Efficient Cyber Attack Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA

Moshe Kravchik and Asaf Shabtai

Abstract—Industrial control systems (ICSs) are widely used and vital to industry and society. Their failure can have severe impact on both the economy and human life. Hence, these systems have become an attractive target for physical and cyber attacks alike. In this article, we examine an attack detection method based on simple and lightweight neural networks, namely, 1D convolutional neural networks and autoencoders. We apply these networks to both the time and frequency domains of the data and discuss the pros and cons of each representation approach. The suggested method is evaluated on three popular public datasets, and detection rates matching or exceeding previously published detection results are achieved, while demonstrating a small footprint, short training and detection times, and generality. We also show the effectiveness of PCA, which, given proper data preprocessing and feature selection, can provide high attack detection rates in many settings. Finally, we study the proposed method's robustness against adversarial attacks that exploit inherent blind spots of neural networks to evade detection while achieving their intended physical effect. Our results show that the proposed method is robust to such evasion attacks: in order to evade detection, the attacker is forced to sacrifice the desired physical impact on the system.

Index Terms—Anomaly detection, industrial control systems, convolutional neural networks, autoencoders, frequency analysis, adversarial machine learning, adversarial robustness

1 Introduction

Industrial control systems (ICSs), also known as supervisory control and data acquisition (SCADA) systems, combine distributed computing with physical process monitoring and control. They are comprised of elements providing feedback from the physical world (sensors) and elements influencing it (actuators), as well as computers and controller networks, which process the feedback data and issue commands to the actuators. Many ICSs are safety-critical, and an attack interfering with their functionality can cause substantial financial and environmental harm, and endanger people.

The importance of ICSs makes them an attractive target for attacks, particularly cyber attacks. Several high impact incidents have been reported in recent years, including the attack on a power grid in Ukraine [1], the infamous Stuxnet malware targeting nuclear centrifuges in Iran [2], and attacks on a Saudi oil company [3]. In the past, ICSs ran on proprietary hardware and software in physically secure locations; more recently they have adopted remote connectivity and a common information technology (IT) stack, a trend exposing ICSs to cyber threats that leverage existing attack tools. However, the ICS defender's toolbox is limited due to the need to support legacy protocols built without modern security features, as well as the endpoints' inadequate processing capabilities. These problems can be addressed by using traditional IT network-based intrusion detection systems (IDSs) to identify malicious activity. However, this approach is ineffective, given the few known attacks on ICSs. Alternatively, model-based methods have been proposed to detect ICSs' anomalous behavior [4], [5], [6], [7]. Unfortunately, creating an accurate model of complex physical processes requires in-depth understanding of the system and its implementation, and is time-consuming and difficult to scale. Thus, recent studies have utilized machine learning (ML) to model the system. Some studies used supervised machine learning [8], [9] and achieved high precision results. With supervised learning, both benign and malicious data must be present and labeled; thus it is limited to known attacks. In contrast, unsupervised learning trains with unlabeled data, assuming that most of the data is normal. Semi-supervised learning combines unlabeled data with some supervision information [10], often in the form of labels, for a small subset of the data. In the context of this work, semi-supervised learning assumes that the training data represents normal system behavior and contains no attacks.

To overcome the limitations of supervised learning, other studies used unsupervised and semi-supervised deep neural networks (DNNs) to detect anomalies and attacks in ICS data [11], [12], [13], [14], [15], [16]. Kravchik and Shabtai [17] suggested using semi-supervised neural networks (NNs) based on 1D CNNs and demonstrated the detection of 31 of 36 cyber attacks in the SWaT dataset [18].

• *The authors are with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Be'er Sheva 8410501, Israel. E-mail: moshekr@post.bgu.ac.il, shabtaia@bgu.ac.il.*

Manuscript received 17 May 2020; revised 23 Dec. 2020; accepted 28 Dec. 2020. Date of publication 8 Jan. 2021; date of current version 9 July 2022.
(Corresponding author: Asaf Shabtai.)
Recommended for acceptance by Q. Li.
Digital Object Identifier no. 10.1109/TDSC.2021.3050101

Despite their achievements, these studies suffer from some limitations. First, they were typically verified on a single dataset, limiting the ability to address the method's generality and applicability in other settings. Second, these studies barely addressed the need to properly preprocess the input data and conduct feature selection, an important step which may have significant impact on performance. In some studies, the proposed model was applied selectively to a subset of features [11], [19]. Others applied different detection mechanisms to different features [13]. However, unified and systematic quantitative criteria for feature selection is still lacking. Third, we believe that the ability of these methods to detect many attacks is limited, because they focus on processing the time domain signals. Recently, electromagnetic side-channel monitoring studies demonstrated the effectiveness of frequency domain analysis in malicious code detection [20]. Advantages of this domain include noise removal and sensitivity to slow attacks or attacks that speed up the controlled process (as in the Stuxnet case) and are difficult to detect in the time domain. Finally, most studies using neural networks do not consider adversarial attacks on the proposed methods.

1.1 Problem Description

The threat model assumed in our research considers a powerful adversary whose goal is to change a physical-level process of the targeted ICS. The adversary considered has penetrated the system and is able to falsify the sensory data, send malicious commands to the actuators, and present a fake system state to the system's operators by manipulating network traffic between the field process and control center. An attacker that can completely fake the system state would be nearly impossible to detect, however it is unlikely that an attacker would have the ability to control the whole system (i.e., all of the sensors, actuators, PLCs). Therefore, we consider an attacker that can only falsify some of the information.

Our goal is to develop an anomaly and attack detection (AD) system that can identify the activity of such an attacker based on data collected on the physical process state of the protected ICS. We assume that the attacker is aware of the AD system's presence and will try to achieve his/her goals while evading detection.

Our research aims to address the following research questions:

1) Can we propose an effective and accurate ICS anomaly detection method based on lightweight neural networks or PCA?
2) Is the proposed method generic and effective across multiple environments and datasets?
3) What quantitative criteria should be used for anomaly detection feature selection?
4) Does detection in the frequency domain provide any benefits (i.e., a better detection rate with fewer false alarms) compared to detection in the time domain?
5) How robust are the proposed NN architectures to adversarial machine learning attacks?

1.2 Contributions

In this paper, we propose a method for detecting anomalies and cyber attacks in physical-level ICS data using 1D CNNs, shallow undercomplete autoencoders (UAEs), variational autoencoders (VAEs), and PCA. The method improves upon the method presented in [17], allowing arbitrary length sequence prediction and an arbitrary prediction horizon, adding a max-based method for threshold detection, and formalizing the detection hyperparameter criteria. In addition, we propose a feature selection approach using the Kolmogorov-Smirnov test and transform time domain signals into frequency representation using short-time Fourier transform and energy binning, and model the system in both the time and frequency domains.

The method was evaluated on three popular public datasets representing both real-world and simulated data (SWaT, BATADAL, WADI) and achieved better detection performance than previously published research in this area. In addition, we demonstrate the effectiveness of the proposed feature selection method and its generalizability.

Finally, the proposed method's robustness to adversarial evasion attacks under a threat model of a white-box attacker that has gained control of the sensor data was evaluated. The results show the method's resilience: to evade detection, the attacker must abandon his/her goal of physically impacting the system.

The main contributions of this paper are as follows.

- An effective and generic method for detecting anomalies and cyber attacks in ICS data using 1D CNNs, shallow UAEs, VAEs, and PCA.
- A method for robust feature selection based on the Kolmogorov-Smirnov test and experimental validation of its effectiveness.
- The efficient and novel application of the abovementioned detection method to the frequency domain.
- An algorithm for conducting adversarial evasion attacks on the proposed detection method and a demonstration of the robustness of the proposed models to such attacks (to the best of our knowledge, this is the first research to cover adversarial attacks on 1D CNNs and AEs, and to examine their robustness under this threat model).

2 Background

2.1 Industrial Control Systems

A typical ICS combines network-connected computers with physical processes, which are both controlled by these computers and provide the computers with feedback. The key components of an ICS include sensors and actuators that are connected to a local computing element, commonly called a programmable logic controller (PLC).

Sensors and actuators are usually connected to the PLC with a direct cable connection, and commands are sent to the PLC via a local networking protocol, such as CAN, Modbus, or S7. Remote nodes' PLCs are managed from the control segment of the network and are connected to it via protocols, such as TCP, over a wireless, cellular, or wired network. A central element of the control segment is an engineering workstation running SCADA software that provides a means to both monitor the PLCs and change their internal logic. A human machine interface (HMI) computer is another important element of the control segment,

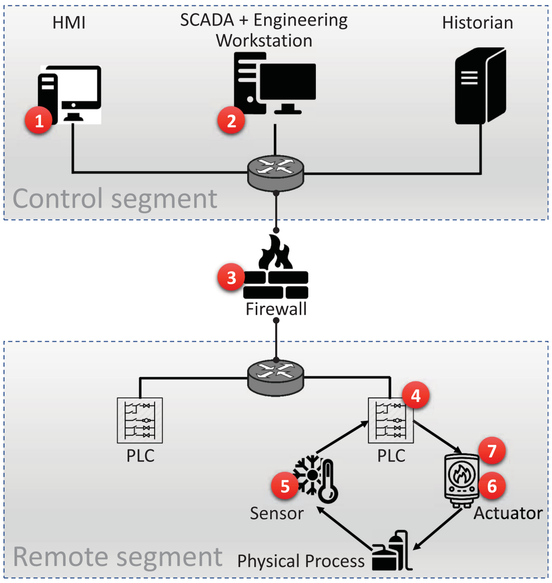

Fig. 1. A schematic of ICS architecture and possible attack vectors.

providing the operator with the current state of the controlled process. A historian server that collects and stores data from the PLCs also resides in the control segment.

2.2 Attacks on ICSs and Threat Model

The central role of ICSs in critical infrastructure, medical devices, transport, and other areas of society makes them an attractive target for attacks. Motives for such attacks are diverse and include criminals seeking control of an important asset, industrial espionage and sabotage, political reconnaissance, cyber warfare, and privacy evasion.

Several attack vectors, or a combination of them, can be used to attack an ICS (see Fig. 1).For example, an adversary can attack:

1) the HMI machine, by exploiting software vulnerabilities in its OS and application stack, presenting a fake view of the process and causing the operator to issue erroneous commands [21];
2) the SCADA and/or engineering workstation machine, by exploiting software vulnerabilities and obtaining full control of the ICS, as occurred in the attack in Ukraine [1];
3) the communication network in the control segment, the remote segment, or between them, by performing eavesdropping, replay, or false packet injection attacks;
4) the PLC, by exploiting software vulnerabilities or trust between the PLC and SCADA - this allows the attacker to change the PLC's logic, influencing the controlled process and causing damage, as in the Stuxnet case;
5) the sensors, by leveraging physical effects, interfering with the measurements, or replacing the sensor with a malicious one, as shown in [22];
6) the actuators, by altering the signal sent by the actuators to the controlled process, as described in [23]; or,
7) the actuators and communication channels to create a covert channel, as demonstrated in [24].

The *threat model* assumed in this research considers an adversary whose ultimate goal, regardless of the attack vector, is a physical-level process change. The adversary considered has penetrated the system and is able to falsify the sensor data, send malicious commands to the actuators, and present a fake system state to its operators by manipulating network traffic between the field process and control center. We assume that the attacker can falsify *only some* of the information.

Therefore, we apply a physics-based attack detection [23] approach. The main idea behind this approach is modeling the physical state of the protected system. Using NNs for modeling does not require defining the specific equations and parameters describing the system modeled, as the NN learns the best system representation itself. Monitoring the physical system state and deviations from the model facilitates the detection of anomalous behavior, including deviations caused by spoofed sensor readings and injected control commands.

Despite the fact that our anomaly detection domain is purely physical, we argue that our method goes *beyond simple anomaly detection*. As we show, it can detect sophisticated multi-point cyber attacks that combine data tampering, malicious commands, and replay attacks. These attacks are targeted, evasive, and performed by means of the cyber domain. Therefore, we classify the method as *cyber attack detection*.

