresponding values of its nearest neighbor. These categorical features will vary from sample to sample due to how SHAP values are calculated. However, finding the nearest neighbor in a continuous space can be slow. Therefore, we propose discretizing both the original dataset and the generated set of adversarial samples. This approach has two benefits. Firstly, it is faster than working with continuous data. Secondly, it allows similar samples to be grouped, removing the potential noise in industrial measurements. Once the datasets are discretized, we search for samples whose originally continuous features (now discretized) match those of the generated adversarial sample. If no sample is found where all the discretized selected continuous features coincide with the generated adversarial sample, we select the one with the highest number of coincidences. Otherwise, if multiple similar samples are available, one is chosen randomly. This process is reflected in lines 5 through 9 of Algorithm 1.

4.4. Ensuring the adversarial validity of the samples

Once the previous step is complete, a series of valid adversarial samples for use in industrial environments are generated. However, there is no guarantee that during the PGD attack, the generated samples have not actually become real normal samples and, therefore, no longer have any impact on the physical world. In general, applying excessive perturbation during the attack will cause an adversarial sample to become a true normal sample, as shown in Section 3.

In this work, we propose a novel approach to validate that the generated samples misclassified as normal by the AD system still retain their anomalous behavior. Moreover, in case they have been converted into true normal samples, we propose a mechanism to recover their anomalous behavior.

For this purpose, our proposal includes the use of a Random Forest (RF) classifier, which trains multiple Decision Trees (DT) and allows us to determine the decision rules applied during the classification of a sample. Therefore, the first step is to use the training dataset to train an RF that achieves acceptable performance in discriminating labeled normal and anomalous samples.

Once the RF has been trained, the rules of all its DTs are available. Fig. 3 displays the rules for a particular DT. Note, however, that the features employed are not the genuine ones; instead they are a simplified analogous set used for illustrative purposes. In this tree, we can see different elements. On the one hand, we identify three types of nodes: root, intermediate, and leaf. For each non-leaf node, the feature to be evaluated and a threshold is shown. This is known as the condition, and a certain path will be taken depending on whether it is true or false. For example, on the root node, feature *Water_Level_1* is compared to the threshold -0.101. If this condition is true for the particular sample being evaluated, the left path will be taken; otherwise, the right path will be taken. Furthermore, each node has an additional set of attributes: *gini* indicates the Gini index which is a measure of the quality of the data partition; *samples* indicates the number of samples evaluated in that node (3190 in the root node); *value* is an array of two values containing the number of samples that meet and do not meet the condition (for the root node 1 468 samples meet the condition and 1 722 samples do not meet the condition); finally, the *class* attribute only makes sense in the leaf nodes, where it tells us whether samples ending in that node are normal or abnormal. Node color indicates the class to which most of the samples at each node belong, namely, orange for the normal class and blue for the abnormal class. Color intensity represents the degree of confidence in the prediction.

In a normal prediction flow, a specific sample begins to be evaluated at the root node. It goes down through the intermediate nodes depending on the conditions until it reaches a leaf node. However, we propose traversing the tree bottom-up to extract the rules for a specific class. Looking at Fig. 3, we can see three paths for the anomalous class, one for each leaf labeled as abnormal. These paths are detailed in Table 2.

When we validate individual adversarial samples to check if they retain their abnormal behavior, the first phase is to evaluate the sample using the RF. If the RF classifies the sample as abnormal, it is presumed that the sample retains its abnormal behavior even though the AD system classifies it as normal. On the contrary, if the RF classifies the sample as normal, we consider that the adversarial sample was converted into a true normal sample during the adversarial attack. Therefore, we proceed to extract all the paths starting from an abnormal leaf node of all the DTs, and some of them will be used to modify the sample to recover its anomalous behavior.

To modify each sample that the RF predicts as normal, we calculate the correction required so that the conditions for each path leading to an anomalous leaf node of each of the DTs are met. This correction can be considered as an error and is computed as shown in Eq. (3), where $threshold_f$ is the threshold used by the DT in a given node for the feature f, and s_f is the value of the feature f in the sample s. The path used to modify the sample and recover its anomalous behavior is obtained using the following criteria. First, the path must not modify features previously updated. Second, the path with the least error that meets the first criterion will be selected. In particular, the modification is made following the rules extracted from the DT and modifying, in the way specified in the rules, those features that make the DT classify the sample as normal.

$$error = (threshold_f - s_f)^2 \tag{3}$$

However, since RF is an ensemble and its result is based on a voting process based on all DTs, it is necessary to repeat this process for, at least, half of the DTs trained. Furthermore, we must remember not to choose paths that modify features that have already been previously modified since this makes previous rules not be fulfilled, and the anomalous behavior can be lost.

This process is illustrated in the lines 10–24 of the Algorithm 1.

4.5. Validation

This step allows the attacker to validate the adversarial samples generated following our proposal. This step is important since it will permit us to evaluate how effective the proposed attack is with respect to other attacks present in the literature.

Mainly, the validation focuses on two key aspects of adversarial attacks. On the one hand, it quantifies the number of adversarial samples that the attack has managed to generate, which helps us understand how easily adversarial samples are generated. On the other hand, it evaluates the mean error of these samples in relation to the original samples from which they were generated. This error determines which adversarial attacks generate samples that closely resemble the originals, making them more challenging for experts to identify.

