This is the author's version of an article that has been published in IEEE Transactions on Emerging Topics in Computing.
Changes were made to this version by the publisher prior to publication. The final version of record is available at https://doi.org/10.1109/TETC.2021.3073017.

DAICS: A Deep Learning Solution for Anomaly Detection in Industrial Control Systems

Maged Abdelaty[α,β], Roberto Doriguzzi-Corin[α], Domenico Siracusa[α]
[α]ICT, Fondazione Bruno Kessler - Italy
[β]University of Trento - Italy

***Abstract*—Deep Learning is emerging as an effective technique to detect sophisticated cyber-attacks targeting Industrial Control Systems (ICSs). The conventional detection approach is to learn the "normal" behaviour of the system, to be then able to label noteworthy deviations from it as anomalies. However, the normal behaviour of ICSs continuously evolves over time for multiple reasons, such as update/replacement of devices, workflow modifications or others. As a consequence, the accuracy of the anomaly detection process may be dramatically affected with a considerable amount of false alarms being generated. This paper presents DAICS, a novel deep learning framework with a modular design to fit in large ICSs. The key component of the framework is a 2-branch neural network that learns the changes in the ICS behaviour with a small number of data samples and a few gradient updates. This is supported by an automatic tuning mechanism of the detection threshold that takes into account the changes in the prediction error under normal operating conditions. In this regard, no specialised human intervention is needed to update the other parameters of the system. DAICS has been evaluated using publicly available datasets and shows an increased detection rate and accuracy compared to state-of-the-art approaches, as well as higher robustness to additive noise.**

***Index Terms*—Anomaly detection, Domain shift, Deep Learning, Industrial control networks**

I. Introduction

Nefarious cyber-attacks targeting Industrial Control Systems (ICSs) may cause service downtime or material losses in industrial production sites, with potential negative repercussions on the lives of citizens. A notable example is the cyberattack against the Iranian nuclear program in 2010, perpetrated using the Stuxnet worm [1]. After infecting the Programmable Logic Controllers (PLCs), the worm used relevant information on the ICS to damage the centrifuges inside the plant by repeatedly changing their rotation speed. Another example is the cyberattack against the Ukrainian power grid in late 2015 [2]. The attackers exploited the BlackEnergy (BE) malware to compromise the information systems of three regional power distribution companies. They executed the malicious code to alter the firmware of specific control devices and instruct unscheduled disconnections from servers. The attack affected thousands of users and left them without electricity.

The increasing occurrence of such attacks and their complexity has motivated the development of intrusion detection solutions for ICSs based on machine learning techniques. Among the different approaches proposed in the scientific literature, a popular and powerful technique is the one-class classification. At the training stage, solutions based on one-class classification build a model that represents the normal behaviour of the ICS (the class of "normality"). At the production stage, the detection system uses that model to verify whether the behaviour of the live system matches the expected normal behaviour. Deviations from the normality, usually determined through thresholds on the classification error, are flagged as anomalies.

These algorithms can be very sensitive to abnormal behaviours, including zero-days anomalies or attacks and faulty devices or sensors, since they may evade the detection process. Moreover, real-world ICSs are dynamic and operate in noisy environments: these factors may hamper the correct functioning of the detection system because they can "shift" the normal behaviour. In such scenarios, two aspects are particularly challenging: (i) updating the model of normality upon changes of the ICS behaviour, and (ii) adjusting the thresholds according to the collected data. Previous studies either do not address such issues [3], [4] or require the intervention of human experts to adjust the model parameters [5], [6].

This paper tackles the aforementioned challenges by proposing DAICS (Deep learning Anomaly detection in Industrial Control Systems), an anomaly detection solution for ICSs based on the one-class classification paradigm. DAICS combines a database-oriented management approach and a deep learning architecture to model the normal behaviour of the ICS. The proposed neural network relies on *wide and deep* learning and *convolutional layers* to memorise and generalise the characteristics of the ICS. Wide and deep learning is a technique that has been recently proposed to improve the performance of recommender systems [7]. Convolutional layers, introduced by Yann LeCun in 1990 [8], apply small filters to detect patterns in the input while requiring a limited number of trainable parameters. Convolutional layers, and Convolutional Neural Networks in general, have become very popular in many application areas, including network security [9]–[11] and anomaly detection in ICSs [12]–[14].

As anticipated above, a common issue of one-class classification mechanisms is the high false alarm rate caused by the progressive evolution of the normal behaviour of the ICS with respect to the initial one, based on which the model was trained. This phenomenon, called *domain shift*, can be caused by changes in the industrial workflow, degradation of devices and communication links over the time, installation/removal of devices, updates in the devices' firmware or configuration, or to external noise on the communication channels. DAICS implements an automatic threshold tuning algorithm and the so-called *few-time-steps* algorithm, the latter based on the few-

shot learning paradigm [15], [16], to quickly update the model with the latest changes of the ICS normal behaviour.

We evaluate DAICS using datasets collected from the Secure Water Treatment (SWaT) and Water Distribution (WADI) testbeds, which are widely used for security research in ICSs [17]. The datasets comprise a training set containing only normal records, and a test set with normal and anomalous records. The anomalies in the test sets are real-world attacks targeting the integrity and the availability of the testbeds [18]. DAICS has been compared with other solutions in terms of anomaly detection accuracy and robustness to Gaussian noise.

The contributions of this work can be summarised as follows:

- An Anomaly Detection System (ADS) that has a modular architecture to fit large ICSs. The key components of the proposed ADS is a 2-branch neural network, in which a *wide branch* is designed to memorise the existing relations between the input features, while a *deep branch* generalises the model to unknown relations.
- An automatic threshold tuning technique that dynamically tunes the detection threshold based on the prediction error observed on the live ICS.
- A sensitivity analysis of hyperparameters of the proposed ADS using two real-world datasets, namely, the SWaT and WADI. The analysis provides practical insights to improve the performance under different operating conditions.

DAICS extends our previous work called AADS (Adaptive Anomaly Detection in industrial control Systems) [19]. With respect to AADS, DAICS scales better to larger ICSs, adopts an automated threshold tuning mechanism based on deep learning, and it has been tested on two public datasets: SWaT (like AADS) and WADI. The comparison presented in this paper shows that DAICS improves AADS in terms of precision (fewer false positives) and robustness to additive noise on the SWaT dataset.

The rest of this paper is organised as follows: Section II reviews state-of-the-art works on anomaly detection in ICSs. Section III defines the threat model used in this research. We describe the SWaT and WADI datasets in Section IV. The problem statement is provided in section V. Section VI introduces the proposed anomaly detection framework. Experimental setup and the evaluation results are presented in section VII. We conclude this paper in section VIII.

II. Related work

In this section, we review the state of the art on anomaly detection in ICSs. We focus our analysis on solutions for water treatment and distribution plants that have been validated on the SWaT and WADI datasets.

Goh et al. [5] propose an approach for anomaly detection in the SWaT testbed based on a Recurrent Neural Network (RNN). The Cumulative Sum (CuSum) technique presented in the paper sets upper and lower control limits for the prediction error of each device. An anomaly is detected when the cumulative sum is outside such limits. Besides suffering from a high false-positive rate, it might need a highly-specialised human intervention to tune and update the control limits.

Kravchik et al. [20] use convolutional and recurrent neural networks for anomaly detection in water treatment plants. The key aspect of the proposed solution is a statistical approach that relies on the normalised value of the prediction error. An anomaly is detected if the normalised error value exceeds a threshold defined by an expert. However, the authors do not discuss how the statistical values are updated in the case of changes in the production environment, quite frequent in real-world ICSs, as discussed in Section I. An extension of this work can be found at [4], in which the authors present their preliminary results (the paper has not been peer-reviewed at the time of writing) on SWaT, WADI and BATADAL datasets. In addition to the 1D Convolutional Neural Network proposed in the first paper, the authors explore the performance of two more models such as Undercomplete Auto-encoder (UAE) and windowed-Principle Component Analysis (PCA). We compare the performance of DAICS and these works in Section VII.

The approach presented in [21] is based on computing the distance between current and previous devices states. It is assumed that the distance should stay below a threshold during normal operations. The proposed solution has been tested only on the SWaT dataset, where it shows similar performance to DAICS in terms of F1 score (see Section VII-C). However, the number of detected attacks is not shown in the paper, preventing a full understanding of the effectiveness of the approach.