2.3 Time - Frequency Domain Transformation

Raw data measured by ICS sensors produces a time series. While in most ICS anomaly and attack detection research this data is processed directly, it is very common in signal processing to analyze data in the frequency domain. Using the Fourier transform (Equation (1)) we can build a signal's frequency domain representation:

$$\hat{f}(k) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi i x k} dx, \tag{1}$$

where f is some function depending on time x, $\hat{f}$ is its Fourier transform, and k is the frequency. When dealing with periodic data samples, rather than a continuous function, the discrete Fourier transform is used:

$$F_k = \sum_{n=0}^{N-1} f_n e^{-2\pi i n k / N}, \tag{2}$$

where f_n denotes the $n\mathrm{th}$ sample of f.

Fourier transform of a time series provides its spectrum over the entire period of time measured. To detect changes in the signal spectrum over time, we use the short-time Fourier transform (STFT) which applies the Fourier transform to short overlapping segments of the time series.

Frequency domain analysis has several advantages: First, it provides a more compact representation of the dominant signal components. Second, it allows for the detection of attacks in which the frequency of regular operation modes is changed, e.g., quickly starting and stopping the engine. Lastly, according to the uncertainty principle [25], functions localized in the time domain (e.g., represented by a short spike) are spread across many frequencies, and functions that are concentrated in the frequency domain are spread across the time domain. This means that slow attacks that

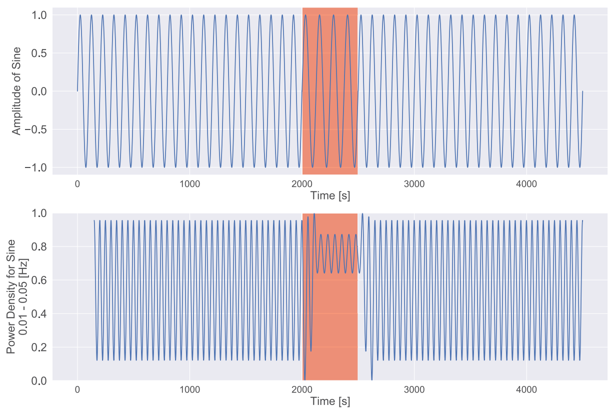

Fig. 2. Frequency domain transformation benefits for anomaly detection. The highlighted part of the top signal has a frequency that is 25 percent smaller than the rest of the signal. This difference is practically indistinguishable in the time domain but clearly stands out in the frequency domain (the bottom part).

usually evade time domain detection methods will stand out in frequency analysis, but quick attacks will be difficult to detect in the frequency domain.

As an illustration, consider a sensor reporting a regular sine signal and an adversary that decreases the signal frequency by 25 percent for some period of time to use it as a covert channel. This change will barely be noticeable in the time domain but will be very distinct in the frequency domain, as shown in Fig. 2.

2.4 Adversarial Attacks on Machine Learning Models

Adversarial data consists of specially crafted input samples which cause the algorithm to produce incorrect results at test time. The field of adversarial learning in DNNs has attracted interest, since [26] showed that NN-based classifiers can be tricked into mislabeling an image by changing a small number of pixels in a way that is imperceptible to the human eye. Since then, adversarial attacks on NNs have been demonstrated in malware detection, speech classification, etc.

Adversarial machine learning attacks can be divided into *poisoning* attacks performed at training time and *evasion* attacks performed at test time. To model adversarial attacks, we need to consider the attacker's goals and knowledge. In [27], the goals are subdivided into the desired violation (integrity, availability, or privacy) and specificity (targeting a set of inputs or all of them, as well as producing specific incorrect output or just any incorrect output). In the context of anomaly detection, the attacker's goal might be to cause the system to classify an anomaly as benign (a specific integrity attack) or to classify many benign samples as anomalous in order to affect the ability to trust the detection results (an indiscriminate availability attack).

Biggio *et al.* [27] defined the attacker's knowledge in terms of the training data $\mathcal{D}$, the feature set $\mathcal{X}$, the algorithm f and its objective function $\mathcal{L}$, and the training hyperparameters and the detection parameters w learned. In our study's context, $\mathcal{X}$ is the set of sensor and actuator states used to train the model, while f and $\mathcal{L}$ represent the selected NN architecture and its loss function. Thus, the attacker's knowledge is represented by components $(\mathcal{D}, \mathcal{X}, f, w)$. The worst-case perfect knowledge white-box scenario happens when all four components are known. Gray-box attacks occur if at least one of the components is unknown and cannot be reproduced; for example, the attacker might know the feature set and the NN type, but not the network parameters and weights. Black-box attacks are characterized by the lack of specific knowledge about any of the four components. We will review the relevant research on adversarial attacks in the ICS anomaly detection context in the next section.

3 Related Work

The area of anomaly and intrusion detection in ICSs has been widely studied. Extensive surveys [28], [29], [30], [31] and surveys of surveys [32] are devoted to classifying research in this field. Our review of related work focuses on ICS anomaly and cyber attack detection using *the physical state of the system*, as measured by the sensors. As noted in [23], the first step in physics-based detection is system state prediction. Some studies used autoregressive models [33], [34] or linear dynamical system modeling [35], [36], [37] for this purpose. Unfortunately, both approaches' assumptions include linearity of the modeled system which is not typically fulfilled in ICSs.

Specification-based system modeling [7], [38] and invariant-based mechanisms [39], [40] were also shown to be effective. One of their main drawbacks is specificity, i.e., the solution should be tailored to the system and its operating conditions or known invariants.

Recently, ML-based modeling became popular in ICS research. PASAD [19] is based on ideas from singular spectrum analysis and detects attacks in the signal subspace representing the deterministic part of the time series. The principle difference between our approach and PASAD is our approach's ability to detect anomalies based on the correlation of multiple input features, while PASAD deals with a single feature.

In a recent competition on water distribution system cyber attack detection (the BATADAL [41]), seven teams demonstrated their solutions on a simulated dataset. The best results were shown by the authors of [42], who modeled the system precisely using MATLAB. The main limitation of this solution is its reliance on the ability to create a precise system model, a non-generic and difficult task. Another work that achieved a high score in the competition is [14], in which the authors proposed a three-layer method: the first layer detects statistical anomalies, the second layer is an NN aimed at finding contextual inconsistencies with normal operation, and the third layer uses principal component analysis (PCA) on all sensor data to classify the samples as normal or abnormal. Our work differs from [14] in the following ways. First, we study the efficiency of a single generic mechanism, as opposed to the multistage system used by [14]. Second, our solution evaluates types of NNs not covered by [14]. Lastly, we study feature selection criteria, frequency domain anomaly detection, and adversarial robustness.

Another NN-based study from the BATADAL competition is [15], which used variational autoencoders (VAEs) to calculate the data's reconstruction probability. In our research, we found that VAEs are not very accurate in reconstructing time series data. Therefore, we suggest using simpler autoencoder models and demonstrate their effectiveness at this task.

NNs have been used in other physics-based cyber attack detection research ([11], [12], [13], [43]). Unlike our work, these studies use more complex recurrent neural networks (RNNs) and graphical models and do not study the frequency domain.

Autoencoders (AEs) have been used for anomaly and intrusion detection before [44], [45]. The differences between our work and [44] are that in our research (1) AEs are applied to raw physical signals without statistical feature extraction, and (2) AEs are applied to the frequency domain. We extend the research in [45] by applying AEs to cyber attack detection in time series, combining control, status, and raw physical data, as well as by applying AEs to the frequency domain. We also enhance the network's architecture and present a feature selection method which improves network performance.

A study by Taormina *et al.* [16] applied AEs to the BATADAL dataset, a method which was able to detect all of the attacks and achieve an $F1$ score of 0.886. Our research differs from that study in the following ways: First, we study both 1D CNNs and AEs on three different datasets, two of which come from real-world testbeds. Second, we use a different AE architecture that is adapted to multivariate time series prediction, uses noise and an inflation layer, and achieves a higher $F1$ score. Third, we study frequency domain detection. Finally, we present adversarial attacks on the proposed network and demonstrate the network's robustness.

Little attention has been given to adversarial attacks in the ICS context, and there are a number of differences between our study and the work in the area where most adversarial research has been done (image and sound processing):

- most existing work is focused on supervised learning problems, while our research deals with semi-supervised learning problems,
- most existing work is focused on classification tasks, while we deal with prediction (regression) tasks,
- while in tasks, such as image classification, the output variable (image class) is not part of the input, in our task the input and output features are the same, and
- in our case, there are multiple constraints on the internal structure of the data, due to the laws of physics and PLC logic, which are not present in other domains.

A successful evasion attack framework on machine learning anomaly detection was demonstrated in [46]. The authors brought a monitored simulated Tennessee Eastman (TE) process to a dangerous pressure level by manipulating sensor measurements in a way that was classified by both a linear regression and feedforward NN as normal. This research differs from [46] in the types of NN detectors attacked, the system modeling approach, and the adversarial attack strategy - we use state-of-the-art gradient-based adversarial optimization, while the authors of [46] used iterative linear approximation of the detector.

TABLE 1
Comparison of Studies on Physics-Based, Machine Learning Anomaly Detection for ICSs

Ref.	Detection method	Dataset	Frequency domain	Feature selection	Adversarial robustness
[11]	RNN	SWaT (subset)	No	No	No
[12]	DNN, SVM	SWaT	No	No	No
[14]	MLP, PCA	BATADAL	No	No	No
[15]	VAE	BATADAL	No	No	No
[13]	Bayesian Network	SWaT	No	Some	No
[19]	Singular Spectrum Analysis	SWaT (subset), proprietary	No	No	No
[17]	1D CNN	SWaT	No	No	No
[16]	AE	BATADAL	No	No	No
[47]	LSTM	Gas pipeline [49]	No	No	Yes
[46]	MLP	Simulated TE	No	No	Yes
[48]	AE	BATADAL	No	No	Yes
[43]	GAN	SWaT, WADI	No	No	No
Our research	1D CNN, UAE, VAE, PCA	SWaT, BATADAL, WADI	Yes	Yes	Yes

In [47], the authors used a generative adversarial network (GAN) to create stealthy attacks on an ICS, evading a baseline LSTM anomaly detector. In our research, we target different types of NN detectors, and use a gradient-based optimization for the attack generation, unlike [47] that uses GANs, which are complex and hard to train.

Most recently, Erba *et al.* [48] showed how to create successful evasion attacks against an autoencoder-based detection mechanism. The main difference between [48] and our approach is the threat model chosen. The authors of [48] consider a very powerful attacker that can both generate arbitrary malicious inputs to the PLC and create fake traffic that is fed to the detector. We consider a more constrained attacker that can only control the sensor data that is seen by the PLC and the detector. We argue that our threat model represents a more realistic scenario, as elaborated upon in Section 4.7. Table 1 summarizes related studies and compares them to our research.

4 Methodology

4.1 Formal Definitions

Consider an ICS containing N measured elements (i.e., sensors and actuators) $s_1, s_2, \ldots, s_N$. The sensor measurements and the actuator states are represented by N-dimensional vectors $y_i \in \mathcal{R}^N$, where y_i contains the values of the features at time i and y_i^s is the value of the sensor (or actuator) s at time i. The physical state of the ICS at a given time y_i is a function $f \in \mathcal{F}$ of its past states $y_{i-1}, y_{i-2}, \ldots$. The machine learning model approximates this function by producing estimates

$$\hat{y}_i = \hat{f}_N(y_{i-1-l}, \ldots, y_{i-1}). \tag{3}$$

Please note that the single vector prediction presented in Equation 3 can be further generalized to a multi-vector prediction (as described in Section 4.5).

TABLE 2
List of Symbols

Symbol	Description
y_i^s	Observed feature (sensor) s value at time i
y_i	Observed feature vector at time i
$\hat{y}_i$	Predicted feature vector at time i
$\hat{r}_i$	Feature residual vector at time i
$\mathbf{w}$	NN model weights
τ	Anomaly detection threshold
w	Anomaly detection window
$\mathcal{A}_i$	Anomaly detection indicator at time i
$\mathcal{D}$	Dataset

Let $L(\mathcal{D}, \hat{f})$ be a *loss function* quantifying the approximation error over all instances of the dataset $\mathcal{D}$. In this paper, the mean squared error (MSE) loss was used: $L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. In the training phase, we search for the largest subset M of all N features that minimizes the value of the loss function L for the validation dataset $\mathcal{D}_{val}$ (for the approximation function trained on a dataset $\mathcal{D}_{train}$). In this paper, our suggested approach for feature selection is based on the Kolmogorov-Smirnov test (described in Section 4.3).