5. Experiments

This section details the steps to deploy our attack in a real industrial scenario of water distribution. First, we present the scenario and the dataset used. Then, we describe the requirements an attacker needs to successfully perform the attack. Finally, we show how we applied the steps explained in Section 4. This process with a specific real portion of a sample from the WADI dataset is shown in Fig. 4, and it will

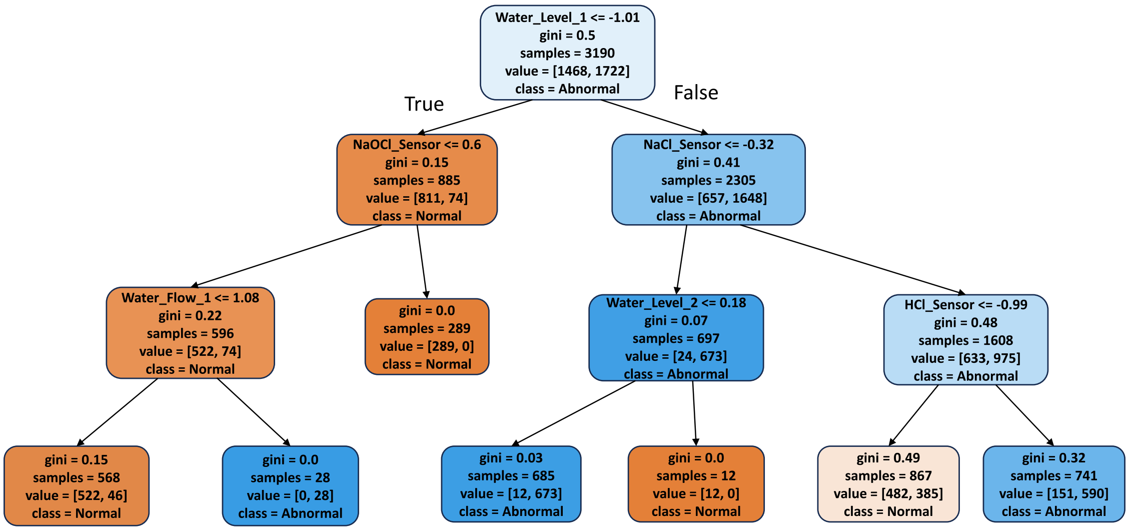

Fig. 3. Illustration of a particular DT in an RF (the features shown are not the actual ones, but a simpler version of them). The tree has three types of nodes: root, intermediate, and leaf. Each non-leaf node compares whether a given feature is less than or equal to a threshold. Depending on whether the condition is true or false, the left or right path is taken. In addition, each node has the Gini index of the partition, the number of samples evaluated in the node, and the number of samples in each class. Finally, node color indicates the class to which the majority of the samples at each node belong. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Paths from the abnormal leaf nodes to the root node in the simplified DT depicted in Fig. 3.

Path 1		Path 2		Path 3	
Feature	Condition	Feature	Condition	Feature	Condition
Water_Flow_1	1.08	Water_Level_2	0.18	HCl_Sensor	−0.99
NaOCl_Sensor	0.6	NaCl_Sensor	−0.32	NaCl_Sensor	−0.32
Water_Level_1	−1.01	Water_Level_1	−1.01	Water_Level_1	−1.01

be referenced in the following sections. For the reader's convenience, this section is divided into the same subsections as Section 4 plus two additional subsections: the dataset and requisites subsections. The former subsection describes the dataset used during the experiments, whereas the latter subsection establishes the set of elements needed by a potential attacker. They are supposed to be already present in the industrial scenario and have to be configured by the administrators.

5.1. Dataset

The Water Distribution (WADI) testbed was launched in 2016, comprising two elevated reservoir tanks, six consumer tanks, two raw water tanks, and a returned tank. Besides, the WADI testbed also incorporated chemical dosing systems, booster pumps, and valves, instrumentation, and analyzers. WADI was designed to account for the likelihood of low demand happening during the weekends and allow users to modify the input flow to simulate the water consumption of a realistic scenario. In addition, WADI is split into three processes, as shown in Fig. 5. The first process (P1), called Primary Grid, contains two raw water tanks and a level sensor (1-LIT-001) that monitors the water level in the tanks. The water intake in this process can come from three different sources: a water treatment plant, a public utility board inlet, and the return water grid in WADI. This process also assures the quality of the water by employing a chemical dosing system. Finally, P1 is also equipped with sensors to monitor the water quality entering and exiting the process. The second process (P2) is called the Secondary Grid, and it is split into two sub-processes: P2 A (Elevated Reservoir) and P2B (Consumer tanks). The P2 A comprises two elevated reservoir tanks, while the P2B consists of six consumer tanks. The water tanks from the Primary Grid supply water to the reservoir tanks, delivering the water to the consumer tanks. When the consumer tanks meet their demands, the water is drained to the Return Water process (P3).

To obtain the data from each process, WADI uses Programmable Logic Controllers (PLC) that control and monitor local sensors and actuators such as pumps and valves. The communication network consists of three layers. Layer 0 (L0) communicates the physical sensors and actuators with their respective PLCs. The communication is carried out by means of the RS485 Modbus protocol. Layer 1 (L1) constitutes the control plant network and serves to communicate all PLCs with a central node. The communication at this level is performed by Ethernet switches using NIP/SIP based on TCP and High-Speed Packet Access (HSPA) cellular gateways using General Packet Radio Service (GPRS) modems. Finally, layer 2 (L2) communicates the touch panel Human Machine Interface (HMI) and the plant control network. The list of sensors and actuators in the WADI testbed, along with the type of data (continuous or discrete) they generate are summarized in the Table 3.

In this industrial setting, 15 different attacks were deployed and executed over the course of 2 days. As a result, the WADI dataset used in this study consists of two files. The first file contains sensor and actuator values during 14 days of normal operation, whereas the second file contains the system's behavior during the various attacks deployed over 2 days. The attacks launched against the testbed are detailed in Table 4. These are genuine attacks feasible within an actual industrial context, thus serving as a means to validate our proposal in a real-world scenario. The dataset features were collected from the sensors and actuators, and, in most cases, they share the name with the sensor/actuator from where they were gathered. The total number of features in the dataset is 123.