Shalyga et al. [6] present an anomaly detection solution for water treatment plants based on a Multilayer Perceptron (MLP) model. Anomalies are detected using a threshold applied to the weighted sum of the prediction errors of all sensors and actuators. Low weights are given to those devices whose normal behaviours are difficult to predict. Although this approach improves the performance on the SWaT dataset, it suffers from high false negative rates that reduce the number of detected attacks, as reported in Section VII.

Tabor [3] combines two different models, namely, Probabilistic Deterministic Finite Automaton (PDFA), and a Bayesian Network (BN). The anomaly detection is based on a combination of results from the two models. Also in this case, the authors do not address the problem of updating the model in case of changes in the normal operations of the ICS. Updating the model here appears significantly cumbersome due to the complex characterisation of the interaction between sensors and actuators needed to build the model.

Frequently ADSs rely on a threshold that specifies the maximum allowed prediction error during normal operations, above which a systems state is considered anomalous. The threshold is either selected empirically by an expert [20] or based on the prediction error on the training set [6], [21]. In this paper, we propose a technique for automated threshold tuning, which dynamically adjusts the threshold, improving the ability of DAICS to adapt to the normal behaviour evolution. In [22], [23], Markov chains and support vector regression are employed to dynamically tune thresholds for anomaly detection in network traffic (e.g., port scans, brute force attacks, SQL injections, etc.) and robot-assisted feeding. To the best of our knowledge, DAICS is the first anomaly detection solution for water treatment plants (and ICSs in general) to implement an automated threshold tuning mechanism.

In summary, a common drawback of available solutions is that they are not flexible enough to quickly and efficiently

adapt to changes in the production environment. In a water treatment plant, examples of such changes are the deployment of a new water tank, which influences the behaviour of the sensors attached to it, or the replacement of a motorised valve with another with different operation modes. Our approach implements an algorithm called *few-time-steps*, described in Section VI-E, which updates the model of normality according to the changes in the normal behaviour of the ICS. The proposed algorithm uses a small amount of data to update the weights of the neural network and requires minimal human intervention. This is supported by an automated threshold tuning technique, which adds to DAICS ability to cope with the dynamic behaviour of industrial environments. Moreover, the proposed neural network architecture is designed to be scalable, so it can model the normal behaviour of large ICSs.

III. Threat Model

We assume that the attacker's capabilities include gaining remote access to the networked control system, to the Supervisory Control and Data Acquisition (SCADA) workstation, or can physically compromise sensors and actuators within the ICS. We also assume that the attacker has domain knowledge of the targeted ICS, e.g., the physical property measured by each sensor and the physical consequences of actuation commands. The attacker's goal is to use the above capabilities and knowledge to damage or modify the ICS operations. This includes: (i) altering the state of actuators through a MITM-like attack, in which the actuator receives commands from the attacker instead of a PLC; (ii) sending spoofed sensors readings to the PLC to drive the PLC to take wrong decisions; and (iii) tampering with the PLC firmware aimed to take the ICS out of service or to change the programmed logic (e.g., the Stuxnet worm [1]). However, the attacker does not have enough knowledge of the ADS to perform adversarial attacks, i.e. to evade the detection logic by introducing noise in the system or by generating crafted data that keeps the anomaly metric below the detection threshold [24]–[26]. Indeed, robustness to adversarial attacks is an important problem that is beyond the scope of this study.

IV. Datasets

We evaluate DAICS using two popular public real-world datasets, namely Secure Water Treatment (SWaT) and Water Distribution (WADI). In this section we provide a short introduction to the main properties of the two datasets, also summarised in Table I.

TABLE I: Properties of the SWaT and WADI datasets.

Dataset	Records	Duration (days)	Attacks	Sensors	Actuators
SWaT	946,722	11	36	25	26
WADI	1,209,610	16	15	69	54

A. The SWaT dataset

This dataset has been collected from the SWaT testbed, a reduced version of an operational clean water treatment plant [17], [27]. The water treatment process is monitored by a SCADA workstation and is divided into six sub-processes, including raw water supply and storage, ultra-filtration and backwash. The sub-processes are controlled by a pair of PLCs that communicate with sensors (water flow indication transmitters, level indicator transmitters, analyser indicator transmitters, such as pH analysers, and ultra-violet modules) and actuators (e.g., pumps, motorised valves). The PLCs collect the readings from the sensors that monitor the status of the physical process, and send actuation commands to the actuators. Based on the PLC's internal logic, such commands may be used to either change or keep the current state of an actuator.

The dataset consists of 946,722 records of sensors and actuators collected during 11 days of operation at a rate of one sample per second. Each record contains 51 attributes, representing 25 sensors readings and 26 actuators states. The dataset is divided into a 7-day portion of normal operations, which has been used as the training set, plus a 4-day portion of normal activity combined with 36 attacks generated by following the attack model in [18] (the test set). 20% of the training set is used for validation (validation set).

The attacks in the test set are of duration ranging between two minutes and nine hours, and involve one or multiple devices at the same time. Attacks include spoofing the readings of sensors or reversing the operation states of actuators. For instance, in the first attack, the state of a motorised valve is switched from closed to open for 15 minutes aiming to cause a tank overflow. In another attack, the value reported by a water level sensor is decreased by 0.5mm/sec for seven minutes, again with the intention of causing a tank overflow. A more detailed description of these attacks is available in [27], [28].

B. The WADI dataset

The WADI testbed is a reduced version of a real-world water distribution network [17], [29]. The testbed network consists of three sub-processes, or stages, including a primary grid for water supply, a secondary grid for water distribution to six consumer tanks, and a return-water grid that handles the excess of water from the consumer tanks. Similarly to the SWaT testbed, also WADI is supervised by a SCADA workstation. Moreover, each sub-process is controlled by a PLC communicating with sensors and actuators. Sensors include water flow indication transmitters, level indicator transmitters, analyser indicator transmitters and pressure meters. Like the SWaT testbed, actuators include pumps and motorised valves.

The WADI dataset comprises 1,209,610 records, each with 123 attributes divided into 69 sensors readings and 54 actuators states collected during a period of 16 days. Normal operation conditions have been recorded during the first 14 days, split into training (95%) and validation (5%) sets. The last two days, with normal activity and 15 attacks, are used as the test set. The attacks follow the same model used in the SWaT dataset and described in [18], although with shorter duration (between 2 and 30 minutes). For instance, one attack cuts off the water supply to consumer tanks; another one aims at increasing the level of chemicals in the water by turning off a sensor and sending false readings to the PLC for ten minutes. More details and examples on the WADI attacks can be found in [29].

V. Problem Statement

Unsupervised anomaly detection solutions for ICSs usually rely on the so-called *one-class classification* technique. The basic idea is to build a model of the normal behaviour of the industrial process and to consider as anomalous every event that does not fit the model. The main challenge with such approaches is dealing with the *domain shift* (also called *concept drift*), which originates from a gap in the data distribution between the training and unseen data [30], [31]. A tangible consequence of the domain shift is an increase of false alarms generated by the anomaly detection system due to normal events classified as anomalies.

The domain shift problem is present in the SWaT dataset, where we can observe changes in the normal behaviour of some devices across the training and test sets. For instance, during the normal operation, the pump P102 has a single state of value 1 in the training set, then it takes an additional "normal" state of value 2 in the test set. Also, the probability distribution of some sensors changes between the two sets. For example, in the training set the output of the Analyser Indication Transmitter AIT402 ranges in the interval [153, 236] mV, while in the test set it ranges in [141, 328] mV with a substantially different distribution, as illustrated in Fig. 1. Another relevant example is the presence of redundant devices, such as the redundant pump P102 in the SWaT testbed. A redundant pump is always off until the primary pump stops working for any reasons (hardware or software faults, power outage, etc.). In such situations, the PLC turns on the redundant pump to take over the work of the primary pump. If this process is not covered in the training set, the forecasting model will consider the operations of the redundant pump as anomalies.