In the testing phase, deviations between the predicted and observed values of the measurements are used by an anomaly detection function $\mathcal{A}(y_i, \hat{y}_i, \theta)$ which compares these deviations to some detection hyperparameters θ and returns one if an anomaly is detected at time i and zero otherwise. In our method, two detection hyperparameters are used: a threshold τ and a window w; in Section 4.5 we provide the precise definition of function $\mathcal{A}$ (presented by Equation (11)).

Let's consider an adversary whose goal is to compromise a measured value of a sensor s at time i and set it to a value within some range that causes the attacker's desired effect. For example, setting the water tank level to above 700 mm causes the system to turn the water pump on, potentially causing an underflow that could empty the tank. The attacker's goal is to introduce the change without triggering anomaly detection alerts, and this might require the attacker to change additional sensors' values at time i or other times before or after the attack. Formally, let $\mathcal{D}_{test}$ denote the test set, $y_i^s \in \mathcal{D}_{test}$ be the value of the sensor measurement the attacker wants to change, and δ be the total perturbation added by the attacker. The attacker's goal is to find a minimal perturbation that will both bring y_i^s into the desired value range and not result in any alerts by function $\mathcal{A}$ (see Equation (4)). In addition, the perturbed dataset must conform to the features' range constraints, which we omit from the following equation for brevity.

$$\min_{\delta} \mathcal{A}(\mathcal{D}_{test} + \delta, \hat{\mathcal{D}}_{test} + \delta, \theta) = 0, \quad s.t.\ s_{al} \le y_i^s \le s_{au}, \tag{4}$$

where s_{al} and s_{au} are the lower and upper limits of the desired attack range. For easy reference, we provide a list of the symbols used in this section in Table 2.

4.2 Method Overview

In this section, we provide an overview of the proposed method. The method's steps are presented in Fig. 3. The input data comprises time series of the monitored system's sensor and actuator values. The data is first normalized (not shown in Fig. 3). In the training phase, the normalized training data passes to the sensor (feature) selection step (step (1)). For each feature we calculate the revised Kolmogorov-Smirnov statistic (see Section 4.3) to select the features with a consistent probability distribution. The selected features will be modeled by the learning algorithm. Feature selection should be performed prior to training the model by using the training and

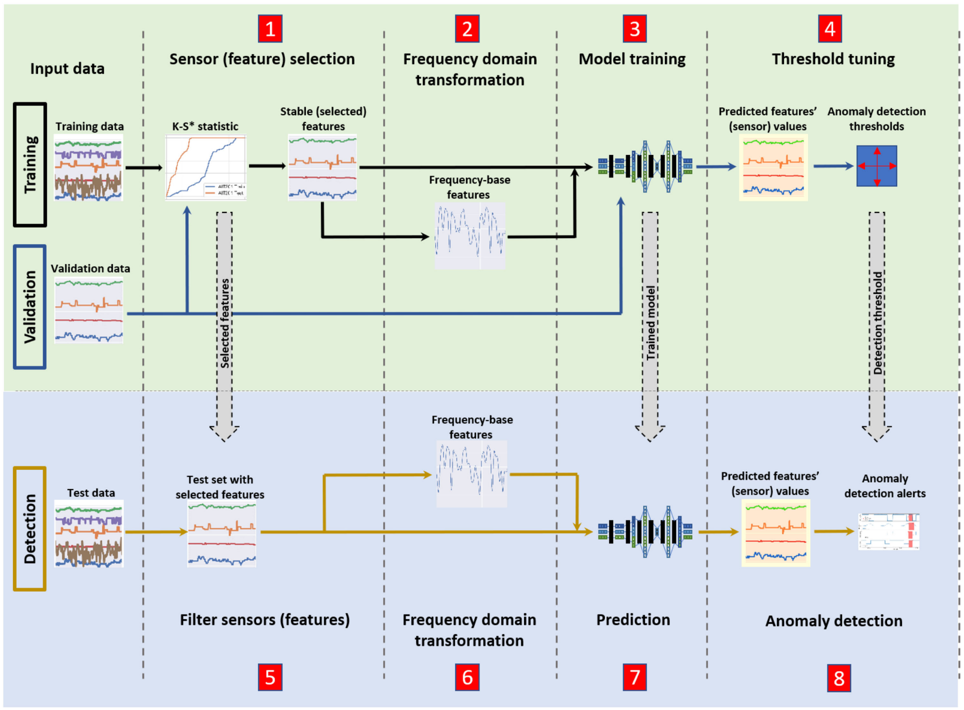

Fig. 3. An overview of the proposed method.

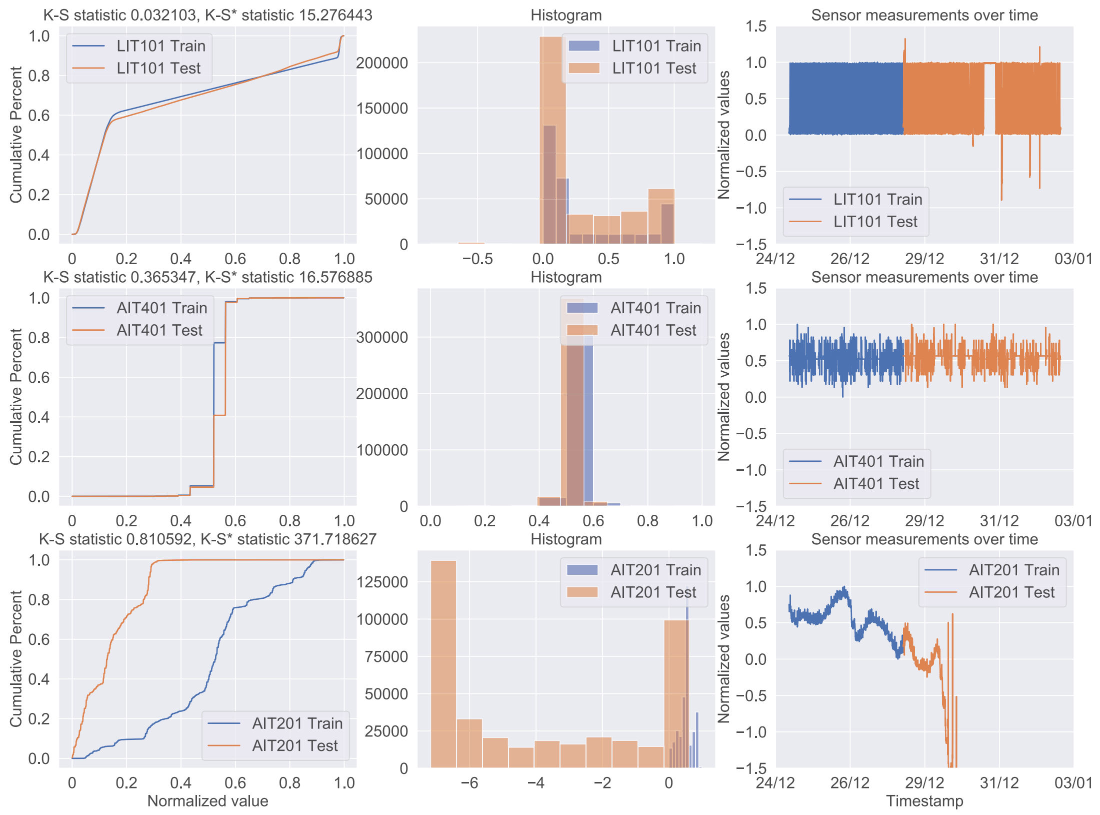

Fig. 4. Feature statistic comparison. LIT101 (first row) has a very similar distribution in both the training and test data. (second row) has a similar but slightly offset distribution. Its K-S has a high value, but its K-S* value correctly classifies the distributions as close. AIT201 (third row) has very different distributions.

validation sets, and comparing the statistics of the two sets. To maintain the detection accuracy, periodic validation of the features' consistency can be performed during the detection phase, and the model can be retrained if significant changes are detected. In step (2) the data of the selected features can be transformed to the frequency domain, as described in Section 4.4 (note that this is an optional step). Then, in step (3) the learning algorithm uses the selected features' data, either in the time (original values) or frequency domain (when step (2) is performed), to train a model that represents the features. The learning algorithms used in this study are discussed in Sections 4.5.1 and 4.5.2.

The trained models are tested on the validation data, and the anomaly detection hyperparameters (the detection threshold and window) are determined, as described in Section 4.6. During the detection (testing) phase, the normalized test data is first filtered (step (5)) according to features selected in the feature selection phase. In cases in which the model was trained on frequency-based features, the normalized, filtered test data is transformed to the frequency domain (step (6)). Then, the time or frequency domain representations of the normalized selected features are passed to the trained model (step (7)), and the attacks are detected based on the difference between the predicted and observed values of the features and the pre-defined threshold and window (step (8)). The steps are described in the subsections that follow.

4.3 Feature Selection Using K-S Test

Our detection mechanism is based on the ability to model and predict the system's behavior. This can be accomplished if the data meets the following requirement: the training data must be representative of the test data; more specifically, it should contain all the (latent) states and the transitions between the states which appear in the test data. Therefore, our anomaly detection method is based on the assumption (common in anomaly detection [50]) that the normal test data has the same probability distribution as the training data (only the anomalies have a different distribution). However, we found that several SWaT features do not have the same distribution in the training and test data (see Fig. 4), which might be a result of a concept drift. To find stable features, we use the Kolmogorov-Smirnov test (K-S test) [51] as a quantitative measure of the similarity between the probability distributions of the training and test data. We chose the K-S test, because it is non-parametric and isn't based on any assumptions on the probability distributions tested. It is also more sensitive than comparing the mean and standard deviation or the using the t-test, both of which do not work well with multimodal and non-normal distributions. While the goal of the feature selection step is to determine the features that are stable across the training and the test sets. However, in many machine learning tasks the real test set is assumed to be unknown. For such settings, feature selection can be performed using the validation set, where the validation set approximates the test set. We present the comparative evaluation results for the validation and tests sets using the K-S test, as well as an alternative feature selection method in Section 5.3. The K-S test statistic for two distributions is the maximal difference between their empirical cumulative distribution functions (ECDFs):

$$K - S = \sup_x |F_1(x) - F_2(x)|, \tag{5}$$

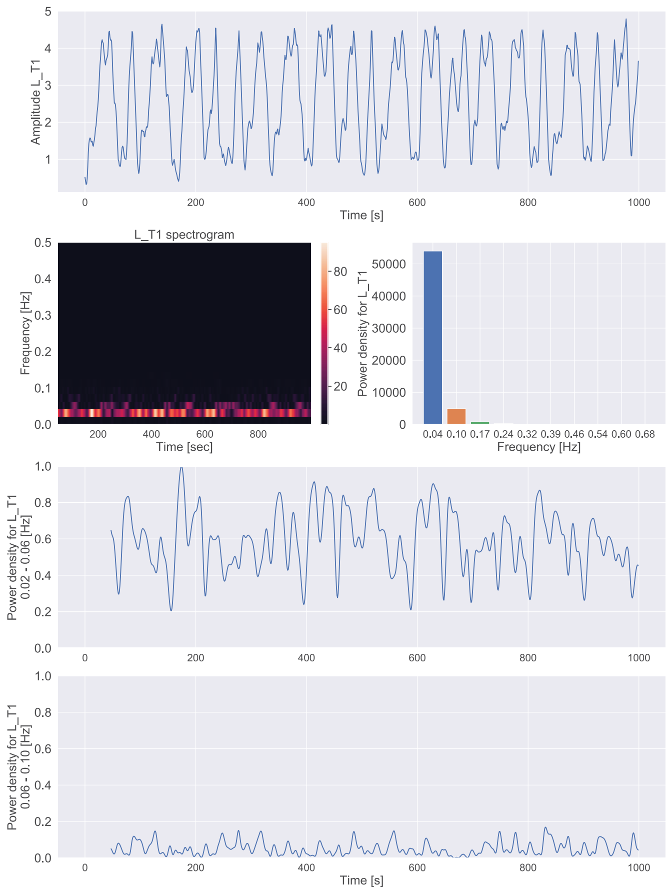

Fig. 5. Frequency transformation of the L_T1 feature. The first row depicts the raw signal. The second row represents the STFT spectrogram (left) and the power density distribution in the spectrum bands (right). The last two rows are the frequency domain features; most of power density is in the first two bands.

where F_1 and F_2 are ECDFs of the compared distributions. They can be calculated as:

$$F_x = \frac{1}{n}\sum_{i=1}^{n} I_{[-\infty,x]}(X_i), \tag{6}$$

where

$$I_{[-\infty,x]} = \begin{cases} 1, & \text{if } X_i < x \\ 0, & \text{otherwise.} \end{cases}$$

The original K-S test is limited to fully specified distributions [52], however we modified it slightly to obtain a concise metric for filtering out features unsuitable for modeling. Using the maximum as a statistic makes the K-S test extremely sensitive to small CDF differences when the distribution's mean is slightly offset on the x-axis. To increase the test's robustness, we used the area between the CDFs instead, which is calculated as:

$$K - S^* = \int_x |F_1(x) - F_2(x)| dx. \tag{7}$$

Fig. 4 illustrates three SWaT features, their values over time, histograms, and K-S and K-S* statistics, providing an example of the the K-S* statistic's advantage over the original K-S statistic. LIT101's (first row) training and test data's distributions are very close, and those of AIT401 (second row) are close as well, as both of the CDF graphs (left column) and histograms (middle column) show. However, the K-S statistic for AIT401 is about 10 times larger than that of LIT101. Using the area between the two CDF curves (K-S*) instead of the maximal distance between them (K-S) allows us to conclude that the feature is stable and leave it in the experiment.