5.2. Requisites

This step is carried out by the administrator of the AD system and is intended for configuring required elements before the attack can be deployed. In particular, a slight preprocessing of the WADI dataset was carried out, as well as the training of the supervised model that offers

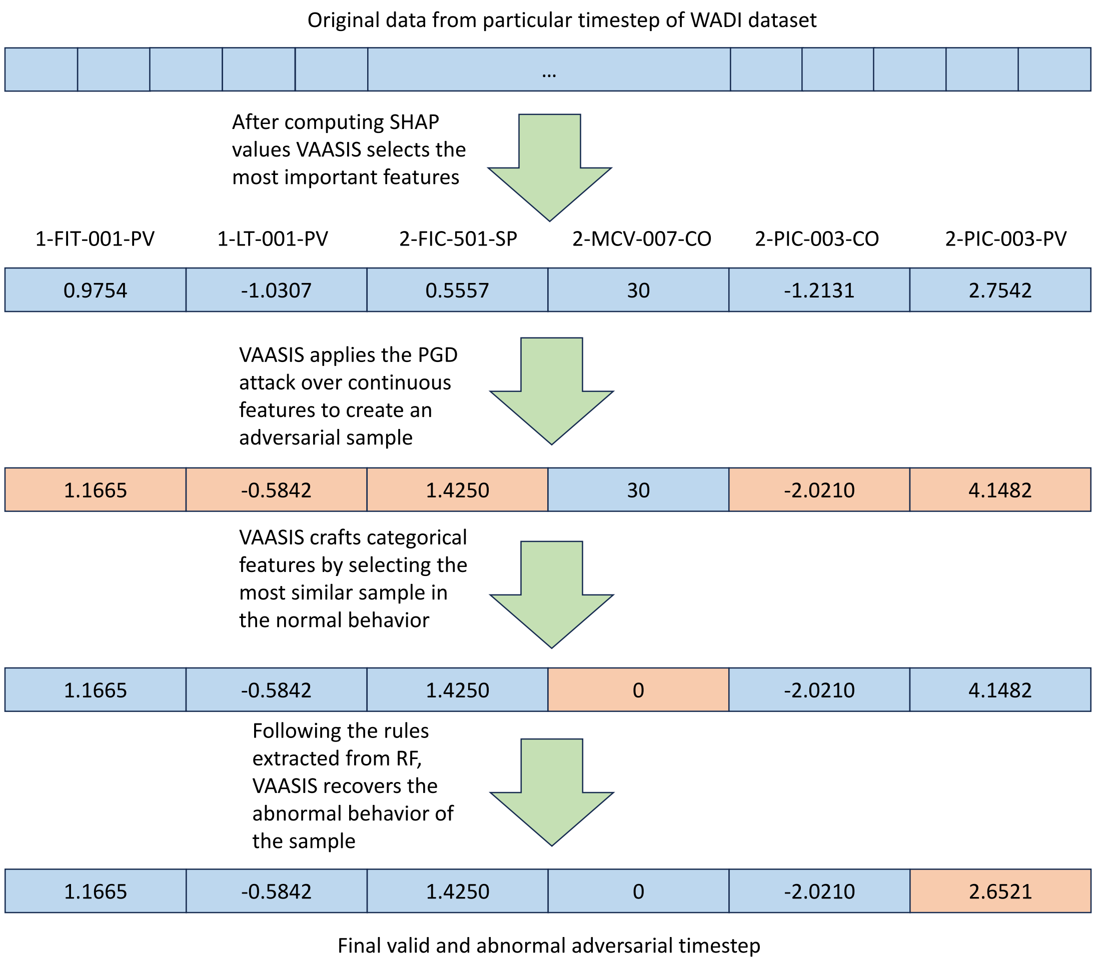
Original data from particular timestep of WADI dataset
...
After computing SHAP values VAASIS selects the most important features
1-FIT-001-PV
1-LT-001-PV
2-FIC-501-SP
2-MCV-007-CO
2-PIC-003-CO
2-PIC-003-PV
0.9754
-1.0307
0.5557
30
-1.2131
2.7542
VAASIS applies the PGD attack over continuous features to create an adversarial sample
1.1665
-0.5842
1.4250
30
-2.0210
4.1482
VAASIS crafts categorical features by selecting the most similar sample in the normal behavior
1.1665
-0.5842
1.4250
0
-2.0210
4.1482
Following the rules extracted from RF, VAASIS recovers the abnormal behavior of the sample
1.1665
-0.5842
1.4250
0
-2.0210
2.6521
Final valid and abnormal adversarial timestep

Fig. 4. Example of generation of a specific adversarial timestep. In the first phase, SHAP is employed to determine the most representative features among the 96 features of the timestep. In this particular case, six features are selected. In the next step, PGD is applied to modify the five continuous features. Subsequently, the most similar sample from the normal behavior of WADI is chosen to modify the only categorical feature (2-MCV-007-CO). Finally, the rules extracted from the RF trees are used to verify that the sample is considered normal and that the feature 2-PIC-003-PV should be modified to restore its anomalous behavior. For the sake of clarity in this example, we have avoided displaying the process of modifying an entire sample and have only illustrated the modification of a portion of it (timestep). However, generating a complete adversarial sample would involve simply extending this process to the remaining timesteps that comprise the sample. The orange color in the image is used to highlight those values that have been modified in each step. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

sufficiently high performance to be deployed in real environments. Note that we are not interested in obtaining the best AD performance; therefore, a grid-search strategy was not performed to find the optimal values for the hyper-parameters. We only need the AD model as a prerequisite to perform the next steps of the proposed attack.