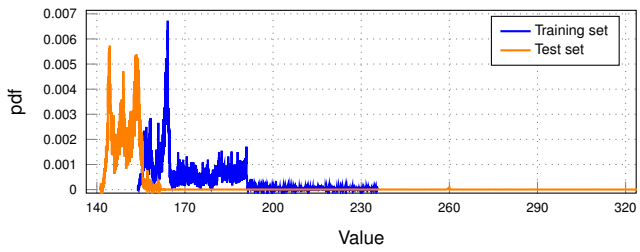

Figure 1: The probability density function for sensor AIT402 of the SWaT testbed. It shows the variation in the normal behaviour between training and test sets.

The domain shift problem can also be observed in the WADI dataset. For instance, the records of one of the *Flow Indication Transmitters* range within [0, 2.3] $m^3/hour$ of water in the training set, while the range changes to [0, 3.3] $m^3/hour$ in the test set. The exact motivation of the domain shift problem in the two datasets is not described in the documentation. However, as these behavioural changes are not part of the datasets' threat models, and as we observe them only in normal operating conditions, we assume they are not caused by malicious actions.

Modelling of the normal behavioural evolvement is still a challenge for the implementation of the anomaly detection solutions in real-world settings [32], [33]. Existing approaches have tackled the domain shift problem by only focusing on adjusting the parameters of the detection algorithms, or by excluding from the modelling process those devices that present the domain shift problem, as discussed earlier in Section II. However, we argue that adjusting the detection parameters without updating the model of normality is not sufficient to cope with dynamic environments such as ICSs.

In the next section, we present our approach to tackling the *domain shift* problem. The main idea is to follow the changes in the industrial process by automatically updating the model of normality and the anomaly threshold.

TABLE II: Glossary of symbols.

WDNN	Wide and Deep Neural Network
TTNN	Threshold Tuning Neural Network
W_{in}	Input time window
W_{out}	Output time window
W_{anom}	Anomaly waiting time
H	Horizon
G	Number of output section of WDNN
$\mu_{out}[g]$	Output section g of WDNN
m_{ac}	Total number of actuators
m_{se}	Total number of sensors
m_{se}^{g}	Number of sensors of output section g
$\text{MSE}_{g,t}$	Prediction error on section g at time t
$\text{MSE}_{g,[a,b)}$	List of prediction errors on section g between time a and $b-1$
T_g	Anomaly threshold for section g
ν_t	Actuator states recorded at time t

VI. The DAICS framework

DAICS builds the model of normality by combining a neural network with a database-like approach. Deep learning is used to model the continuous readings of sensors, whose values are sampled periodically by a PLC, while the database stores the states of actuators controlled by the PLC. Moreover, DAICS implements the few-time-steps learning algorithm to update the model based on the evolution of the ICS. Anomaly detection involves comparing the actual behaviour of the system against the model. Deviations beyond a dynamic threshold are considered as anomalies.

A. Prediction of sensor states

The normal behaviour of sensors is modelled using a deep neural network named Wide and Deep Neural Network (WDNN), which predicts the normal states of sensors based on the past states of both sensors and actuators. The idea of wide and deep neural networks was introduced in [7] to build a recommender system that suggests apps based on the user's query and preferences. Our WDNN, shown in Fig. 2, has two goals: memorization through the wide branch and generalization through the deep branch. Memorization means learning the relationship between feature-pairs in the training set, hence recording the co-occurrence of combinations of sensors values. Generalization means the ability to explore relationships that do not exist in the training set.

Like in our previous work [19], the neural network comprises a *Feature extractor* section that learns the relations between all sensors and actuators during the normal operation of an ICS and extracts the features required to predict the normal states of sensors. Instead, the output section is split into multiple

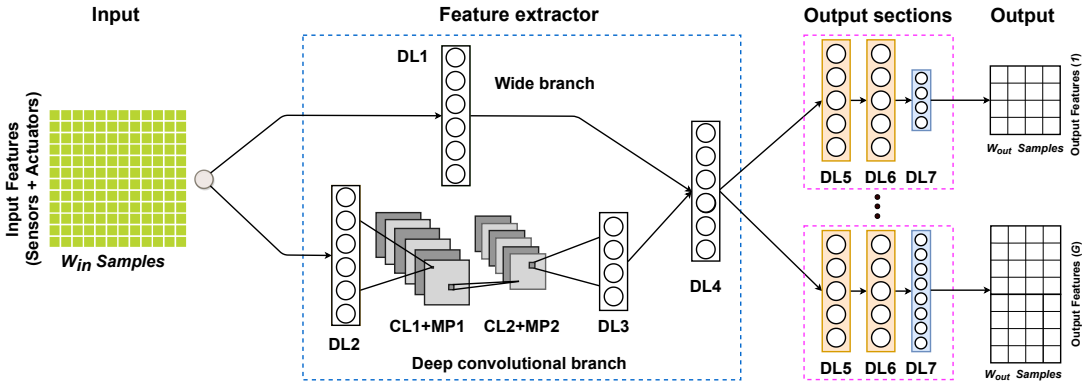

Figure 2: Architecture of Wide and Deep Neural Network (WDNN). The prediction of the sensors states is computed combining the states of both sensors and actuators. The size of DL7 varies according to the number of output features of the output section.

branches to serve large-scale ICSs where the sensors are controlled by several PLCs, usually specialised on a specific part of the industrial process with specific behaviour and anomaly threshold.

Architecture. The neural network takes as input an array X of data samples collected during a time window of length W_{in} seconds, corresponding to W_{in} samples, as the dataset was collected at a sampling rate of one sample per second. The size of X is $m \times W_{in}$, where $m = m_{se} + m_{ac}$ is the number of features for each sample including the state of m_{se} sensors and m_{ac} actuators taken at time t. Observe that we use the states of both sensors and actuators because the future states of sensors depends on their current states and the actions taken by actuators. The output Y is the expected normal states of sensors during a future time window W_{out}. Both time-windows W_{in} and W_{out} are separated by a time interval called horizon H. The purpose of H is to prevent the neural network from replicating the last values of the input time window W_{in} into W_{out}, as pointed out by Shalyga et al. [6].

As shown in Fig. 2, the *Feature extractor* section comprises two convolutional layers and four fully connected layers organised into two branches. The fully connected layer DL1 represents the *wide branch*, which learns the normal interactions between sensors and actuators using cross-product transformations between the input features.

The *deep branch* allows the WDNN to generalise to events not covered in the training set, hence to reduce the prediction error in the case of new normal combinations of input states. This branch comprises the fully connected layer (DL2), two one-dimensional convolutional layers (CL1 and CL2), each one followed by a max-pooling layer (MP1 and MP2), and the fully connected layer (DL3). As shown in other works (e.g., [20]), one-dimensional Convolutional Neural Networks (CNNs) are particularly suited for modelling time series data. The purpose of layers CL1 and CL2 is to model the data collected from sensors and actuators in a specific time window. With max-pooling, we down-sample the output of each convolutional layer by a factor of 2. Finally, the fully connected layer DL3 re-shapes the output of MP2 to allow its concatenation with the output of the wide branch.

Both branches are aggregated in another fully connected layer (DL4) followed by multiple dense output sections, each one consisting of the three fully connected layers DL5, DL6 and DL7. Each output section learns the most relevant information from the aggregation layer in order to predict the normal behaviour of a group of sensors controlled by a specific PLC. Each fully connected layer can be described as $Y = LeakyReLU(\mathbf{W}^T X + \mathbf{b})$, where Y is the output, X input, $\mathbf{W}$ is an array of weights the model learns during the training, and $\mathbf{b}$ is the bias. We introduce non-linearity in the model by using the leaky rectified linear activation function defined as follows: $LeakyReLU(x) = max(0, x) + 0.01 * min(0, x)$. Similarly, the convolutional layers can be described as $Y_i = LeakyReLU(Conv(X, \mathbf{W}_i, \mathbf{b}_i))$, where Y_i is the output of the convolution on the input X using the i-th filter with weights $\mathbf{W}_i$ and bias $\mathbf{b}_i$.

Cost function. The neural network presented above has been trained to find the set of weights and biases that minimises the Mean Square Error (MSE) cost function. The cost function computes the error between the model predictions on sensors readings and the corresponding observed values. Hence, by minimising the cost, we reduce the prediction error. At the training stage, the cost function for a batch size of s samples (i.e., s different time windows) in a specific output section g can be formally written as:

$$c_g = \frac{1}{s} \sum_{t=1}^{s} \left(\frac{1}{m_{se}^g} \sum_{i=1}^{m_{se}^g} (Y_t[i] - \tilde{Y}_t[i])^2 \right) \quad (1)$$

where $Y_t[i]$ is the predicted value for sensor i at sample t (i.e. time-step t), while $\tilde{Y}_t[i]$ is the corresponding observed sensor value (the value present in the training set). m_{se}^g is the number of sensors in an output section g. The final cost c is the sum of costs of all G output sections $c = \sum_{g=1}^{G} c_g$, which is minimised using Stochastic Gradient Descend (SGD) [34].