4.4 Frequency Domain Transformation

In our experiments (see Section 5) we explore and evaluate the usefulness of the frequency domain features in detecting attacks. Therefore, in addition to testing the time domain features (i.e., the sensors' values over time), in this step, we transform the signals from the time domain to the frequency domain. The following method was used to create signal representation in the frequency domain (outlined in Algorithm 1).

1) Determine the dominant frequency of each signal (the frequency with the most energy) using the discrete fast Fourier transform (DFFT) (*frequencyAnalysis* function, described in lines 1-13 and called in line 18).
2) Determine the window for the short-time Fourier transform (STFT) based on the dominant frequency period. It was found that the optimal window is between one and two periods of the dominant frequency (line 19).
3) Transform the signals into their frequency representation.
   a) Split the entire signal into overlapping windows.
   b) Perform STFT for each window (these two items are presented by the call of the *spectrogram* function in line 20).
   c) Binarize the entire STFT spectrum into a number of bins. Calculate the total energy value of the signal in each bin (lines 21-23).
   d) Select a small number of bins with the most energy (line 24). The energy values will represent the feature in the frequency domain for the corresponding time window. We found that two or three bins were sufficient for representing the features. It is also possible to calculate the number of bins based on the percentage of the total energy value they contain (e.g., at least 90 percent).

This process is also illustrated in Fig. 5 which presents the original signal of the L_T1 sensor from the BATADAL dataset, its spectrogram, and two features representing the power density of the L_T1 readings over time in the two frequency bands with the most energy.

4.5 Data (System) Modeling

The following four approaches were used to model the sensor data by training the models either on the time or frequency domain features:

1) 1D CNN - using the architecture described in [17];
2) UAE - using a modified version of an undercomplete autoencoder which is described later in this section (Section 4.5.1);

3) VAE - using a standard variational autoencoder (VAE) architecture (see Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2021.3050101), with the addition of the inflation layer used in the UAE approach; and,
4) PCA - using two modes of the principle component analysis (PCA) algorithm which is described later in this section (Section 4.5.2).

Algorithm 1. Transform Signal s Into the Frequency Domain Representation

```
1:  function FREQUENCYANALYSIS($s, sp$)
2:      ▷ Find the dominant frequency of the signal and its period given the signal $s$ and the sampling period $sp$
3:      $n \leftarrow len(s)$
4:      $freq \leftarrow DFFT(s)$ ▷ The discrete FFT of the signal
5:      $mag \leftarrow abs(freq)[: n/2] * 1/n$ ▷ The magnitudes of the real part of the FFT
6:      $freq\_mag \leftarrow listOf(freq, mag)$
7:      $sorted\_freq \leftarrow decSort(freq\_mag)$
8:      if $sorted\_freq[0][0]$ then ▷ Ignore the constant component if it has the most energy
9:          $ff \leftarrow sorted\_freq[0][0]$ ▷ Fundamental frequency
10:     else
11:         $ff \leftarrow sorted\_freq[1][0]$
12:     $period \leftarrow (1.0/ff)/sp$
13:     return $(ff, period)$
14:
15: function FREQTRANSFORM($s, sp, ratio, b\_num$)
16:     ▷ Represent the signal $s$ as energy in the $b\_num$ most dominant frequency bands given the sampling period $sp$ and STFT window to the dominant signal period $ratio$
17:     $all\_freq\_bins \leftarrow 10$
18:     $(f\_freq, period) \leftarrow frequencyAnalysis(s, sp)$
19:     $STFT\_wind \leftarrow period * ratio$
20:     $freqs, Sx \leftarrow spectrogram(s, rate, STFT\_wind)$
21:     $bands \leftarrow linspace(0, len(freqs), all\_freq\_bins + 1)$
22:     for $i = 0$ to $all\_freq\_bins$ do
23:     $bands\_energy_i \leftarrow \sum_{i=bands_i}^{bands_{i+1}} Sx_i$
24:         $dom\_bands \leftarrow decSort(bands\_energy)[: b\_num]$
25:     return $dom\_bands$
```

4.5.1 Undercomplete Autoencoders

After experimenting with multiple AE architectures, including LSTM-based AEs, variational AEs, and denoising AEs, we discovered that the best detection performance is achieved with the simplest undercomplete AEs. We describe the selected UAE architecture here, and a comparison of the results in provided in Section 5.4. The best results were achieved using the UAE network variant adapted for multivariate sequence reconstruction.

The network of the modified UAE contains the following:

- an optional corruption layer applying Gaussian noise to the input sequence,
- a fully connected layer with a ReLU or tanh activation function *inflating* the input; the purpose of this layer is to enlarge the hypothesis space,

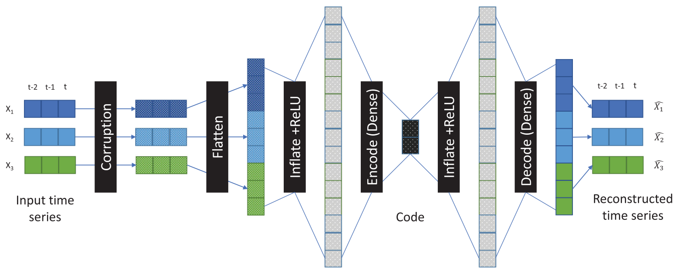

Fig. 6. The architecture of the undercomplete autoencoder used in this research.

- an encoding layer that flattens the input and produces its compact representation using a fraction of the input size; in our experiments the best results were achieved using a compact representation that is 50 percent the size of the input,
- a decoding layer that reconstructs the original sequence from its compact representation.

This architecture can deal with sequences of arbitrary length and is presented in Fig. 6.

4.5.2 Principal Component Analysis

Principal component analysis (PCA) transforms a set of variables into a set of values of linearly uncorrelated principal components (PCs), orthogonal to each other. We used PCA as a baseline to compare the performance of the neural networks to. The use of PCA for anomaly detection in ICSs is not new; it was suggested as one of the detection layers in [14], and [45] used it for detecting anomalies in a simulated Lorenz system and telemetry data. In [14], detection was conducted in the *PC subspace*. Anomalies found in this subspace do not have a direct physical meaning in the original data feature space and therefore are difficult to interpret and explain.

In our research, we used the approach outlined in Algorithm 2.

Algorithm 2. Use PCA to Predict x_{test} Given Training Data x_{train}

```
1: function PCAANALYSIS($x_{train}, x_{test}, components$)
2:     $pcaModel \leftarrow PCA(components)$
3:     $pcaModel.fit(x_{train})$
4:     $x_{test}PCA \leftarrow pcaModel.transform(x_{test})$
5:     $\hat{x}_{test} \leftarrow pcaModel.inverse\_transform(x_{test}PCA))$
6:     return $\hat{x}_{test}$
```

Our analysis restores the prediction to the original feature space and therefore, unlike previously presented methods that used PCA for anomaly detection, our method allows for the natural application of the detection method we use for the neural networks (described below). To distinguish this detection method from detection in the PC subspace, we refer to it as *PCA-Reconstruction* in Section 5. We implemented an extension to the classic PCA analysis. PCA usually operates on single time step vectors. This allows for the detection of context inconsistencies between multiple features but is less powerful in the detection of time-related inconsistencies in a single feature. To compensate for this deficiency, we implemented a *windowed-PCA* algorithm (similar to [53]), which divides the data into time windows of a given width, performs the analysis depicted in Algorithm 2 on vectors containing multiple data points, and then restores the window predictions into the original signal shape. Two variants of this *windowed-PCA* algorithm were implemented: with overlapping windows and with non-overlapping windows. The standard PCA algorithm implementation from the Python scikit-learn package with the number of components equal to half of the modeled features was used as the basis for our experiments.

4.6 Threshold Tuning, Anomaly Detection, and Scoring Method

The anomaly detection method used in this research is based on the one used in [17], however in the current study we extend it in a number of ways which are elaborated upon below.

An NN or PCA is used to predict the future values of the data features based on previous values (either in the time or frequency domains). Thus, the network performs the function

$$(\hat{y}_{h+n}, \hat{y}_{h+n+1}, \ldots, \hat{y}_{h+n+m}) = f(y_{n-1-l}, \ldots, y_{n-1}), \tag{8}$$

where l and m represent the input and output sequence length respectively and h is the prediction horizon. We generalized the method described in [17] to allow the prediction of arbitrary length sequences in the future with a specified horizon, e.g., predicting 256 time steps starting with the fifth time step from the last input time. Next, the residual vectors are calculated as:

$$\vec{r_t} = \left|\vec{y_t} - \vec{\hat{y}_t}\right|. \tag{9}$$

The residuals are used to trigger anomaly alerts in one of two ways: max-based and std-based scoring.

Max-Based Scoring. In this approach, the residuals are normalized by dividing them by the maximal per feature residuals for the validation set data, and the maximum of the normalized residuals is compared to a threshold τ:

$$\vec{R_t} = \frac{\vec{r_t}}{\max \vec{r}} \tag{10}$$

In order to prevent false alarms on short-term deviations, we require that the residual exceed the threshold for at least a specified duration of time window w. Thus, an anomaly alert $\mathcal{A}_{\rangle}$ at time i is determined by:

$$\mathcal{A}_{\rangle} = \prod_{t=i-w}^{i} \max \vec{R_t} \, > \, \tau. \tag{11}$$

The hyperparameters τ and w are determined by setting a maximal accepted false alarm rate for the validation data and finding the solution to:

$$(\tau, w) = \underset{\tau,w}{\arg\min}(\tau_n + w_n)\{(\tau, w) \,|\, |\mathcal{A}(\tau, w)| \leq FP_{max}\}, \tag{12}$$

where τ_n and w_n are respectively the threshold and the window values normalized to the (0, 1) range, $\mathcal{A}(\tau, w)$ is the set of anomalies detected with the specific threshold and window values, and FP_{max} is the maximal allowed number of false alarms in the validation data. In other words, we are looking for the hyperparameter values that do not produce more than the permitted number of false alerts, while minimizing the sum of their normalized values. Using normalized values allows us to optimize both hyperparameters with equal weight (regardless of their absolute values). Minimizing the sum of normalized values enables us to strike a balance between the detection sensitivity controlled by the threshold and the detection latency controlled by the window size.

Std-based Scoring. This approach for detecting the attacks differs from the max-based scoring approach by normalizing the residuals using their mean and standard deviation (for each feature individually) and is described in [17].

Please note that with the frequency domain detection, transforming between domains spreads the energy across the STFT time window thus pushing the attack's start forward and pushing its end back. To compensate for this effect, we subtract the duration equal to one third of the STFT window from both ends of the detected attack period.