First, a series of actions were taken to make it easier to work with the dataset. For example, the last two rows of the dataset were removed since they contained null values. Additionally, the columns *2-LS-001-AL*, *2-LS-002-AL*, *2-P-001-STATUS*, and *2-P-002-STATUS* were removed since all its values were Not A Number (NAN). In addition, columns *Row*, *Date*, and *Time* and all the columns whose values were constant throughout the entire dataset (Table 5) were removed, resulting in samples with 96 features.

Next, the dataset was partitioned into training and test, and the continuous features were normalized using standard normalization. To perform the partition, we first created a sliding window of size 5. This converted the shape of the original dataset from $(samples, features)$ into a shape of $(samples, 5, features)$. The next step was to create a balanced dataset. In particular, we took the total number of attack samples in the dataset, 9 977, and selected the same amount of normal samples using random sampling throughout the entire dataset. Finally, and with the goal of capturing the different patterns of the normal behavior in the whole dataset, a procedure based on folds was established to create the training and test dataset. In particular, the normal and attack sample sets were divided into 20 folds, and in each of these folds, the first 80% of the samples were chosen for the training dataset and the remaining 20% of the samples for the test dataset.

Finally, we used Tensorflow [27] and Keras [28] libraries to train a recurrent neural network with 2 LSTM layers and 3 dense layers by means of the Adam optimizer. ReLU activation function was used in each hidden dense layer together with dropout regularization. The activation function of the output layer was softmax. The full network architecture is detailed in Table 6, and the performance achieved by the AD system is shown in Table 7.

5.3. Selection of features to modify

In this step, we used the SHAP library [26] to extract the importance of each feature in samples. First, we created a background dataset by selecting 100 normal and 100 abnormal samples from the training dataset. Second, we extracted the importance of each feature in test samples. This results in an array of shape $(2, 2\,000, 5, 96)$ being 2 the different classes in the WADI dataset (normal and abnormal), 2 000 the total number of attack samples in the test dataset, 5 the timesteps used to create the time-series sequence, and 96 the number of features. Finally, from the test dataset, we selected the features whose SHAP values were highest for the abnormal class. This told us the most important features of each sample that contribute to being abnormal.

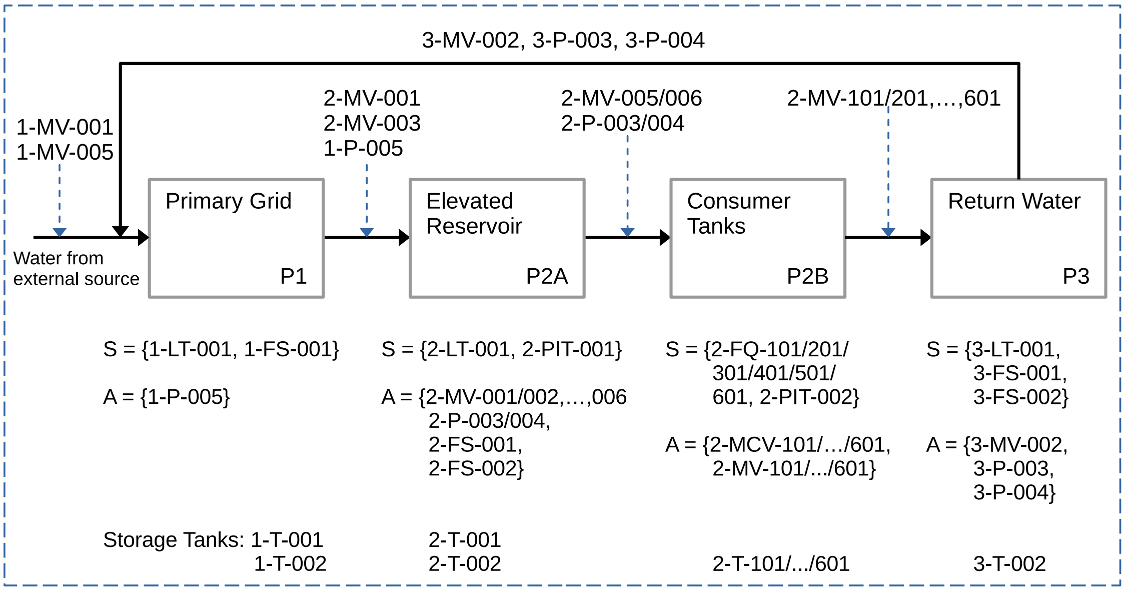
3-MV-002, 3-P-003, 3-P-004
1-MV-001
1-MV-005
2-MV-001
2-MV-003
1-P-005
2-MV-005/006
2-P-003/004
2-MV-101/201,…,601
Primary Grid
P1
Elevated Reservoir
P2A
Consumer Tanks
P2B
Return Water
P3
Water from external source
S = {1-LT-001, 1-FS-001}
A = {1-P-005}
S = {2-LT-001, 2-PIT-001}
A = {2-MV-001/002,…,006 2-P-003/004, 2-FS-001, 2-FS-002}
S = {2-FQ-101/201/ 301/401/501/ 601, 2-PIT-002}
A = {2-MCV-101/…/601, 2-MV-101/.../601}
S = {3-LT-001, 3-FS-001, 3-FS-002}
A = {3-MV-002, 3-P-003, 3-P-004}
Storage Tanks: 1-T-001 1-T-002
2-T-001 2-T-002
2-T-101/.../601
3-T-002

Fig. 5. The stages of WADI[9]. The solid arrows illustrate the flow of the water. S and A stand for Sensor and Actuator, respectively. 1-MV-001 represents the motorized valve 1 for stage 1, while 2-MV-001 represents the motorized valve 1 for stage 2. MV: Motorized Valve, LT: Level sensor of a tank, P: pump, FS: Flow meter, and MCV: Motorized Consumer Valve.