B. Anomaly detection in actuators

The SWaT and WADI testbeds include discrete actuators such as pumps and motorised valves. Actuators in the SWaT and WADI testbeds include pumps and motorised valves. The pumps are arranged in pairs of primary and redundant hot-standby pumps. A redundant pump is turned on only in the case

the respective primary pump stops working. This operational mode complicates building a forecasting model that predicts the actuator states, as some actuators (such as the redundant pumps), are rarely used during the normal operations. As a consequence, after measuring a high prediction error with Deep Learning (DL)-based methods due to lack of normal operation records, we designed a light and straightforward approach based on querying a database containing all the normal actuator states. A database entry is n-tuple, where n is the number of actuators in the testbed ($n = 26$ in the SWaT, $n = 54$ in the WADI). Each entry is a combination of actuators states labelled as normal, for a total of 146 entries available in the SWaT dataset and 2001 entries in the WADI dataset. An example of tuple from the SWaT testbed is provided in Table III.

TABLE III: Tuple in the database of actuators normal states.

Actuator	MV101	P101	P102	...	P601	P602	P603
Tuple	2	2	1	...	1	1	1

At testing time, the combinations in the test set with no occurrences in the training set are marked as anomalies, as explained in Section VI-C.

As demonstrated in Section VII, this approach works well with benchmark datasets such as SWaT and WADI, where only discrete actuators are present. Nevertheless, real-world ICSs may also comprise actuators with continuous output values (e.g., continuous valve actuators [35]). Due to the huge number of possible output combinations, a database-oriented does not seem very practical for such deployments. With enough training data, the output of continuous actuators can be predicted with WDNN, as done for the continuous sensors states. Of course, this requires the extension of one or more WDNN output sections to support the actuators states, depending on the adopted ICS partitioning scheme (as also discussed in Section VI-A for the prediction of sensor states).

C. Detection logic

DAICS operates on batches of actuators and sensors values retrieved from the PLCs at regular time intervals of duration s seconds. For the sake of simplicity, we assume that s also corresponds to the number of readings in one batch for each device (like in the two datasets used in the experiments). We therefore define $I = [\bar{t}, \bar{t} + s)$ the time interval under analysis (where $[a, b)$ indicates the interval between a and $b-1$). Given the observation time $t \in I$, DAICS verifies whether the current combination of actuator values is present in the database A built at the training stage. An anomaly is reported otherwise.

In the case of the sensors, the values observed at time t are compared against those predicted by WDNN using past values of sensors and actuators, as previously explained in Section VI-A. More precisely, the values observed at time $t \in I$ are compared with the first element of the output of WDNN obtained from values of sensors and actuators collected in the time window $[t', t'+W_{in})$ (one input sample for WDNN), where $t' = t - H - W_{in}$. The MSE value of each WDNN output section is evaluated against an adaptive threshold to determine the presence of any anomalies.

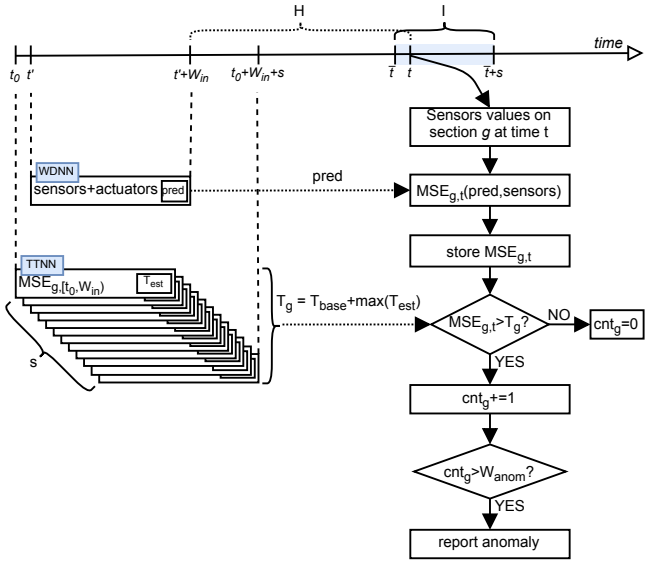

Figure 3: Detection process on section g of sensors through WDNN and adaptive thresholds.

The detection process on one single output section g of the neural network is summarised in Fig. 3, which also depicts an adaptive threshold mechanism for automatically tuning the anomaly threshold based on observed noise conditions on the sensors readings. Such a mechanism is described in Section VI-D. The region of the timeline highlighted in light blue represents the time interval I described above.

More precisely, the samples observed on the group g of sensors at time t are identified as anomalous when the anomaly condition $\text{MSE}_{g,t} > T_g$ is met, where T_g is a threshold on the prediction error for the group of sensors g. Using the same notations as for the cost function in Equation 1, we define $\text{MSE}_{g,t}$ as follows:

$$\text{MSE}_{g,t} = \frac{1}{m_{se}^g} \sum_{i=1}^{m_{se}^g} (Y_t[i] - \tilde{Y}_t[i])^2 \tag{2}$$

To reduce the false positives caused by sudden changes in the underlying physical process (as also observed by Kravchik et al. [20]), DAICS reports an anomaly at time t only if the anomaly condition has also been previously observed for W_{anom} consecutive sampling intervals. In Fig. 3, this process is represented through increasing the counter cnt_g.

In summary, the anomaly condition can be expressed as:

$$L_t = \begin{cases} 1, & \text{if } \exists g \in [1, G] \text{ s.t.} \\ & \text{MSE}_{g,i} > T_g \ \forall \ i \in [t - W_{anom}, t] \\ 1, & \text{if } \nu_t \notin A \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where ν_t is the combination of the actuators values observed at time t, while $L_t = 1$ defines the anomaly condition at time t. W_{anom} is one of the hyper-parameters used to tune DAICS.

D. Threshold Tuning

The industrial environments are prone to electromagnetic noise that can interfere with the operations of the ICS, hence

hampering the accuracy of anomaly detection systems [36], [37]. In this regard, a major challenge for such systems is finding the threshold to classify an event either as an anomaly or as normal activity. Existing approaches tackle this problem through empirically, by adjusting the threshold at test time as a hyper-parameter [5], [6], [20]. While empirical thresholds produce good results in the laboratory with static datasets such as SWaT and WADI, production systems can hardly afford long threshold tuning sessions upon new noise levels. Instead, here we propose an adaptive technique to dynamically tune the threshold based on the prediction error trend.

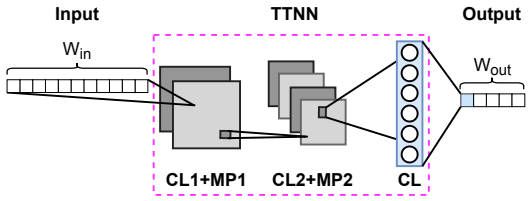

Figure 4: Architecture of Threshold Tuning Neural Network (TTNN). Only the first element of the output is kept. The size of the 1-dim kernel of CL1 and CL2 is 2 with stride 1.

Ali et al. [22] used a machine learning technique to estimate the threshold based on the historical prediction error of their ADS. In this paper, we treat the prediction error MSE as a univariate time series that is modelled using a neural network that we called TTNN, whose architecture is depicted in Fig. 4. We use G instances of TTNN, one for each output section of the WDNN model, to tune the thresholds $T_g,\ g \in [1, G]$. Each instance is trained with the prediction error MSE_g measured on a single output section g on the validation set, which only contains benign records, as the training set. The trained model is used in the online system to compute the optimal anomaly threshold T_g using past prediction errors. As represented in Fig. 3, the threshold T_g used for anomaly detection on sensors values collected in time interval $[\bar{t}, \bar{t}+s)$ is obtained using past prediction errors computed in time interval $[t_0, t_0 + s + W_{in})$, where $t_0 = \bar{t} - H - W_{in}$.