To summarize, our extensions to the anomaly detection method used in [17] are:

- generalization of the prediction, allowing arbitrary length sequence prediction and an arbitrary prediction horizon,
- addition of a max-based method for determining the desired threshold, and
- formalization of the hyperparameter selection criteria.

4.7 Adversarial Threat Model and Robustness Analysis

In addition to evaluating the performance (in terms of the detection rate) of the proposed method (described in Fig. 3), we are also interested in evaluating the robustness of the proposed method to adversarial evasion attacks. Therefore, in this section we describe the implementation of an adversarial attack on the trained models which is used to evaluate the method's robustness.

4.7.1 Threat Model

In this research, we assumed the worst-case scenario - a white-box attacker that knows everything about the model used for detection, including , for example, the weights of the NN. We consider an attacker that is trying to perform a specific integrity attack, namely, to cause a physical-level change in the system's behavior while remaining undetected by the monitoring anomaly detection system. The attacker can influence the values of sensors sent to the PLC but does not have complete control of the network, in either the remote or control segment. Such an attack scenario is very common, especially in ICSs with sensors distributed over a large area that send their data to a PLC residing in a physically protected and monitored center.

In this setup, an adversary can replace the original sensor with a malicious one, reprogram the sensor, change its calibration, influence the sensor externally, or just send false data to the PLC over the cable/wireless connection. We argue that this setup is much more practical than an attacker controlling the internal network of the remote segment, or even the network of the control segment, which was the scenario examined in [48]. In our threat model the attacker's input is processed by *both the PLC and the detection system*; hence, the attacker's goal is to produce the input that will simultaneously: (1) cause the intended physical impact on the system, and (2) be close enough to the prediction of the detector to stay under the detection threshold.

Algorithm 3. Find x_{adv} Given a Trained Model $\mathcal{M}$, Test Data With Attack x_{att}, a Detection Threshold τ, Acceptable Noise Level ϵ, Input Constraints ϕ. ADV_LR is the Adversarial Learning Rate

```
1:  function FINDADVINPUT($\mathcal{M}$, $x_{att}$, $l$, $\tau$, $\epsilon$, $\phi$)
2:      $\mathcal{M}' \leftarrow \mathcal{WM} + \nabla_x L(\mathbf{w}, x, y)$ ▷ Add to the model graph the
        wrapper model containing the cost function gradient calculation $L$ given the model parameters $\mathbf{w}$, input $x$ of the
        arbitrary length, and correct output $y$ with respect to $x$.
3:      $noise \leftarrow zeros\_like(x_{att})$
4:      $advIt \leftarrow 0$
5:      while $advIt < MAX\_ADV\_ITERS$ do
6:          $noisy\_input \leftarrow x_{att} + noise$
7:          $noisy\_input \leftarrow enforce(noisy\_input, \phi)$
8:          $model\_residue, grad \leftarrow runModel(\mathcal{M}')$
9:          if $model\_residue < \tau$ then break
10:         $step \leftarrow ADV\_LR * max(abs(grad))$
11:             ▷ Update the noise and make sure it does not pass
                the acceptable level $\epsilon$
12:         $noise \leftarrow noise - step * grad$
13:         $noise \leftarrow clip(noise, \epsilon)$
14:         $advIt \leftarrow advIt + 1$
15:     return $noisy\_input$
```

Other characteristics of our threat model include: (i) the attacker can change multiple sensors, (ii) the attacker can prepare the attacks offline; we argue that if the system is characterized by periodic behavior, the attacker can choose the moment of the attack and precompute the system state in our perfect knowledge threat model, and (iii) the values of the sensors after the attack should be within, or close to, the valid range of the sensor values (e.g., an on/off sensor can't accept any other value).

4.7.2 Adversarial Attack Algorithm

The evaluation of the model's robustness to adversarial evasion attacks was conducted as follows. For each sensor spoofing attack we performed gradient-based search for the adversarial input, as outlined in Algorithm 3. The goal of this algorithm is to search for the minimal perturbation of the sensor's data that is sufficient for performing the attack without being detected. The algorithm aims to achieve the attacker's objective of evading detection (expressed by Equation (4)) by minimizing the value of the model's loss function for the attacker-controlled input, thus bringing the model's prediction close to the attacker's desired result. As the loss function is differentiable, we can solve the attacker's problem by the iterative gradient descent.

This algorithm is adapted to a sequence prediction model that processes the input using subsequences of length l, which we assume is also known to the attacker. The use of a single model to predict multiple overlapping sequences creates additional constraints for the attacker:

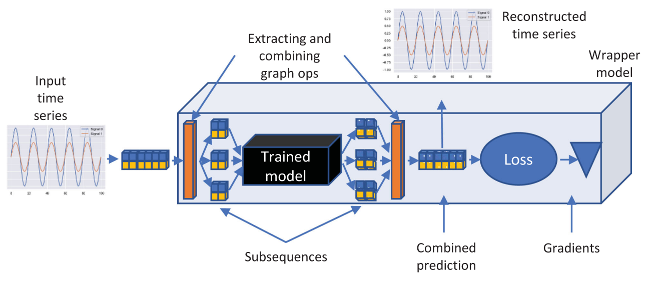

Fig. 7. Wrapper model that allows the calculation of the gradients and optimization of the samples for arbitrary long input sequences based on an original model that predicts short sequences.

- each data point is used (i.e., examined) in l subsequences;
- as the previous data points are used to predict the next one, perturbing a data point at time t will require changes to earlier data points so that the prediction at time t will be close enough to the desired value; these changes will need to propagate back in time.

In order to cope with these constraints, we created a *wrapper model* ($\mathcal{WM}$) for the original model $\mathcal{M}$. $\mathcal{WM}$ represents the processing of all of the input, including enrichment feature generation (as in [17]) and subsequence generation as one graph, allowing for full gradient propagation from the model's prediction to all of the original input, not just a specific subsequence. In other words, each iteration of generating the adversarial sample considers the gradients of all subsequences in which the perturbed data point is used.

This wrapper model (see Fig. 7) is used in Algorithm 3 as follows. First, the algorithm constructs the wrapper model around the attacked model. Then, the algorithm starts searching for the target adversarial input beginning with the attacker's desired result. Then, iteratively, it calculates the gradient of the model's loss function for the adversarial input with respect to the input $\nabla_{x_{att}} L(\mathbf{w}, x_{att}, y)$, where $\mathbf{w}$ are the model parameters, x_{att} denotes the input (partially under the attacker's control), y is the real measured system output, and L is the loss function. L is the same loss function used to train the model, namely the MSE. The gradient is multiplied by the selected learning rate and is used to update the adversarial input for the next iteration. The algorithm proceeds until detection is successfully evaded or until the maximal number of iterations has been reached. Note that we approximated the attack evasion by reducing the prediction residue so that it is less than the threshold. This criteria can easily be extended to use the detection window in addition to the threshold.

5 Experiments and Results

5.1 Datasets

Three datasets commonly used for research in the domain of cyber-physical systems were used in our evaluation. We provide a brief description of the datasets below, and a detailed description is provided in Appendix B, available in the online supplemental material.

5.1.1 SWaT

The Secure Water Treatment (SWaT) testbed was built at the Singapore University of Technology and Design.

The testbed is a scaled-down fully operational six-stage water treatment plant. The dataset contains seven days of recording under normal conditions and four days during which 36 attacks were conducted [18].

5.1.2 BATADAL

The BATADAL dataset represents a water distribution network comprised of seven storage tanks with eleven pumps and five valves, controlled by nine PLCs. The network was generated with epanetCPA [54], a MATLAB toolbox that allows the injection of cyber attacks and simulates the network's response to the attacks. The test dataset contains 2,089 records (from 87 days of recording) with seven attacks.

5.1.3 WADI

Finding another high-quality real-world cyber-physical dataset containing attacks was not easy. The best candidate is the WADI dataset [55], collected from a scaled-down water distribution testbed and compiled by the developers of SWaT. The testbed consists of large water tanks that supply water to consumer tanks. The dataset contains 16 attacks whose goal is to stop the water supply to the consumer tanks. The dataset is significantly larger than the SWaT and BATADAL datasets; there are 1,209,610 data points in the training set and 126 features.

5.2 Research Questions

In our evaluation we aim at addressing the following research questions:

- *RQ#1:* Is the proposed K-S feature selection approach effective and does it assist in generating more accurate anomaly detection models?
- *RQ#2:* What is the influence of the detection mechanism hyperparameters on the detection performance?
- *RQ#3:* What is the maximal detection performance achieved by each detection mechanism for each dataset examined?
- *RQ#4:* Can we suggest a set of hyperparameters that result in high and robust detection performance across all of the datasets examined?
- *RQ#5:* How robust are the proposed models to adversarial evasion attacks?

5.3 Data Preprocessing and Feature Selection

In the first stage of our evaluation, we performed data normalization and feature selection, as described in Section 4.3. Our goal in this stage is to evaluate and answer our first research question *RQ#1*. First, the features were normalized to (0,1) scale. Then, we evaluated the features' stability in the SWaT, BATADAL, and WADI datasets. The evaluation was performed using the K-S* test; features with a K-S statistic of less than 0.2 and with a K-S* statistic of less than 80 were selected for further modeling. To evaluate the effectiveness of the proposed feature selection method, we compared the average detection results produced by the proposed K-S* method with the results of detection done with: *i)* no feature selection, and *ii)* feature

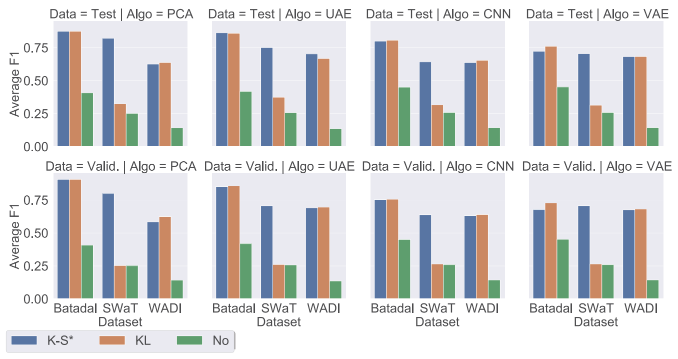

Fig. 8. Feature selection method comparison for three datasets and four model types, both with validation and test datasets. The top row presents the results for the test dataset, and the bottom row presents the results for the validation dataset.

selection using Kullback-Leibler (KL) divergence [56], an established method for measuring the probability distribution distance. As the true probability density functions for the signals are not known, a histogram was used as a probability density estimator [57]. It should be noted that prior work by Lin *et al.* [13] that claimed to use some form of feature selection (attempting to test the stability or consistency of the signal) did not specify any quantitative criteria for the selected features and therefore their method could not be used for comparison. For the evaluation, we trained the same model of each algorithm with features selected based on the three selection criteria results (i.e., none, K-S*, and KL) in two different settings: (1) using a validation set, and (2) using the test set. Note that our initial data exploration revealed that one sensor in BATADAL (P_J280) and one sensor in WADI (2B_AIT_002_PV) are erroneous due to a fault in the data collection process and therefore were removed and were not included in the evaluation.[1]

As Fig. 8 shows, the validation set can provide a good approximation of the test set. It also illustrates the superior performance of the K-S* test over the KL test, especially for SWaT. In addition, both K-S and KL result in significant improvement of the detection score when compared to no feature selection. Based on the test results, we removed the following sensors: AIT201, AIT202, AIT203, P201, AIT401, AIT402, AIT501, AIT502, AIT503, AIT504, FIT503, FIT504, PIT501, PIT502, PIT503 from SWaT; P_J280 from BATADAL; and 1_AIT_001_PV, 1_AIT_003_PV, 1_AIT_004_PV, 1_AIT_005_PV, 2_LT_001_PV, 2_PIT_001_PV, 2A_AIT_001_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_005_PV from WADI.

In addition to data normalization and feature statistic profiling, we subsampled the training and test data of SWaT at a five second rate; this rate was selected, as the SWaT testbed controls water and chemical processes that do not change drastically within seconds. This subsampling served as a low pass filtering mechanism, removing some sensor noise. In addition, the reduction in data volume improved the training speed. However, this subsampling may introduce a slight delay (up to 2.5 seconds) in the attack

1. Sensor P_J280 in BATADAL changes its type from continuous in the training data to discrete in the test data; the values of sensor 2B_AIT_002_PV in WADI become one thousand times larger in the middle of the test data.