Table 3
Sensors and actuators from WADI testbed.

Type of device	Feature name	Datatype
Pressure Meter	2-PIT-001-PV, 2-PIT-002-PV, 2-PIT-003-PV	Continuous
Modulating Consumer Valve	2-MCV-101-CO, 2-MCV-201-CO, 2-MCV-301-CO, 2-MCV-401-CO, 2-MCV-501-CO, 2-MCV-601-CO, 2-MCV-007-CO	Continuous
Flow Totalizer	2-FQ-101-PV, 2-FQ-201-PV, 2-FQ-301-PV, 2-FQ-401-PV, 2-FQ-501-PV, 2-FQ-601-PV	Continuous
Level Transmitter	1-LT-001-PV, 2-LT-001-PV, 2-LT-002-PV, 3-LT-001-PV	Continuous
Analyzer Indicator Transmitter	1-AIT-001-PV, 1-AIT-002-PV, 1-AIT-003-PV, 1-AIT-004-PV, 1-AIT-005-PV, 2A-AIT-001-PV, 2A-AIT-002-PV, 2A-AIT-003-PV, 2A-AIT-004-PV, 2B-AIT-001-PV, 2B-AIT-002-PV, 2B-AIT-003-PV, 2B-AIT-004-PV, 3-AIT-001-PV, 3-AIT-002-PV, 3-AIT-003-PV, 3-AIT-004-PV, 3-AIT-005-PV	Continuous
Flow Indicator Transmitter	1-FIT-001-PV, 2-FIT-001-PV, 2-FIT-002-PV, 2-FIT-003-PV, 3-FIT-001-PV	Continuous
Differential Pressure Transmitter	2-DPIT-001-PV	Continuous
Flow Indicator Controller	2-FIC-101-CO, 2-FIC-101-PV, 2-FIC-101-SP, 2-FIC-201-CO, 2-FIC-201-PV, 2-FIC-201-SP, 2-FIC-301-CO, 2-FIC-301-PV, 2-FIC-301-SP, 2-FIC-401-CO, 2-FIC-401-PV, 2-FIC-401-SP, 2-FIC-501-CO, 2-FIC-501-PV, 2-FIC-501-SP, 2-FIC-601-CO, 2-FIC-601-PV, 2-FIC-601-SP	Continuous
Pressure Indicator Controller	2-PIC-003-CO, 2-PIC-003-PV, 2-PIC-003-SP	Continuous
Pump Speed	2-P-003-SPEED, 2-P-004-SPEED	Continuous
Pump	1-P-001-STATUS, 1-P-002-STATUS, 1-P-003-STATUS, 1-P-004-STATUS, 1-P-005-STATUS, 1-P-006-STATUS, 2-P-001-STATUS, 2-P-002-STATUS, 2-P-003-STATUS, 2-P-004-STATUS, 3-P-001-STATUS, 3-P-002-STATUS, 3-P-003-STATUS, 3-P-004-STATUS	Discrete
Motorized Valve	1-MV-001-STATUS, 1-MV-002-STATUS, 1-MV-003-STATUS, 1-MV-004-STATUS, 2-MV-001-STATUS, 2-MV-002-STATUS, 2-MV-003-STATUS, 2-MV-004-STATUS, 2-MV-005-STATUS, 2-MV-006-STATUS, 2-MV-009-STATUS, 2-MV-101-STATUS, 2-MV-201-STATUS, 2-MV-301-STATUS, 2-MV-401-STATUS, 2-MV-501-STATUS, 2-MV-601-STATUS, 3-MV-001-STATUS, 3-MV-002-STATUS, 3-MV-003-STATUS	Discrete
Selenoid Valve	2-SV-101-STATUS, 2-SV-201-STATUS, 2-SV-301-STATUS, 2-SV-401-STATUS, 2-SV-501-STATUS, 2-SV-601-STATUS	Discrete

Table 4
Attacks deployed in the testbed.

Attack	Duration (min)	Description
1	25.16	Motorized valve 1-MV-001 is maliciously turned on
2	9.50	Flow Indicator Transmitter 1-FIT-001 is turned off
3-4	29.00	Drain the elevated reservoir tank 2 (2-T-002)
5	14.10	Turned off valves to consumers (2-MCV-101, 2-MCV-201, 2-MCV-301, 2-MCV-401, 2-MCV-501, 2-MCV-601)
6	11.10	Turn on maliciously 2-MCV-101, 2-MCV-201
7	11.38	Supply contaminated water to the elevated reservoir tank
8	9.40	Maliciously open 2-MCV-007
9	1.27	Maliciously open 1-P-006
10	14.26	Damage 1-MV-001
11	11.29	Maliciously open 2-MCV-007
12	6.00	Maliciously open 2-MCV-007
13	3.22	Reducing booster set point pressure
14	9.36	Stop chemical dosing to the raw water
15	11.30	Flood the elevated reservoir tank 2 (2-T-002)

Table 5
Features whose values remained constants in the whole dataset.

1-LS-001-AL	2-P-004-STATUS	3-LS-001-AL
1-LS-002-AL	2-SV-101-STATUS	3-MV-001-STATUS
1-P-002-STATUS	2-SV-201-STATUS	3-MV-002-STATUS
1-P-004-STATUS	2-SV-301-STATUS	3-MV-003-STATUS
2-MV-001-STATUS	2-SV-401-STATUS	3-P-001-STATUS
2-MV-002-STATUS	2-SV-501-STATUS	3-P-002-STATUS
2-MV-004-STATUS	2-SV-601-STATUS	3-P-003-STATUS
2-MV-005-STATUS	3-AIT-001-PV	3-P-004-STATUS
2-MV-009-STATUS	3-AIT-002-PV	PLANT-START-STOP-LOG

Table 6
Architecture of LSTM model used to detect AD.