As shown in Fig. 4, TTNN consists of two 1-dimensional convolutional layers, each followed by a max pooling layer, and of one final fully connected classification layer, represented in the figure as CL1, MP1, CL2, MP2 and DL respectively. The input is a time series of prediction errors measured for the samples collected in past time window of length W_{in} seconds, the output contributes to the estimation of the optimal anomaly threshold for the current sensors states. A detailed description of the threshold tuning process is described in Algorithm 1.

Referring to the timeline depicted in Fig. 3, the anomaly threshold used at time $t \in [\bar{t}, \bar{t}+s)$ for output section g is computed by using a batch E_g of s samples defined as time series of past prediction errors of length W_{in}, starting from $\text{MSE}_{g,[t_0,t_0+W_{in})}$, $\text{MSE}_{g,[t_0+1,t_0+W_{in}+1)}$, to $\text{MSE}_{g,[t_0+s,t_0+W_{in}+s)}$.

At line 3 in of the algorithm, a median filter is applied to the input batch of past prediction errors E_g to reduce the impact of short-term changes of the prediction error on the threshold tuning process. Median filtering is accomplished by

Algorithm 1 Threshold tuning algorithm

Input: Batch of past prediction errors on section g ($E_g = \{\text{MSE}_{g,[t_0,t_0+W_{in})},...,\text{MSE}_{g,[t_0+s,t_0+W_{in}+s)}\}$), where $t_0 = \bar{t} - H - W_{in}$

Output: Anomaly threshold for WDNN output section g (T_g)

1: $T_{est} = []$; ▷ List of estimated thresholds
2: TTNN$[g] \leftarrow$ load the weights and biases for section g;
3: $\bar{E}_g \leftarrow \text{median}(E_g)$ ▷ Median filter on the input
4: **for** $X = \text{MSE}_{g,[t_0+i,t_0+W_{in}+i)} \in \bar{E}_g$ s.t. $i \in [0, s)$ **do**
5: $\hat{Y} = \text{TTNN}[g](X)$; ▷ Estimated prediction error
6: $T_{est}.append(\hat{Y}[0])$;
7: **end for**
8: $T_g = T_{base} + max(T_{est})$;

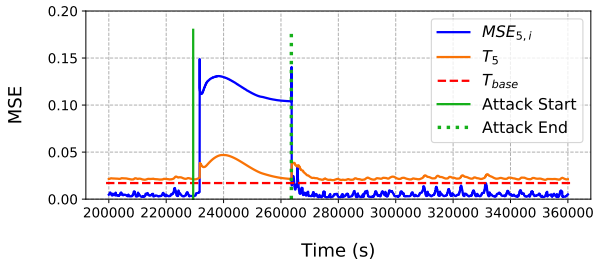

Figure 5: T_g and $\text{MSE}_{g,i}$ of the 5th WDNN's output section.

sliding a window over E_g while computing the median value of the elements of E_g under the window. The resulting median values are stored in array $\bar{E}_g$. One variable to consider in this process is the size of the median filter. We experimented with values between 50 and 120. As reported in Table IV, we obtained the best results with the median filter size of 59. In the loop at lines 4-7, the algorithm computes the estimated prediction error using the median values in $\bar{E}_g$. As shown at line 6, only the first element of the output layer is memorised for further processing. The threshold T_g is computed at line 8 as the maximum of the predicted thresholds plus the offset T_{base}. T_{base} is set as the sum of the mean plus the standard deviation of the prediction error on the validation set, and is kept constant after the deployment. This offset is added to reduce the sensitivity of the WDNN to small variations on sensors values. As an example, Fig. 5 shows how the trend of threshold T_5 (the threshold for the output section number 5) follows the prediction error $\text{MSE}_{5,i}$ without going below it, except in presence of anomalies or malicious activities.

As a final step, the algorithm executes one SGD epoch to update weights and biases of each of the G instances of TTNN using the respective input batch E_g.

E. The few-time-steps algorithm

The few-time-steps algorithm has been designed to efficiently reconfigure DAICS in a production environment in the case an anomaly is identified by the system and then recognised as a false alarm by the technician.

In such a situation, DAICS is triggered by the technician to update the database A of actuators states and to tune the output section of the pre-trained WDNN. The only assumption is that

the technician can recognise false alarms caused by changes in the normal operating condition of the ICS (e.g., changes in the hardware/software configurations). Unlike state-of-the-art solutions, the proposed approach only assumes a domain-specific understanding of the ICS operations, without requiring a deep knowledge of the algorithms and their thresholds and hyper-parameters.

Of course, the number of human interventions needed to handle false alarms is an important parameter to determine the usability of the anomaly detection system. If the system is too sensitive, the technician would be overwhelmed by a large number of alarms to be verified, making it hard to spot those that are due to normal changes in the ICS behaviour, hence false. On the contrary, a conservative approach might not reveal true malicious activities or anomalies. Both cases can render the system ineffective, if not unusable. The sensitivity of DAICS to this parameter will be analysed in Section VII.

Algorithm 2 Few-time-steps learning algorithm

Input: Batch of samples (S), Technician's input: False alarm at time t ($FA_t[i]$) $i \in [0, G]$
Output: Retrained output section ($\mu_{out}[g]$)

1: **if** $\nu_t \notin A$ **and** $FA_t[0] ==$ **True then**
2: $A = A \cup \nu_t$; ▷ Update the database of actuators
3: **end if**
4: $EP_{tuning} \leftarrow$ number of fine-tuning epochs;
5: **for** $g \in [1, G]$ **do**
6: **if** $FA_t[g] ==$ **True then**
7: $\mu_{out}[g] \leftarrow$ weights and biases
8: **for** epoch **in** EP_{tuning} **do**
9: $c_g \leftarrow$ cost of section g on S;
10: $SGD(\mu_{out}[g])$ step to minimise c_g;
11: $\mu_{out}[g] \leftarrow$ update weights and biases;
12: **end for**
13: **end if**
14: **end for**

Algorithm 2 is called upon a technician's decision that an alarm detected by the system at time t is false. This means that Algorithm 2 is called when both of the following conditions are satisfied: the anomaly condition $L_t = 1$ and at least one of the elements of the Boolean vector $FA_t[i]$ ($i \in [0, G]$) is *True*. FA_t indicates where the false alarm has been detected, either among the actuators ($FA_t[0] = True$), or in one or more groups of sensors ($FA_t[i] = True,\ i \in [1, G]$), or both.

In the first case, with $FA_t[0] = True$, the algorithm adds the combination of actuator states ν_t to database A (lines 1-3), as the technician has determined that ν_t is normal and must be treated as such by the system. In lines 5-14, the output sections of the neural network that have produced the false alarm are updated. Specifically, in lines 8-12, SGD is employed to fine-tune the output section through multiple gradient steps. We calculate the prediction loss for the data samples aggregated in a *batch* of S samples containing the false alarm (S is passed as an input to the algorithm). The optimiser minimises this loss by tuning the parameters of the output layers DL, DL6, and DL7. After EP_{tuning} optimisation steps (around 400 *ms* on average for 100 epochs on our testing environment described in Section VII-A), the updated output section $\mu_{out}[g]$ replaces the previous one in the anomaly detection process (line 11). Note that, grouping the sensors into different sections speeds up the execution of the few time steps algorithm, since only a portion of the output section is updated in the case of false alarms.

VII. Experimental evaluation

In this section, we present a detailed evaluation of DAICS obtained using the SWaT and WADI datasets presented in Section IV. The evaluation comprises a comparison with state-of-the-art solutions in terms of detection accuracy, number of detected attacks and robustness to noisy data.

A. Experimental Setup

DAICS has been implemented in PyTorch 1.0 [38] and validated using a Singularity container [39] running in a shared machine configured with 16 CPU cores, 64 GB virtual RAM and an NVIDIA 1080Ti GPU. The database of actuators normal states A is implemented as a NumPy array populated using the training datasets. The maximum size of the array is less than 1MB for each of the two datasets. Prior to our experiments, we also normalised the sensor readings between 0 and 1.