TABLE 3
Comparative Performance of Attack Detection for the BATADAL Dataset[1, 2]

Method	Model Hyperparameters	Detection parameters	Precision	Recall	$F1$
Abokifa *et al.* [14]			0.844	0.921	0.88
Chandy *et al.* [15]			0.392	0.857	0.538
1D CNN	LN=18, LR=8, K=2, F=32, A=tanh	D=max, T=1.6, W=20	0.901	0.867	0.883
UAE	LN=4, LR=1, R=0.5, N=0, A=tanh	D=max, T=1.3, W=5	0.957	0.928	**0.943**
VAE	LN=1, LR=1, R=0.5, N=0, A=relu	D=max, T=1.35, W=5	0.963	0.762	0.851
PCA	LN=1, R=0.5	D=max, T=1.35, W=5	0.956	0.806	0.875
UAE Frequency	LN=1, LR=1, R=0.5, N=0.01, A=relu	D=std, T=2.1, W=20	0.918	**0.961**	0.939
PCA Frequency	LN=3, R=0.5	D=std, T=2.58, W=15	0.919	0.806	0.859

[1]*The results of [14] and [15] were calculated from the raw data in [41].*
[2]*Legend: LN - sequence length, LR - number of layers, K - kernel size, F - number of filters, A - activation function, D - detection scoring, R - code or principal components ratio, T - detection threshold, W - detection window. D, T, and W are introduced in Section 4.6. The bold script indicates the best results.*

detection time. For the WADI dataset, subsampling at a 10 second rate was applied, followed by additional low-pass data filtering. This noise removal also helps in coping with PCA's sensitivity to outliers.

5.4 Models' Performance and Generality

In this section, we aim to evaluate the performance of the anomaly detection algorithm and its ability to generalize over the three different datasets: BATADAL, SWaT, and WADI. For consistency with previous publications, we used precision, recall, and $F1$ as the detection performance metrics for all datasets (note that the original BATADAL competition used a different scoring system).

5.4.1 Parameter tuning

To answer the second research question ($RQ\#2$), for each of the models (1D CNN, UAE, VAE, and PCA), we tested multiple hyperparameter configurations, using both grid search and genetic algorithms [58], in order to determine the hyperparameters' influence on the detection effectiveness. Due to space limitations, we present the list of hyperparameters examined for each model type and the optimal parameter values observed across all datasets. We also provide a summary of our findings. More details can be found in the supplementary material, available online.

1D CNN:

- *number of layers (LR)* - eight to 10 layers provided optimal performance;
- *kernel size (K)* - small filters with sizes of two to four were the optimal ones;
- *number of filters (F)* - increasing the number of filters improved the detection performance, where 16 and 32 filters were found to be the optimal number overall, even though eight filters worked better for SWaT ;
- *activation function (A)* - there was no definitive indication whether ReLU or tanh resulted in better performance in all cases;
- *sequence length (LN)* - the optimal length was found to be between 15 and 18 samples.

UAE and VAE:

- *number of layers (LR)* - for the time domain signals, increasing the number of layers did not improve the model's performance, and a single layer was sufficient. For the frequency domain, since multiple frequency bands representing the same time feature resulted in more data, increasing the number of layers to three was optimal;
- *sequence length (LN)* - for the time domain data, using sequence lengths between one and eight was optimal, and increasing the length did not improve the results significantly or at all. For the frequency domain, the optimal sequence length was three;
- *activation function (A)* - no clear advantage was found for either ReLU or tanh;
- *code-to-input ratio (R)* - 0.5 was found to be the optimal ratio overall, and this is probably related to the PCA findings described below;
- *corrupting noise level (N)* - low levels, such as 0.01, or no noise at all produced the best results; this probably stems from the natural noise present in the data;
- *UAE versus VAE* - we found that with equivalent hyperparameters, the UAE performed consistently better than the VAE.

PCA:

- *sequence length (LN)* - in BATADAL, one was the optimal length, while in SWaT and WADI, increasing it (up to seven) provided a small improvement;
- *number of principal components-to-input ratio (R)* - having it equal to the half of the modeled features provided optimal performance; a further increase did not result in improvement, while smaller values caused degradation. We found that this number of principal components captures 99 percent of the variance in the datasets used.

In the next subsection, we use the findings presented above in order to search for a setup (configuration) for each model that is optimal across all datasets.

5.4.2 Attack Detection Performance

We first attempt to answer the third research question ($RQ\#3$), and then we present the best detection results of the optimized models for the BATADAL, SWaT, and WADI datasets. The best detection results were obtained as follows: for each dataset, we evaluated each model type with the configurations obtained from all possible combinations of hyperparameter values identified in Section 5.4.1. The results in terms of the precision, recall, and F1 measure are presented in Tables 3, 4, and 5 for the BATADAL, SWaT, and WADI datasets respectively. For each of the models

TABLE 4
Comparative Performance of Attack Detection for the SWaT Dataset

Method	Hyperparameters	Detection parameters	Precision	Recall	$F1$
DNN [12]			**0.983**	0.678	0.803
SVM [12]			0.925	0.699	0.796
TABOR [13]			0.862	0.788	0.823
1D CNN	LN=15, LR=8, K=2, F=8, A=relu	D=std, T=5.3, W=40	0.868	0.854	0.861
UAE	LN=1, LR=1, R=0.5, N=0, A=tanh	D=std, T=2.9, W=19	0.965	0.778	0.861
VAE	LN=1, LR=1, R=0.5, N=0, A=tanh	D=std, T=2.51, W=37	0.94	0.785	0.855
PCA	LN=4, R=0.5	D=std, T=2.85, W=19	0.92	0.841	0.879
UAE Frequency	LN=3, LR=3, R=0.5, N=0.01, A=tanh	D=std, T=4.13, W=9	0.911	**0.860**	**0.885**
PCA Frequency	LN=3, R=0.5	D=std, T=2.3, W=130	0.925	0.727	0.815

TABLE 5
Comparative Performance of Attack Detection for the WADI Dataset[1]

Method	Hyperparameters	Detection parameters	Precision	Recall	$F1$
PCA[2]			0.3953	0.0563	0.10
KNN[2]			0.0776	0.0775	0.08
FB[2]			0.086	0.086	0.09
EGAN[2]			0.1133	0.3784	0.17
MAD-GAN[2]			0.4144	0.3392	0.37
1D CNN	LN=16, LR=8, K=2, F=16, A=relu	D=std, T=5, W=29[3]	0.697	**0.731**	0.714
UAE	LN=1, LR=1, R=0.5, N=0, A=relu	D=max	**0.916**	0.640	**0.754**
VAE	LN=7, LR=1, R=0.5, N=0, A=relu	D=std, T=6.16, W=14	0.853	0.621	0.718
PCA	LN=7, R=0.5	D=max, T=1.2, W=5	0.807	0.593	0.683

[1]*The WADI dataset did not appear to be suitable for frequency domain analysis as the vast majority of its features do not have clear periodicity.*
[2]*As reported in [43].*
[3]*Each stage of the process was modeled separately, and the results were merged.*

evaluated, we also present the configuration of hyperparameters leading to the best detection results. Note that we denote both the windowed and regular variants of PCA as simply PCA. In addition, for the frequency domain features, the UAE model outperformed the 1D CNN and VAE, and therefore we only present the results of the PCA and UAE models.

As presented in Tables 3, 4, and 5, our method's best detection results come close to or improve upon prior research. For each dataset, we were able to achieve scores higher than previously reported with at least two different methods. In particular, the results for the WADI dataset stand out, as we were able to improve upon the previously reported results by 38 percent. Based on the results presented in Tables 3, 4, and 5, we make the the following observations regarding the comparison between the different methods examined in this study. Among the NN models, UAEs were able to achieve the best scores both for the time and frequency domain features. They also outperformed VAEs for every dataset. 1D CNNs produced comparable, but slightly worse results. Surprisingly, PCA performed very well, and produced the best result for the time domain on the SWaT dataset. We attribute this success to the careful feature selection and noise reduction we performed, as well as to the linear nature of dependencies between many remaining features of the SWaT dataset.

As can be seen in Tables 3, 4, and 5, the hyperparameters of the best models were different across different datasets. To evaluate the robustness of the proposed method in terms of the ability to find a model and configuration that provides the best results across the different datasets (*RQ#4*), we performed experiments aimed at determining the universal sets of hyperparameters' values that provide optimal average detection performance across all three datasets. The universal hyperparameters' values were found by testing all combinations of the best models' hyperparameters on all three datasets and identifying the best configuration for each model; we refer to this model as a universal model. The universal models' parameters (determined as described above), the average detection scores for each dataset, as well as the average $F1$ score for all datasets are presented in Table 6. As can be seen, while the universal models achieve good detection scores, their generality comes at the price of reduced performance. However, we can still observe the same trends: UAE was the best NN model, and 1D CNN was the second best. With the universal models, PCA's performance stands out again.

For both BADATAL and SWaT, UAEs provided the highest detection results across all methods and domains. The PCA's result for the frequency domain was high but significantly lower than the UAE's result. We observed that the shallow, one-layer UAEs that performed well for the time domain did not fare as well in the frequency domain where there was greater variance in the performance results. The reason for this performance difference is a larger number and variety of the frequency domain features comparatively to the time domain, as each time domain feature is represented by several frequency bins. To provide strong and stable results for frequency domain representation, we increased the model's capacity by increasing the inflation layer to be ten-fold instead of three-fold which was used for the time domain and by using more layers (three instead of

TABLE 6
Average Detection (Over 10 Runs) $F1$ of Universal Models[1]

Method	Hyperparameters	BATADAL	SWaT	WADI	Average
1D CNN	LN=18, LR=8, K=2, F=32, A=relu, D=std	0.834	0.773	0.649	0.752 (0.804)
UAE	LN=8, LR=1, R=0.5, A=tanh, D=max	0.876	0.791	0.685	0.784 (0.833)
VAE	LN=1, LR=1, R=0.5, A=relu, D=std	0.751	0.837	0.689	0.759 (0.794)
PCA	LN=1, R=0.5, D=max	0.875	0.820	0.683	0.793 (0.848)
UAE Frequency	LN=3, LR=3, R=0.5, A=tanh, D=std	0.842	0.870	-	**0.856**[2]
PCA Frequency	LN=3, R=0.5, D=std	0.859	0.809	-	0.834[2]
W-UAE	LN=3, LR=3, R=0.5, A=tanh, D=std	0.868	0.775	-	0.822[2]
W-PCA	LN=3, R=0.5, D=std	0.766	0.836	-	0.801[2]

[1]*The detection parameters are not presented, as they are dependent on the specific dataset and found according to Equation (12) as a function of the user-selected false positive rate.*
[2]*Averaged over two datasets only.*

TABLE 7
Training Time, Online Testing Time, and Model Size Comparison for BATADAL[1] [2]

	UAE				1D CNN			PCA			
Sequence length	1	3	5	18	18	18	18	1	3	5	18
Layers	1	1	1	1	4	8	12	1	3	5	18
One training epoch time, s	0.268	0.288	0.306	0.459	0.641	0.878	1.761	0.061	0.100	0.207	0.439
Online testing time, s	0.00032	0.00034	0.00037	0.00118	0.00050	0.00057	0.00164	0.00003	0.00011	0.00055	0.00106
Model size, Kb	67	587	1624	20957	697	3689	50417	6	15	24	81

[1]*Both UAEs and 1D CNNs used a three-fold inflation layer. UAEs did not use inflation in decoding. CNNs used 32 filters.*
[2]*Experiments were performed on a desktop equipped with Intel i7-6700K CPU, 32GB of RAM, and NVIDIA GeForce 1080 GPU.*

one). Such UAE models were robust, as can be seen from the results of the universal UAE frequency model in Table 6.