Layers	Hyper-parameters
LSTM Layer	Nodes: 16
LSTM Layer	Nodes: 8
Dense Layer	Nodes: 16
Activation	ReLU
Dropout	Rate: 0.7
Dense Layer	Nodes: 8
Activation	ReLU
Dropout	Rate: 0.7
Dense Layer	Nodes: 2
Activation	Softmax
Loss	Categorical crossentropy
Epochs	100
Batch size	1000
Optimizer	Adam

Table 7
AD system performance.

Precision	Recall	F1-score
0.974	0.972	0.973

Finally, we selected a particular number of features to be modified in the PGD attack. In this case, we selected all features whose SHAP values were above the 90th percentile. This value can be a hyper-parameter and can be tuned for better performance. However, in this study, we have verified that the selected value offers good performance. At this point, we discriminated which of these features were continuous and would be modified by PGD and which were discrete and would be modified by copying the categorical value of the most similar sample in the normal training dataset. In the specific sample depicted in Fig. 4, we show which of the 96 original features of the fifth timestep obtained the highest SHAP values: 1-FIT-001-PV, 1-LT-001-PV, 2-FIC-501-SP, 2-MCV-007-CO, 2-PIC-003-CO, and 2-PIC-003-PV.

5.4. *Generation of continuous features*

In this step, the PGD attack provided by the Adversarial Robustness Toolbox (ART) [29] was launched. First, we selected all attack samples in the test dataset. Then, we selected the attack parameters. In particular, we established the maximum perturbation allowed, ϵ, to 4, while the maximum perturbation per iteration allowed, ϵ_{step}, was set to 0.05. Finally, the number of iterations was fixed to 50. As can be observed in Section 4, these are hyper-parameters that can be tuned to get better adversarial samples. However, we have verified that the selected values achieved a good trade-off between the number of adversarial samples generated and the error produced. The goal is to convert all the attack samples in the test dataset into normal samples. This causes samples to be misclassified by the AD system and reach the industrial devices, impacting the physical world.

After launching the attack, we converted the original anomalous samples of the test dataset into adversarial samples. The majority of them (83.62%) were classified as normal by the AD system, and the remaining samples (16.38%) were still classified as anomalous by the AD.

Fig. 4 illustrates how PGD was applied to modify five of the continuous features present in the fifth timestep of a sample. In particular, the modified features were: 1-FIT-001-PV, 1-LT-001-PV, 2-FIC-501-SP, 2-PIC-003-CO, and 2-PIC-003-PV.

5.5. *Generation of categorical features*

In this step, we modified the categorical features of the generated adversarial samples to convert them into valid features in an industrial environment, where the presence of continuous values in these features would make the sample invalid. To facilitate the nearest-neighbor search restricted to continuous features, the samples of the training dataset and the previously generated adversarial samples were discretized. This discretization was performed using the KBinsDiscretizer class of the scikit-learn [30] library, configured with 10 bins and a uniform discretization strategy. The number of bins must be adapted to the particularities of the dataset.

Next, for each discretized adversarial sample, and using only the continuous modified features, the sample with the highest number of matches was located in the discretized normal training dataset. A match was considered to occur when the value of a continuous feature that was discretized from the set of adversarial samples matches the value of a discretized feature from the normal training dataset. Finally, the values of the categorical features from the matched sample are copied into the corresponding features of the adversarial sample.

Fig. 4 shows that the categorical feature (2-MCV-007-CO) was not modified in the initial steps. Therefore, in this step, VAASIS compared this specific timestep with the timesteps in the normal behavior and selected the most similar sample. Then, the value of 2-MCV-007-CO was taken from the most similar sample, 0, in this example.

Table 8
RF performance.

Precision	Recall	F1-score
0.971	0.968	0.969

5.6. *Ensurance the validity of the samples*

In this step, an RF was used to validate the previously generated adversarial samples and, thus, ensure that they retain their anomalous behavior. To do this, we trained an RF using the training dataset created in Section 5.2. The RF was trained using the scikit-learn library, the number of estimators was set to 10, and the maximum depth of the trees to 10. The RF performance is shown in Table 8.

Next, we checked whether the RF predicted each previously generated adversarial sample as normal. If so, it means that the sample lost its anomalous behavior, and it actually became a true normal sample. On the contrary, if the RF predicts the sample as anomalous, it means that the sample maintained its anomalous behavior. Once we knew which samples were converted to true normal ones, we employed the paths of the DTs trained in the RF to determine the necessary changes in each sample to return them to their anomalous behavior.

In the example shown in Fig. 4, the resulting sample was classified as normal, thus losing its effectiveness. Therefore, VAASIS employed the rules extracted from the trees of the RF to recover the abnormal behavior of the sample. In this specific example, it was necessary to modify the value of the 2-PIC-003-PV feature, resulting in a change from 4.1482 to 2.6521. With this change, the sample was deemed abnormal according to the RF model. Consequently, we consider that VAASIS generated a valid abnormal adversarial sample.