B. Methodology

As per convention in the literature, we configure and evaluate DAICS using the following metrics:

$$Pr = \frac{TP}{TP + FP} \quad Re = \frac{TP}{TP + FN} \quad F1 = 2 \cdot \frac{Pr \cdot Re}{Pr + Re}$$

where *Pr=Precision*, *Re=Recall*, *F1=F1 Score*, *TP=True Positives*, *FP=False Positives*, *FN=False Negatives*. It is important to highlight that we adopt a point-based approach to compute these metrics. Thus, precision, recall and F1 score are computed using the labelled records (points) of the datasets, which report whether a record (a combination of actuators states and sensors readings recorded at a given time) is normal or anomalous. Like most of related works in the literature, we adopt this approach to tune the hyper-parameters of WDNN and TTNN models, to assess the overall performance of DAICS and for comparison with the state-of-the-art.

We used a grid search strategy to explore the set of hyper-parameters, including time windows W_{in}, W_{out} and W_{anom}, learning rate, batch size, number of layers in the output sections, number of neurons in each layer, and others listed in Table IV. In the table we report the final values that maximise the F1 score on the two datasets. These hyper-parameters are kept constant throughout our experiments presented below in this section, except when we study the sensitivity of DAICS to a specific hyper-parameter.

In our experiments, we evaluate the sensitivity if DAICS to noise. In fact, ICSs operate on harsh industrial environments [40], [41], where the communication channels are often subject to interference (e.g., in the case of employing wireless communication devices [41]). For evaluation purposes, we assume that the sensors measurements of the SWaT and WADI datasets can be perturbed with Gaussian noise with mean

TABLE IV: Hyperparameters.

Hyperparameter	**SWaT**	**WADI**
Horizon (H)	50	20
Input time window (W_{in})	60	50
Output time window (W_{out})	4	4
Anomaly time window (W_{anom})	30	30
WDNN		
Hyperparameter	**SWaT**	**WADI**
Learning rate (α)	0.01	0.001
Batch size (s)	32	32
Tuning epochs (EP_{tuning})	100	100
Output sections layers	3	3
DL1 Neurons	W_{out}	W_{out}
DL2 Neurons	$3 * W_{in}$	$3 * (m_{se} + m_{ac})$
DL3 Neurons	W_{out}	W_{out}
DL4 Neurons	80	80
DL5 Neurons	2.25*m^g_{se}	2.25*m^g_{se}
DL6 Neurons	1.5*m^g_{se}	1.5*m^g_{se}
DL7 Neurons	m^g_{se}	m^g_{se}
CL1 Kernels, Kernel size	64, 2	64, 5
MP1 Pooling size	(64, 2)	(64, 2)
CL2 Kernels, Kernel size	128, 2	128, 5
MP2 Pooling size	(128, 2)	(128, 2)
TTNN		
Hyperparameter	**SWaT**	**WADI**
Learning rate (α)	0.01	0.01
Batch size (s)	32	32
Tuning epochs (EP_{tuning})	1	1
Median kernel size	59	59
CL1 Kernels, Kernel size	2, 2	2, 2
MP1 Pooling size	(2, 2)	(2, 2)
CL2 Kernels, Kernel size	W_{out}, 2	W_{out}, 2
MP2 Pooling size	(W_{out}, 2)	(W_{out}, 2)
DL Neurons	1	1

$\mu = 0$ and standard deviation $\sigma \in \{1, 2, 3, 5, 10, 15\}$. The noise distribution and the values of μ and σ have been selected following similar assumptions as in other studies on noisy networked control systems [36], [37].

The validation presented below is divided into three different experiments. In Experiment 1 and 2, we compare the performance of DAICS with relevant works in the state-of-the-art in terms of precision, recall, F1 score, and the number of attacks correctly detected. Experiment 1 evaluates DAICS using the SWaT dataset, while experiment 2 uses the WADI dataset. In Experiment 3, we evaluate the robustness of DAICS to additive noise applied on the SWaT and WADI test sets. We also compare the results with the frameworks proposed in [19], [20] in case of the SWaT dataset.

C. Experiment 1: Detection accuracy in the SWaT

As shown in Table V, DAICS outperforms existing state-of-the-art detection on the SWaT test set with 88.9% F1 score, and correctly recognises 33 out of 36 attacks in the test set. It is worth noting that DAICS also detects a higher rate of attacks during their execution (97% against 93% of [20]), hence allowing the operator to activate the adequate countermeasures more promptly. Two of the three attacks missed by DAICS (both attacks to motorised valves) are marked as unsuccessful in the dataset documentation, as they do not cause any noticeable changes to the ICS. The third one tries to cause the overflow of a tank by decreasing the values of a water level sensor. Compared to other similar attacks in the dataset, this one is shorter and starts when the water level is already decreasing, hence causing a little deviation from the normal behaviour.

TABLE V: State-of-the-art comparison (SWaT dataset). In parenthesis, the number of attacks detected after their end.

Architecture	**Precision**	**Recall**	**F1**	**Detected attacks**
DAICS	0.9185	0.8616	0.8892	33(1)
AADS [19]	0.866	0.861	0.863	33(1)
MLP [6]	0.967	0.696	0.812	25
CNN [20]	0.867	0.854	0.860	31(2)
TABOR [3]	0.861	0.788	0.823	24
Windowed-PCA [4]	0.92	0.841	0.879	-
Online-ADS [21]	0.8638	0.9064	0.8846	-

Few-time-steps algorithm. We also evaluated the contribution of the few-time-steps algorithm to the detection accuracy of DAICS. To this aim, we repeated the experiment disabling the algorithm. As the SWaT test set contains normal records that are not present in the training set, the model of normal behaviour built with the training set leads to several false positives and, more precisely, to a low F1-score measure of 77.7% and 1251 false alarms (compared to the 645 experienced with the algorithm enabled).

The update process is fast. Indeed, with our setup, the execution of one cycle of the few-time-steps learning algorithm, including the $EP_{tuning} = 100$ epochs, with a batch of 32 samples takes 400 ms on average. In an online system, the algorithm can be triggered by the technician upon identifying a false alarm, e.g. caused by a known unstable device. It is important to emphasise that this is the only requirement for the technician, unlike other approaches in which an in-depth knowledge of the underlying algorithms is necessary to update detection thresholds or other parameters.

Hyper-parameters. The sensitivity of DAICS to the hyper-parameter W_{anom} is presented in Fig. 6. We remind that W_{anom} is defined in Equation 3 as the number of consecutive times the prediction error MSE is higher of the anomaly threshold on the same output section of WDNN.

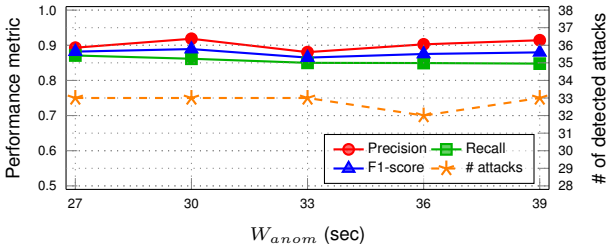

Figure 6: Sensitivity of WDNN to W_{anom} on SWaT dataset.

Although the number of detected attacks is quite stable to 33 (except for W_{anom}=36), the highest F1 score (0.8892) is measured at W_{anom}=30.

False alarms. DAICS relies on the feedback from the technician to fine-tune the output sections with new information of the normal behaviour. Of course, the rate of technician's interventions is a relevant metric to consider, as too frequent

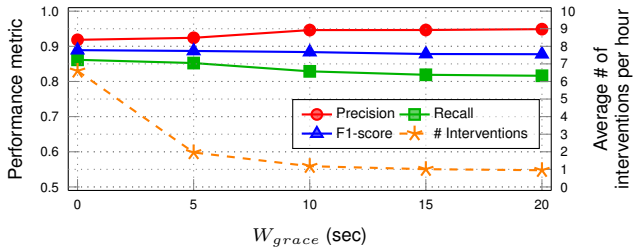

Figure 7: Sensitivity of WDNN to W_{grace} on SWaT dataset.

false positives can undermine the usability of the system. With the chosen settings, we measured 645 false alarms in total on the test set of the SWaT dataset, equivalent to 6.6 human interventions/hour on average. Although this seems a manageable rate of intervention, in more complex industrial environments compared to the SWaT testbed.

One way to reduce the rate of intervention is to allow a *grace time* W_{grace}, through which alarms are reported to the technician only if they last for at least W_{grace} seconds. Of course, this practice may hide short true positives, hence preventing the detection of a certain number of anomalies.