While these results are encouraging and merit further study and validation, we discovered one limitation of frequency domain detection. As the frequency representation transformation requires the use of windows of at least one period of the signal's dominant frequency, in the frequency domain it is impossible to distinguish between brief attacks that quickly follow one another within the same window. Although in reality this might be a mild concern, in the SWaT dataset many quick attacks occur in succession. Our method usually detects them as one long attack, which reduces the precision metrics. The WADI dataset did not appear to be suitable for frequency domain analysis. Only 44 of 127 features had a clear dominant frequency, and their frequency was very low (with a dominant frequency period of 1440 minutes or 24 hours). Such very long periods result in poor resolution in detecting short attacks (attacks in the WADI dataset are about 10 minutes long). Other features did not have any clear periodicity. We consulted with the dataset developers who indicated that WADI's production cycle was driven by consumer demand, and those demand patterns change hourly. Thus, WADI represents ICSs without a stable production cycle, and therefore frequency domain analysis is not applicable to them. We conclude that in general, frequency domain analysis can contribute to attack detection, however it is less effective in systems with unstable periodicity and is imprecise in detecting short attacks.

We compared detection in the frequency domain with another popular approach to frequency analysis of time-series data, wavelet transformation. Among several wavelet neural network architectures described in literature, we selected a W-ANN architecture, as the most consistent with the methodology used in our study. This architecture is also becoming increasing popular in various research fields such as rainfall prediction [59], tomography [60], and wastewater prediction [61]. In the W-ANN architecture, the signal is first decomposed using a wavelet transformation and then passed to the NN for prediction or classification. We used 3 decomposition levels using the Discrete Meyer wavelet as the mother wavelet function, found to be the best configuration in [61]. The decomposed signal is then passed to the UAE (W-UAE) or to the PCA (W-PCA) with the universal frequency configuration. The results are presented in Table 6. As the results show, the tested W-ANN architecture had lower average detection score than the proposed method. However, this promising direction will be studied further in future research.

5.5 Model Size and Speed Comparison

As shown in [17], 1D CNNs are smaller and faster to train than the recurrent neural networks commonly used for time series prediction. We evaluated the model sizes and the training and testing times for the 1D CNNs, UAEs, and PCA proposed in this research (Table 7 presents the results of this evaluation). The training times were measured in a batch setting, while the testing time was measured in an online mode simulation, evaluating the test data as it arrived. It is evident that UAE-based networks are smaller (for short sequences), take less time to train, and are slightly faster to test than 1D CNNs. However, PCA provides an even smaller footprint and faster times in both settings.

To conclude, our evaluation demonstrates the generality and efficiency of the proposed detection method and models used. The results show that UAEs are a lightweight and effective NN architecture that can be used for anomaly

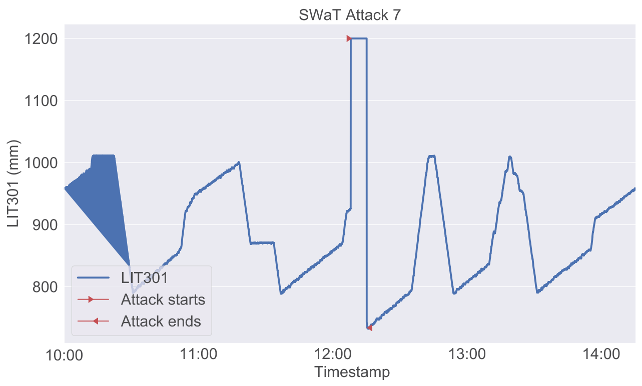

Fig. 9. SWaT attack 7. LIT301 is spoofed to cause an underflow.

and cyber attack detection in CPSs. In addition, PCA provides a simple alternative that can be sufficient in many real-world setups.

5.6 Adversarial Robustness of the Proposed Method

In order to answer *RQ#5*, we evaluated the robustness of the neural network-based models (1D-CNN and UAE) to the attack presented in Section 4.7.2. Note that we did not include the VAE model in this evalaution, since it performed worse than the others. As we did not have access to the testbeds, we based our study on adversarial manipulation of the attacks in the datasets that were caused by sensor value spoofing; this includes 18 of the SWaT dataset attacks and one attack from the WADI dataset. Unfortunately, the BATADAL attacks and the rest of the WADI attacks were not suitable for this experiment due to the replay of the valid signal present in the dataset.

While this method cannot replace testing with a real system, it can provide an approximation of the ability to produce the desired adversarial inputs. The two main limitations of such data-only research are:

1) it is limited to the attacks already present in the dataset; even if these attacks cannot be concealed by an adversarial input, there may be other attacks that have this capability,
2) there is no way of testing the physical effect of an adversarial sample found analytically in the real system.

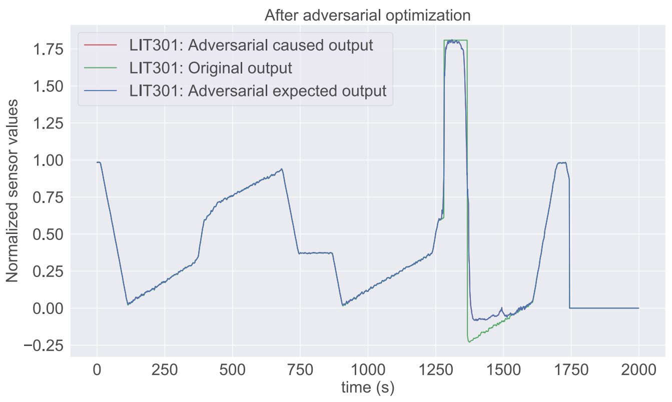

Fig. 11. A 1D CNN model's prediction for SWaT attack 7 after adversarial input optimization (see Section 4.7). The adversarial input is expected to cause an underflow and is undetected by the model. The *adversarial expected output* denotes the signal the attacker wants the model to produce as a prediction, the *adversarial caused output* is the actual model's prediction

We evaluated the ability to create adversarial examples on a model by manipulating a single feature (i.e., sensor). In order to consider the worst-case scenario, no constraints were set on the adversarial noise allowed. The experiments show that our wrapper model-based method is indeed capable of creating adversarial examples that cause the desired malicious physical effect and are not detected by the 1D CNN model.

Fig. 9 illustrates attack 7 from the SWaT dataset in which the measurement of the water level sensor LIT301 is spoofed to be much higher, causing an underflow. When a 1D CNN model created for LIT301 was used to detect anomalies in the relevant time period, it produced the prediction shown in Fig. 10. After the adversarial optimization (described in Section 4.7), we were able to produce input that retains the physical characteristics of the attack (maintaining the spoofed high level for the attack period) and was predicted by the model very closely, thus going undetected (see Fig. 11). However, when an additional feature was added to the model, the adversarial optimization prevented the attacker from achieving his/her goal of physically affecting the system - the adversarial input conforms to the original model's prediction and does not cause an underflow (see Figs. 12 and 13).

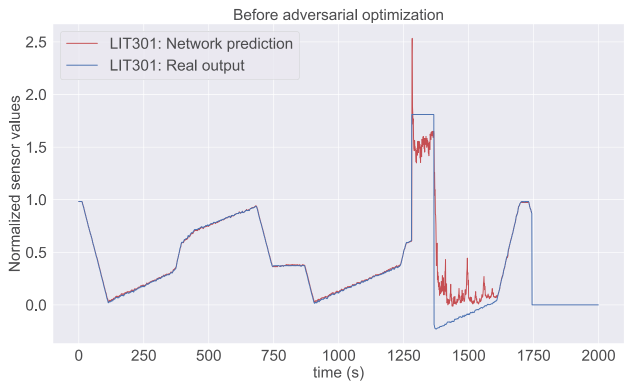

Fig. 10. A 1D CNN model's prediction for SWaT attack 7. During the attack and its recovery, the prediction is very different from the observed value.

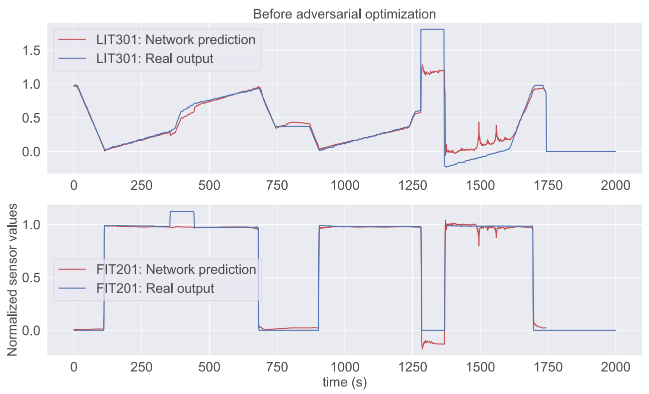

Fig. 12. A 1D CNN model's prediction for SWaT attack 7 with two fields (the artifacts of another attack that occurred before attack 7 can also be seen.

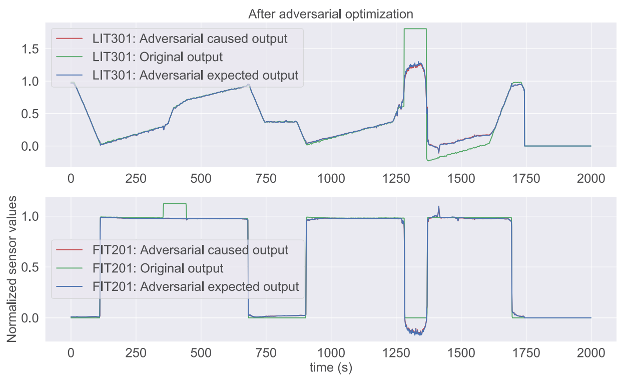

Fig. 13. A 1D CNN model's prediction for SWaT attack 7 after adversarial input optimization when using two features. The adversarial input loses its desired physical impact.

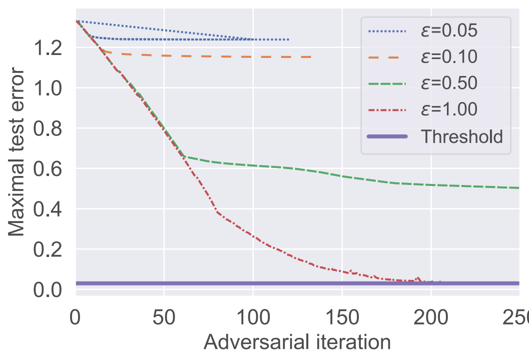

Fig. 14. Change of model prediction error for SWaT attack 7 with iterations of Algorithm 3 for different noise constraint levels ϵ and a threshold of 0.03. With lower noise levels, the attack fails to reach the threshold; with the noise level of one (allowing practically unconstrained signal manipulation), the threshold is reached but at the expense of losing the desired physical impact.

The same behavior was observed with UAE-based models. For attack 2 from the WADI dataset, the adversarial optimization was able to produce the input that maintained the spoofed value of sensor 1_FIT_001_PV while being undetected by an UAE model. However, in order to achieve this effect, the attacker was forced to change the values of *all* of the sensors and actuators modeled significantly. As stated in our threat model, an attacker that is able to spoof all sensor values of the system is highly unlikely and thus is considered out of the scope of this study.

In addition, we observed that using noise to corrupt the input, as described in Appendix B, available in the online supplemental material, increases the robustness of the model to the adversarial input even further, because the random noise applied to the adversarial examples is different during the adversarial training and testing.

Generating adversarial inputs for other attacks demonstrated the same or even more robust behavior: sometimes it was not possible to create adversarial input that preserves the intended physical effect for a one-feature model. In addition, if we constrain the level of noise allowed (e.g., to 0.05), the generation of adversarial inputs fails completely in all of our experiments.

Fig. 14 illustrates the influence of noise on the adversarial input generation for SWaT attack 7. For this attack, the detection threshold for a 1D CNN model found by Equation (12) for a false positive rate of zero is 0.03. As can be seen in Fig. 14, with low noise levels, the attack fails and does not reach the detection threshold. With a very high noise level of one, which allows the attacker to change the signal completely, the threshold is reached, but the attack loses its desired effect, as illustrated in Fig. 13.

The results of our experiments suggest that the proposed detection mechanisms are robust to adversarial evasion attacks. Validating this finding on a real system is a task for future research. This adversarial robustness stands out against adversarial learning's success in other domains (e.g., image processing). A number of factors make the cyber security and cyber-physical domains different from computer vision with regard to the adversarial attacks [62]. The most relevant differences in our case are: the use of regression machine learning algorithms versus classification algorithms used in computer vision, the strong dependencies between the past and future values of the signals, and the strong mutual dependencies between the features due to the laws of physics. These differences impose hard constraints on the sensory data, unlike images where such constraints are largely absent.