5.7. *Validation*

Finally, in this step, we validate the results and compare them with those obtained using raw PGD and CW methods, which are the state of the art in adversarial attacks. To carry out this validation, we consider the number of adversarial samples generated and the average error of these samples. To perform the comparison, we executed our attack as explained previously and, in addition, executed PGD with different configurations of the maximum permissible disturbance, ϵ. In particular, we tested with 0.1, 0.3, 0.5, 0.7, and 1.0. In all the simulations, the number of iterations was set at 50, and the maximum disturbance allowed in each iteration, ϵ_{step}, was set at 0.05. Furthermore, we considered the CW attack, as it is the most powerful adversarial attack. However, due to the high computational cost of this attack, we limited the number of iterations to 10. A critical parameter of this attack is the confidence level. Higher confidence produces samples that are farther away from the original sample but are classified with more confidence as the target class. For this parameter, we tested values of 0 0.1, 0.2, and 0.3, but the results were almost similar. Therefore, in this section, we only consider the execution of CW using a confidence level of 0. Finally, we tested our proposal and the methods mentioned above using several l_p norms, specifically the l_1, l_2, and l_∞ norms. Table 9 shows the results for all the methods configured as described above.

As evident from the results, our proposal achieved the best trade-off between the number of adversarial samples generated and the resulting error. These adversarial samples are particularly challenging for expert detection. In particular, for the l_1 norm, our proposal attained an error of 0.006, slightly surpassing PGD(0.1), which achieved an error 0.005. However, the latter configuration only generated half the number of the adversarial samples (2.8%) compared to our proposal (5.4%).

On the other hand, PGD(1.0) managed to generate 3.1% of adversarial samples with an error of 0.054. However, our proposal significantly outperformed it, which achieved an error nearly ten times lower. Shifting to the l_2 configuration, our proposal demonstrated the lowest error at 0.028, which was nearly four times less than the second-best error achieved by PGD(0.1).

In terms of the number of adversarial samples generated, our proposal excelled by producing 62.25% more than PGD. The CW method, while generating 86.25% of adversarial samples, incurred a significantly higher error, 30 times that of our proposal.

Lastly, in the l_∞ configuration, our proposal reached the second-best error at 0.205, with the lowest error belonging to CW at 0.027. Notwithstanding this, our proposal surpassed CW in terms of the quantity of generated adversarial samples, producing 85.50% compared to CW's 23.34%.

In summary, our proposal achieved the best balance between generating adversarial samples and the resulting error, rendering the samples challenging for expert identification. Specifically, our proposal can generate a substantial number of adversarial samples with minimal error. Finally, in industrial environments, our proposal can generate valid and consistent samples as it addresses the creation of categorical features and guarantees the anomalous behavior of the generated samples, advantages that are absent in the original PGD and CW methods.

6. Conclusion and future work

The AD paradigm, along with ML and DL techniques, has achieved significant performance in industrial scenarios. However, these techniques are vulnerable to adversarial attacks capable of modifying samples in a way that causes the AD systems to misclassify them. These attacks mainly focus on computer vision and present important limitations when applied in industrial environments. In this work, we propose a novel targeted adversarial attack called VAASI, capable of generating valid adversarial samples in industrial settings.

First, we propose selecting the most important features for each sample through SHAP. The selected features are the ones that will be modified in further steps. Then, the PGD method generates trivial adversarial samples by modifying the continuous features. Afterward, the categorical features are modified using the samples from the normal dataset that have the highest match with the particular adversarial sample. Finally, a novel approach based on the DTs trained by an RF is used to ensure the abnormal behavior of the samples. If adversarial samples are considered normal, the DT rules are used to modify the sample and recover its abnormal behavior.

This novel attack was validated using a WADI dataset gathered from a water distribution industrial plant. We compared our proposal with PGD and CW methods with different configurations for the maximal allowed disturbance and confidence parameters, respectively. The selection of these methods is based on the fact that they are the most powerful methods available in the literature. We concluded that our proposal achieved the best balance between the number of adversarial samples generated and the error produced to generate them. To be specific, under a l_1 norm, our proposal had an error only 0.001 higher than the methods with the least error. However, our proposal generated more adversarial samples than any of the tested methods. Under a l_2 norm, we obtained the least error (0.028) while the percentage of adversarial samples produced (62.25%) was higher than any of the PGD configurations tested. Finally, under a l_∞, our proposal generated 85.80% of samples with an error of 0.205. In the case of CW, it generated 23.34% of adversarial samples with an error of 0.027, while PGD produced 92% of adversarial samples with an error of 17.574.

As future work, we plan to study defense mechanisms against the adversarial attacks developed in this work. In particular, we intend to test the effectiveness of the adversarial training to enhance the robustness of the model against VAASIS. Another line to explore in the future is to migrate VAASIS to a black-box scenario where the attacker has no knowledge about the model and its features.

Table 9
Performance comparison between state-of-the-art adversarial attacks (PGD and CW with different parameters) and our proposal.

		PGD(0.1)	PGD(0.3)	PGD(0.5)	PGD(0.7)	PGD(1.0)	CW	Ours
L_1	Samples generated	2.8%	2.8%	2.8%	2.9%	3.1%	–	5.4%
	Average error	0.005	0.016	0.027	0.038	0.054	–	0.006
L_2	Samples generated	3.35%	9.25%	22%	33.4%	49.5%	86.25%	62.25%
	Average error	0.087	0.248	0.389	0.519	0.721	0.823	0.028
L_∞	Samples generated	66.80%	86.10%	87.30%	89.75%	92%	23.34%	85.80%
	Average error	1.844	5.435	9.016	12.540	17.574	0.027	0.205

CRediT authorship contribution statement

Angel Luis Perales Gómez: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **Lorenzo Fernández Maimó:** Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft. **Alberto Huertas Celdrán:** Data curation, Investigation, Methodology, Writing – review & editing. **Félix J. García Clemente:** Funding acquisition, Investigation, Project administration, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work has been funded under Grant TED2021-129300B-I00, by MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, UE, Grant PID2021-122466OB-I00, by MCIN/AEI/10.13039/501100011033/FEDER, UE, by the strategic project DEFENDER from the Spanish National Institute of Cybersecurity (INCIBE), by the Recovery, Transformation and Resilience Plan, Next Generation EU, by the Swiss Federal Office for Defense Procurement (armasuisse) with the CyberForce (CYD-C-2020003), and by the University of Zurich (UZH).