We tested the sensitivity of DAICS to the grace time by varying W_{grace} from 1 to 20 seconds. In Fig. 7 we can observe a minimal decrease of the F1 score (0.877 with W_{grace}=20) and a consistent reduction of the technician intervention rates (from 6.6/hour with no grace time, to 1.96/hour with W_{grace}=5, to 0.95/hour with W_{grace}=20). Moreover, we did not experience any variations in the number of detected anomalies with W_{grace}=5 (still 33), while with W_{grace}=20 this number decreases to 28, as somehow expected.

D. Experiment 2: Detection accuracy in the WADI

DAICS matches the state of the art results in terms of number of attacks detected, 14 out of 15 attacks in the WADI dataset, while it outperforms the other solutions in terms of precision, recall and F1 score, as summarised in Table VI. The undetected attack is meant to cause a water leakage by opening a motorised valve. However, as the attack does not succeed, no anomalous patterns are observed in the ICS behaviour.

TABLE VI: State-of-the-art comparison (WADI dataset).

Architecture	Precision	Recall	F1	Detected attacks
DAICS	0.9083	0.7205	0.8036	14
AADS [19]	0.7794	0.6591	0.7142	14
MAD-GAN [42]	0.4144	0.3392	0.37	-
1D CNN [4]	0.697	0.731	0.714	14
AE [4]	0.834	0.681	0.750	14

Few-time-steps algorithm. Given the setup described in section VII-A, DAICS takes 800 ms to fine-tune an output section with 100 epochs and a batch of 32 samples. Like in the case of the SWaT dataset, we demonstrate how the few-time-steps algorithm contributes to keeping weights and biases of WDNN up-to-date with the changes of the normal behaviour. Indeed, when disabling the updating mechanism, the F1-score drops to 0.5621, while the number of false alarms increases from 226 to 684.

Hyper-parameters. Increasing the anomaly window W_{anom} determines an increase on the number of detected attacks, as shown in Fig. 8. This is mainly due to the condition $\text{MSE}_{g,i} > T_g$ in Equation 3, which yields to the detection of short attacks when W_{anom} is large enough. However, for the state-of-the-art comparison reported in Table VI, we set W_{anom}=30, as this value maximises the F1 score.

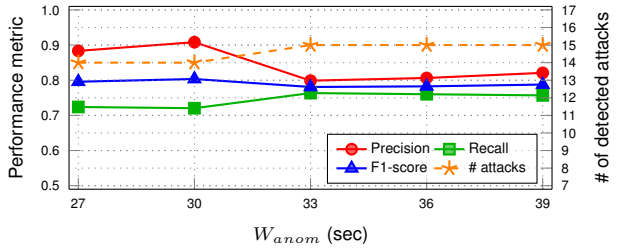

Figure 8: Sensitivity of WDNN to W_{anom} on WADI dataset.

False alarms. With the settings listed in Table IV, DAICS produces 226 false positives on the WADI test set, equivalent to 5.3 human interventions per hour on average. As for the SWaT dataset, we measured the sensitivity of DAICS to the *grace time* W_{grace}. The results, reported in Fig. 9, show that increasing W_{grace} reduces the number of interventions to 1.95 per hour on average at W_{grace}=20, although negatively impacting on precision, recall and F1 score even at W_{grace}=5.

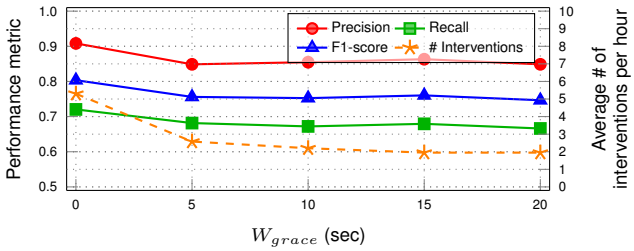

Figure 9: Sensitivity of WDNN to W_{grace} on WADI dataset.

E. Experiment 3: robustness to additive noise

In this experiment, we measure the robustness of DAICS to noise added to the sensors readings. In our experiments, we add various levels of synthetic noise to the sensor readings of both SWaT and WADI datasets. As anticipated in Section VII-B, we use white Gaussian noise with mean $\mu = 0$ and increasing standard deviation $\sigma \in \{1, 2, 3, 5, 10, 15\}$. We then observe the behaviour of DAICS with respect to the number of detected attacks and the detection accuracy measured with the F1 score metric. We then extend the experiment 2 in [19], where AADS and the CNN proposed in [20] are compared, by adding the performance of DAICS on the SWaT dataset. Note that, we have implemented the best CNN architecture as detailed in the original paper [20].

Fig. 10 reports on the performance of DAICS as a function of the Gaussian noise level. Although the trend of the point-based F1 score is similar on both datasets, we can observe a

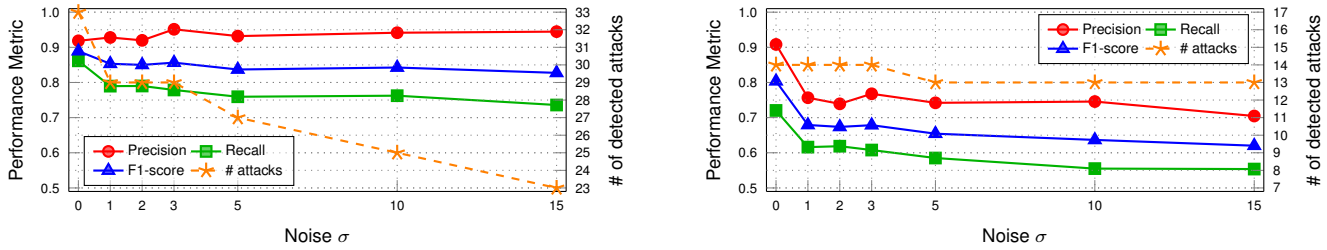

Figure 10: Performance of DAICS when adding Gaussian noise to the SWaT (left) and WADI (right) test sets.

higher impact of the synthetic noise on the detection of attacks in the SWaT dataset. We recall that an attack is determined when the conditions in Equation 3 are met. In the case of sensors, the MSE must be above the threshold for W_{anom} consecutive samples. However, the increasing number of point-based false negatives introduced with the noise, as shown in the two figures by the trend of the recall measure, reduces the chances of finding W_{anom} consecutive point-based positive samples. Of course, this affects more the detection of short attacks, which are more frequent in the SWaT dataset.

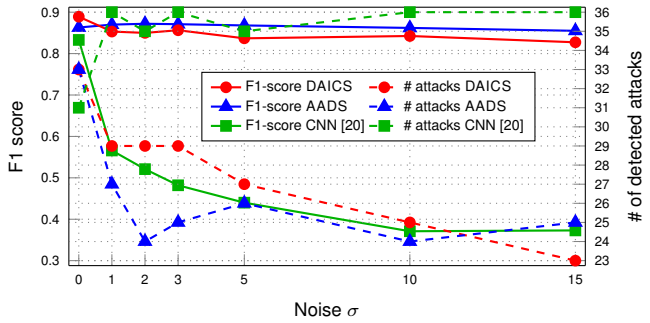

Figure 11: Performance when adding Gaussian noise to the SWaT test set.

Fig. 11 shows the values of F1-score and number of detected attacks as functions of σ. We can notice that DAICS improves AADS in the number of detected attacks at any level of noise, except for σ=15 while almost matching the F1 score (we measured a slight decrease of 0.015 on average for $\sigma > 0$). This demonstrates that the new mechanism is also solid in noisy conditions. On the contrary, the F1 score of the state of the art CNN drops drastically (green solid curve). This is mainly due to the statistical approach and to the static threshold employed to detect the anomalies, empirically selected as the value $T \in [1.8, 3]$ that maximizes the F1-score. The low values of the F1-Score for $\sigma > 0$ are due to a low precision score (around 0.31 on average), meaning that the CNN classifies most of the records as anomalies, including normal samples. This is the reason why the 36 attacks in the SWaT test set are almost all correctly classified (green dashed line). However, as everything looks like an anomaly, in a real-world deployment the output of this approach would be unusable.