6 Conclusion

In this paper, we studied the effectiveness of 1D CNN and AE-based anomaly and cyber attack detection mechanisms. Based on our experiments, we conclude that both 1D CNNs and AEs achieve or exceed the state-of-the-art performance on the three public datasets used in this study, while maintaining generality, simplicity, and a small footprint. It is not clear whether one of these architectures is always preferable over another, and we plan to extend our research with additional datasets to investigate this further. We recommend an ensemble consisting of both models when possible. If a single model must be chosen, in most cases, AEs will likely work out of the box, while 1D CNNs will require a round of hyperparameter tuning to eliminate false positives. We discovered that given proper data preparation and feature selection, PCA-Reconstruction and windowed-PCA can serve as simple and efficient detectors in many real-life cases. Therefore, we recommend to try PCA as a baseline detector before applying NN-based detectors. We also found frequency domain analysis helpful in anomaly and attack detection. Its applicability is subject to several practical requirements; if they are met, frequency domain analysis can provide strong results. The proposed detection method was found to be resilient to adversarial evasion attacks. This finding is a promising one as it allows the system operator to trust the method's decisions. There is a need to confirm these results using actual testbeds. Future research could also examine the effect of adversarial poisoning attacks on the proposed method.

Acknowledgments

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 830927. The authors thank the iTrust Centre in SUTD for creating and providing the SWaT and WADI datasets, Dr. Elchanan Zwecher for his valuable insights, and Rafael Advanced Defense Systems for supporting this work.

References

[1] Wired, "Inside the cunning, unprecedented hack of Ukraine's power grid," Accessed: Mar. 2018, 2016. [Online]. Available: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/

[2] D. Kushner, "The real story of stuxnet," *IEEE Spectrum*, vol. 3, no. 50, pp. 48–53, Mar. 2013.

[3] ForeignPolicy, "Cyberattack targets safety system at Saudi Aramco," Accessed: Mar. 2018, 2017. [Online]. Available: http://foreignpolicy.com/2017/12/21/cyber-attack-targets-safety-system-at-saudi-aramco/

[4] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf.*, 2011, pp. 2195–2201.

[5] A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson, "Attack models and scenarios for networked control systems," in *Proc. 1st Int. Conf. High Confidence Networked Syst.*, 2012, pp. 55–64.

[6] A. Jones, Z. Kong, and C. Belta, "Anomaly detection in cyber-physical systems: A formal methods approach," in *Proc. IEEE 53rd Annu. Conf. Decis. Control*, 2014, pp. 848–853.

[7] V. K. Mishra, V. R. Palleti, and A. Mathur, "A modeling framework for critical infrastructure and its application in detecting cyber-attacks on a water distribution system," *Int. J. Critical Infrastructure Protection*, vol. 26, 2019, Art. no. 100298.

[8] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious scada communications," in *Proc. 12th Int. Conf. Mach. Learn. Appl.*, 2013, pp. 54–59.

[9] R. C. B. Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *Proc. 7th Int. Symp. Resilient Control Syst.*, 2014, pp. 1–8.

[10] O. Chapelle *et al.*, *Semi-Supervised Learning*, vol. 2, Cambridge, MA, USA: MIT Press, 2006.

[11] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," in *Proc.IEEE 18th Int. Symp. High Assurance Syst. Eng.*, 2017, pp. 140–145.

[12] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *Proc. IEEE Int. Conf. Data Mining Workshops*, 2017, pp. 1058–1065.

[13] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, "TABOR: A graphical model-based approach for anomaly detection in industrial control systems," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 525–536.

[14] A. Abokifa, K. Haddad, C. Lo, and P. Biswas, "Detection of cyber physical attacks on water distribution systems via principal component analysis and artificial neural networks," in *Proc. World Environ. Water Resour. Congress*, 2017, pp. 676–691.

[15] S. Chandy, A. Rasekh, Z. Barker, B. Campbell, and M. Shafiee, "Detection of cyber-attacks to water systems through machine-learning-based anomaly detection in SCADA data," in *Proc. World Environ. Water Resour. Congress*, 2017, pp. 611–616.

[16] R. Taormina and S. Galelli, "Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems," *J. Water Resour. Planning Manage.*, vol. 144, no. 10, 2018, Art. no. 04018065.

[17] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proc. Workshop Cyber-Physical Syst. Secur. PrivaCy*, 2018, pp. 72–83.

[18] 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. Int. Conf. Critical Inf. Infrastructures Secur.*, 2016, pp. 88–99.

[19] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 817–831.

[20] N. Sehatbakhsh, A. Nazari, H. Khan, A. Zajic, and M. Prvulovic, "EMMA: Hardware/software attestation framework for embedded systems using electromagnetic signals," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitect.*, 2019, pp. 983–995.

[21] A. Kleinmann, O. Amichay, A. Wool, D. Tenenbaum, O. Bar, and L. Lev, "Stealthy deception attacks against scada systems," in *Computer Security*. Berlin, Germany: Springer, 2017, pp. 93–109.

[22] C. M. Ahmed, J. Zhou, and A. P. Mathur, "Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in CPS," in *Proc. 34th Annu. Comput. Secur. Appl. Conf.*, 2018, pp. 566–581.

[23] J. Giraldo *et al.*, "A survey of physics-based attack detection in cyber-physical systems," *ACM Comput. Surv.*, vol. 51, no. 4, 2018, Art. no. 76.

[24] A. Herzberg and Y. Kfir, "The leaky actuator: A provably-covert channel in cyber physical systems," in *Proc. ACM Workshop Cyber-Physical Syst. Secur. Privacy*, 2019, pp. 87–98.

[25] N. De Bruijn, "Uncertainty principles in fourier analysis," *Inequalities*, vol. 2, pp. 57–71, 1967.

[26] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[27] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, 2018.

[28] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surv.*, vol. 46, no. 4, 2014, Art. no. 55.

[29] S. Han, M. Xie, H.-H. Chen, and Y. Ling, "Intrusion detection in cyber-physical systems: Techniques and challenges," *IEEE Syst. J.*, vol. 8, no. 4, pp. 1052–1062, Dec. 2014.

[30] A. Humayed, J. Lin, F. Li, and B. Luo, "Cyber-physical systems security–a survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1802–1831, Dec. 2017.

[31] C. Alcaraz, L. Cazorla, and G. Fernandez, "Context-awareness using anomaly-based detectors for smart grid domains," in *Proc. Int. Conf. Risks Secur. Internet Syst.*, 2014, pp. 17–34.

[32] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu, "Security and privacy in cyber-physical systems: A survey of surveys," *IEEE Design Test*, vol. 34, no. 4, pp. 7–17, Aug. 2017.

[33] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, 2014, pp. 126–135.

[34] D. Mashima and A. A. Cárdenas, "Evaluating electricity theft detectors in smart grid networks," in *Proc. Int. Workshop Recent Advances Intrusion Detection*, 2012, pp. 210–229.

[35] S. Mishra, Y. Shoukry, N. Karamchandani, S. N. Diggavi, and P. Tabuada, "Secure state estimation against sensor attacks in the presence of noise," *IEEE Trans. Control Netw. Syst.*, vol. 4, no. 1, pp. 49–59, Mar. 2017.

[36] C. Murguia and J. Ruths, "Characterization of a cusum model-based sensor attack detector," in *Proc. IEEE 55th Conf. Decis. Control*, 2016, pp. 1303–1309.

[37] Y. Mo, S. Weerakkody, and B. Sinopoli, "Physical authentication of control systems: Designing watermarked control inputs to detect counterfeit sensor outputs," *IEEE Control Syst. Magazine*, vol. 35, no. 1, pp. 93–109, Feb. 2015.

[38] R. Mitchell and R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 1, pp. 16–30, Jan./Feb. 2015.

[39] M. S. Rahman, M. A. Mahmud, A. M. T. Oo, and H. R. Pota, "Multi-agent approach for enhancing security of protection schemes in cyber-physical energy systems," *IEEE Trans. Ind. Inform.*, vol. 13, no. 2, pp. 436–447, Apr. 2017.

[40] T. Roth and B. McMillin, "Physical attestation in the smart grid for distributed state verification," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 2, pp. 275–288, Mar./Apr. 2018.

[41] R. Taormina *et al.*, "Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks," *J. Water Resour. Planning Manage.*, vol. 144, no. 8, 2018, Art. no. 04018048.

[42] M. Housh and Z. Ohar, "Model-based approach for cyber-physical attack detection in water distribution systems," *Water Res.*, vol. 139, pp. 132–143, 2018.

[43] 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 *Proc. Int. Conf. Artif. Neural Netw.*, 2019, pp. 703–716.

[44] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Annu. Netw. Distrib. Syst. Security Symp.*, 2018.

[45] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proc. MLSDA 2nd Workshop Mach. Learn. Sensory Data Anal.*, 2014, Art. no. 4.

[46] A. Ghafouri, Y. Vorobeychik, and X. Koutsoukos, “Adversarial regression for detecting attacks in cyber-physical systems,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3769–3775.
[47] C. Feng, T. Li, Z. Zhu, and D. Chana, “A deep learning-based framework for conducting stealthy attacks in industrial control systems,” 2017, *arXiv: 1709.06397*.
[48] A. Erba *et al.*, “Real-time evasion attacks with physical constraints on deep learning-based anomaly detectors in industrial control systems,” 2019, *arXiv: 1907.07487*.
[49] T. H. Morris, Z. Thornton, and I. Turnipseed, “Industrial control system simulation and data logging for intrusion detection system research,” in *Proc. 7th Annu. Southeastern Cyber Secur. Summit*, pp. 3–4, 2015.
[50] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning*, vol. 1, Cambridge, MA, USA: MIT Press, 2016.
[51] I. Chakravati, R. Laha, and J. Roy, *Handbook of Methods of Applied Statistics*, Hoboken, NJ, USA: Wiley, 1967.
[52] C. Croarkin *et al.*, “NIST/SEMATECH e-handbook of statistical methods,” *Nist/Sematech*, 2006. [Online]. Available: http://www.itl. nist. gov/div898/handbook
[53] W. Ku, R. H. Storer, and C. Georgakis, “Disturbance detection and isolation by dynamic principal component analysis,” *Chemometrics Intell. Lab. Syst.*, vol. 30, no. 1, pp. 179–196, 1995.
[54] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, “Characterizing cyber-physical attacks on water distribution systems,” *J. Water Resour. Planning Manage.*, vol. 143, no. 5, 2017, Art. no. 04017009.
[55] 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. 3rd Int. Workshop Cyber-Physical Syst. Smart Water Netw.*, 2017, pp. 25–28.
[56] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[57] M. Fryer, “A review of some non-parametric methods of density estimation,” *IMA J. Appl. Math.*, vol. 20, no. 3, pp. 335–354, 1977.
[58] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, 1988.
[59] M. Shoaib, A. Y. Shamseldin, and B. W. Melville, “Comparative study of different wavelet based neural network models for rainfall–runoff modeling,” *J. Hydrol.*, vol. 515, pp. 47–58, 2014.
[60] R. Arulmurugan and H. Anandakumar, “Early detection of lung cancer using wavelet feature descriptor and feed forward back propagation neural networks classifier,” in *Computational Vision and Bio Inspired Computing*. Berlin, Germany: Springer, 2018, pp. 103–110.
[61] M. Zeinolabedini and M. Najafzadeh, “Comparative study of different wavelet-based neural network models to predict sewage sludge quantity in wastewater treatment plant,” *Environ. Monit. Assessment*, vol. 191, no. 3, 2019, Art. no. 163.
[62] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, “Adversarial learning in the cyber security domain,” 2020, *arXiv: 2007.02407*.

Moshe Kravchik received the MSc degree in computer science from the Open University of Israel, in 2007. He is currently working toward the PhD degree at the Ben-Gurion University of the Negev. His research interests include the topics of software and systems security, trusted execution environments, anomaly detection, and the security of industrial control systems.

Asaf Shabtai is a professor with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev. His main research interests include computer and network security, machine learning, security of the IoT and smart mobile devices, and security of avionic and operational technology (OT) systems.