References

[1] Xu LD, Xu EL, Li L. Industry 4.0: state of the art and future trends. Int J Prod Res 2018;56:2941–62.
[2] Nguyen H, Tran K, Zeng X, Koehl L, Castagliola P, Bruniaux P. Industrial internet of things, big data, and artificial intelligence in the smart factory: A survey and perspective. In: ISSAT international conference on data science in business, finance and industry. 2019, p. 72–6.
[3] Nahavandi S. Industry 5.0–a human-centric solution. Sustainability 2019;11(4371).
[4] Al-amri R, Murugesan RK, Man M, Abdulateef AF, Al-Sharafi MA, Alkahtani AA. A review of machine learning and deep learning techniques for anomaly detection in IoT data. Appl Sci 2021;11(5320).
[5] Perales Gomez AL, Fernández Maimó L, Huertas Celdran A, Garcia Clemente FJ, Gil Pérez M, Martínez Pérez G. Safeman: A unified framework to manage cybersecurity and safety in manufacturing industry. Softw - Pract Exp 2021;51:607–27.
[6] Gómez ÁLP, Maimó LF, Celdrán AH, Clemente FJG, Sarmiento CC, Masa CJDC, et al. On the generation of anomaly detection datasets in industrial control systems. IEEE Access 2019;7:177460–73.
[7] Ren K, Zheng T, Qin Z, Liu X. Adversarial attacks and defenses in deep learning. Engineering 2020;6:346–60.
[8] Madry A, Makelov A, Schmidt L, Tsipras D, Vladu A. Towards deep learning models resistant to adversarial attacks. 2017, arXiv preprint arXiv:1706.06083.
[9] Ahmed CM, Palleti VR, Mathur AP. Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In: Proceedings of the 3rd international workshop on cyber-physical systems for smart water networks. 2017, p. 25–8.
[10] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. 2014, arXiv preprint arXiv:1412.6572.
[11] Kurakin A, Goodfellow I, Bengio S. Adversarial machine learning at scale. 2016, arXiv preprint arXiv:1611.01236.
[12] Moosavi-Dezfooli SM, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 2574–82.
[13] Papernot N, McDaniel P, Jha S, Fredrikson M, Celik ZB, Swami A. The limitations of deep learning in adversarial settings. In: 2016 IEEE European symposium on security and privacy. IEEE; 2016, p. 372–87.
[14] Carlini N, Wagner D. Towards evaluating the robustness of neural networks. In: 2017 IEEE symposium on security and privacy. IEEE; 2017, p. 39–57.
[15] Gómez ÁLP, Maimó LF, Celdrán AH, Clemente FJG, Cleary F. Crafting adversarial samples for anomaly detectors in industrial control systems. Procedia Comput Sci 2021;184:573–80.
[16] Anthi E, Williams L, Javed A, Burnap P. Hardening machine learning denial of service (dos) defences against adversarial attacks in iot smart home networks. Comput Secur 2021a;108:102352.
[17] Ibitoye O, Shafiq O, Matrawy A. Analyzing adversarial attacks against deep learning for intrusion detection in iot networks. In: 2019 IEEE global communications conference. IEEE; 2019, p. 1–6.
[18] Singh A, Sikdar B. Adversarial attack and defence strategies for deep-learning-based iot device classification techniques. IEEE Internet Things J 2021;9:2602–13.
[19] Anthi E, Williams L, Rhode M, Burnap P, Wedgbury A. Adversarial attacks on machine learning cybersecurity defences in industrial control systems. J Inf Secur Appl 2021b;58:102717.
[20] Demontis A, Melis M, Pintor M, Jagielski M, Biggio B, Oprea A, et al. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In: 28th USENIX security symposium. 2019, p. 321–38.
[21] Zizzo G, Hankin C, Maffeis S, Jones K. Adversarial attacks on time-series intrusion detection for industrial control systems. In: 2020 IEEE 19th international conference on trust, security and privacy in computing and communications. IEEE; 2020, p. 899–910.
[22] Basak A, Rathore P, Nistala SH, Srinivas S, Runkana V. Universal adversarial attack on deep learning based prognostics. In: 2021 20th IEEE international conference on machine learning and applications. IEEE; 2021, p. 23–9.
[23] Wu T, Wang X, Qiao S, Xian X, Liu Y, Zhang L. Small perturbations are enough: Adversarial attacks on time series prediction. Inform Sci 2022;587:794–812.
[24] Perales Gómez ÁL, Fernández Maimó L, Huertas Celdrán A, García Clemente FJ. Madics: A methodology for anomaly detection in industrial control systems. Symmetry 2020;12(1583).
[25] Kane MJ, Price N, Scotch M, Rabinowitz P. Comparison of arima and random forest time series models for prediction of avian influenza H5N1 outbreaks. BMC Bioinform 2014;15:1–9.
[26] Lundberg SM, Lee SI. A unified approach to interpreting model predictions. In: Advances in neural information processing systems. 30, 2017, p. 4765–74.
[27] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015, https://www.tensorflow.org/. software available from tensorflow.org.
[28] Chollet F. Keras. 2015, https://github.com/fchollet/keras.
[29] Nicolae MI, Sinn M, Tran MN, Buesser B, Rawat A, Wistuba M, et al. Adversarial robustness toolbox v1. 0.0. 2018, arXiv preprint arXiv:1807.01069.
[30] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.