In conclusion, DAICS overcomes state-of-the-art solutions on both SWaT and WADI datasets with respect to point-based F1 score and number of detected attacks. In addition, DAICS is more robust to the additive noise, hence potentially more reliable in real-world industrial scenarios where the electromagnetic noise and the natural degradation of the devices might result in noisy data.

VIII. Conclusion

In this paper, we have presented DAICS, a framework for anomaly detection in Industrial Control Systems (ICSs) based on the one-class classification paradigm and unsupervised learning. Our framework has been designed to overcome two major limitations of state-of-the-art solutions. The first, called *domain shift*, is observed when the normal behaviour of the industrial process evolves over time. If not correctly handled, this phenomenon might be source of false alarms. The second problem is caused by the presence of noisy data, either due to interference on the communication channel within the ICS or to the ageing of devices, which prevents the detection system from correctly segregating the anomalies from normal operations.

We have tackled the aforementioned problems by introducing a fast and automated mechanism for updating the ICS model based on the detected false alarms caused by changes in the normal behaviour. Such a mechanism, based on the so-called *few-time-steps algorithm*, has been designed to be usable in production environments. Indeed, the only assumption is that the technician can identify the false alarms caused by maintenance operations (hardware/software updates) or known misbehaving devices, and to signal them to DAICS. This sets DAICS apart from similar solutions, where an in-depth knowledge of the underlying algorithms is necessary to update thresholds or other parameters. The combination of the few-time-steps algorithm with a dynamic threshold mechanism, also proposed in this work, allows DAICS to overcome state-of-the-art solutions in terms of number of detected attacks and resilience to noisy data samples.

References

[1] S. Karnouskos, "Stuxnet worm impact on industrial cyber-physical system security," in *IECON 2011-37th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2011, pp. 4490–4494.

[2] NCCIC/ICS-CERT, "Cyber-attack against ukrainian critical infrastructure," 2016. [Online]. Available: https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01

[3] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, "Tabor: A graphical model-based approach for anomaly detection in industrial control systems," in *Proc. of the 2018 on Asia Conference on Computer and Communications Security*, 2018.

[4] M. Kravchik and A. Shabtai, "Efficient cyber attack detection in industrial control systems using lightweight neural networks and pca," *Preprint arXiv:1907.01216*, 2019.

[5] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *Proc. of the IEEE International Symposium on High Assurance Systems Engineering*, 2017.

[6] D. Shalyga, P. Filonov, and A. Lavrentyev, "Anomaly detection for water treatment system based on neural network with automatic architecture optimization," *arXiv preprint arXiv:1807.07282*, 2018.

[7] H. Cheng, et al., "Wide & deep learning for recommender systems," in *Proc. of the 1st workshop on deep learning for recommender systems*. ACM, 2016, pp. 7–10.

[8] Y. L. Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson, *Handwritten Digit Recognition with a Back-Propagation Network*, 1990, p. 396–404.

[9] S. Potluri, S. Ahmed, and C. Diedrich, "Convolutional neural networks for multi-class intrusion detection system," in *Proc. of International Conference on Mining Intelligence and Knowledge Exploration*, 2018.

[10] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón, and D. Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[11] M. T. Nguyen and K. Kim, "Genetic convolutional neural network for intrusion detection systems," *Future Generation Computer Systems*, vol. 113, pp. 418 – 427, 2020.

[12] S. Potluri, S. Ahmed, and C. Diedrich, *Securing Industrial Control Systems from False Data Injection Attacks with Convolutional Neural Networks*, 2020, pp. 197–222.

[13] H. Yang, L. Cheng, and M. C. Chuah, "Deep-Learning-Based Network Intrusion Detection for SCADA Systems," in *Proc. of IEEE Conference on Communications and Network Security (CNS)*, 2019.

[14] C. Chang, W. Hsu, and I. Liao, "Anomaly Detection for Industrial Control Systems Using K-Means and Convolutional Autoencoder," in *Proc. of International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, 2019.

[15] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," *Preprint arXiv:1904.04232*, 2019.

[16] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[17] iTrust, "iTrust Datasets." [Online]. Available: https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat

[18] S. Adepu and A. Mathur, "Generalized attacker and attack models for cyber physical systems," in *Proc. of 40th Annual Computer Software and Applications Conference (COMPSAC)*, 2016.

[19] M. Abdelaty, R. Doriguzzi-Corin, and D. Siracusa, "AADS: A noise-robust anomaly detection framework for industrial control systems," in *Proc. of the International Conference on Information and Communications Security (ICICS)*, 2019.

[20] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proc. of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, 2018.

[21] H. Farsi, A. Fanian, and Z. Taghiyarrenani, "A novel online state-based anomaly detection system for process control networks," *International Journal of Critical Infrastructure Protection*, vol. 27, p. 100323, 2019.

[22] M. Q. Ali, E. Al-Shaer, H. Khan, and S. A. Khayam, "Automated anomaly detector adaptation using adaptive threshold tuning," *ACM Transactions on Information and System Security*, vol. 15, no. 4, pp. 1–30, 2013.

[23] D. Park, Y. Hoshi, and C. C. Kemp, "A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1544–1551, 2018.

[24] M. Kravchik, B. Biggio, and A. Shabtai, "Poisoning attacks on cyber attack detectors for industrial control systems," *arXiv preprint arXiv:2012.15740*, 2020.

[25] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, and N. O. Tippenhauer, "Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems," in *Annual Computer Security Applications Conference*, 2020, pp. 480–495.

[26] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! attacking plcs with physical model aware rootkit." in *NDSS*, 2017.

[27] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Proc. of International Conference on Critical Information Infrastructures Security*, 2016.

[28] S. Adepu and A. Mathur, "An investigation into the response of a water treatment system to cyber attacks," in *Proc. of IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, 2016.

[29] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "Wadi: a water distribution testbed for research in the design of secure cyber physical systems," in *Proc. of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, 2017.

[30] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. The MIT Press, 2009.

[31] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[32] P. Mulinka and P. Casas, "Adaptive network security through stream machine learning," in *Proc. of ACM SIGCOMM*, 2018.

[33] P. Mulinka and P. Casas, "Stream-based machine learning for network security and anomaly detection," in *Proc. of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks*, 2018.

[34] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[35] B. I. System. Continuous Valve Actuator. [Online]. Available: https://infosys.beckhoff.com/english.php?content=../content/1033/tcbaframework/html/TcBAManager_ValveActuatorType03.htm

[36] G. C. Goodwin, D. E. Quevedo, and E. I. Silva, "Architectures and coder design for networked control systems," *Automatica*, vol. 44, no. 1, pp. 248–257, 2008.

[37] X.-S. Zhan, Z.-H. Guan, F.-S. Yuan, and X.-H. Zhang, "Performance analysis of networked control systems with snr constraint," *International Journal of Innovative Computing, Information and Control*, vol. 8, no. 12, pp. 8287–8298, 2012.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[39] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLOS ONE*, 2017.

[40] ODVA, "Technology overview series: Ethernet/ip," Tech. Rep., 2016.

[41] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Communications surveys & tutorials*, vol. 15, no. 2, pp. 860–880, 2012.

[42] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S.-K. Ng, "Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 703–716.

Maged Abdelaty received the M.Sc. in communication and electronics from the University of Alexandria, Egypt. He is currently working towards his PhD degree in information and communication technology at the Robust and Secure Distributed Computing (RiSING) research unit, Fondazione Bruno Kessler, and at the University of Trento, Italy. His research interests include anomaly detection, industrial control systems, and network security.

Roberto Doriguzzi-Corin is a researcher at Fondazione Bruno Kessler in Trento, Italy. He received the M.Sc. in Mathematics from the University of Trento and the Ph.D. in telecommunication engineering from the University of Bologna. Prior to the current role, Roberto spent more than fifteen years working as software engineer in industrial and research projects. His research interests include SDN/NFV, Machine Learning, Network Security and Linux embedded systems.

Domenico Siracusa is the head of the Robust and Secure Distributed Computing (RiSING) research unit at Fondazione Bruno Kessler (FBK). He received his Master's Degree in Telecommunication Engineering and his PhD in Information Technology both at Politecnico di Milano, respectively in 2008 and 2012. His current research interests include orchestration of next generation Internet infrastructures, cloud and fog computing, SDN/NFV and virtualization, security and robustness. Domenico authored more than 100 publications appeared in international peer reviewed journals and in major conferences on networking and cloud technologies.