International Journal of Information Security (2023) 22:1099–1114
https://doi.org/10.1007/s10207-023-00677-z

SPECIAL ISSUE PAPER

Real-time detection of deception attacks in cyber-physical systems

Feiyang Cai[1] · **Xenofon Koutsoukos**[1]

Published online: 29 March 2023

Abstract
Detection of deception attacks is pivotal to ensure the safe and reliable operation of cyber-physical systems (CPS). Detection of such attacks needs to consider time-series sequences and is very challenging especially for autonomous vehicles that rely on high-dimensional observations from camera sensors. The paper presents an approach to detect deception attacks in real-time utilizing sensor observations, with a special focus on high-dimensional observations. The approach is based on inductive conformal anomaly detection (ICAD) and utilizes a novel generative model which consists of a variational autoencoder (VAE) and a recurrent neural network (RNN) that is used to learn both spatial and temporal features of the normal dynamic behavior of the system. The model can be used to predict the observations for multiple time steps, and the predictions are then compared with actual observations to efficiently quantify the nonconformity of a sequence under attack relative to the expected normal behavior, thereby enabling real-time detection of attacks using high-dimensional sequential data. We evaluate the approach empirically using two simulation case studies of an advanced emergency braking system and an autonomous car racing example, as well as a real-world secure water treatment dataset. The experiments show that the proposed method outperforms other detection methods, and in most experiments, both false positive and false negative rates are less than 10%. Furthermore, execution times measured on both powerful cloud machines and embedded devices are relatively short, thereby enabling real-time detection.

Keywords Deception attack · Replay attack · Inductive conformal anomaly detection · Cyber-physical systems

1 Introduction

Cyber-physical systems (CPS) involve the integration of sensors, actuators, and computation into physical systems using networking, and they are increasingly deployed in a wide range of domains such as aerospace, automotive, healthcare, manufacturing, and transportation [26]. Networking interconnections between components expose attack surfaces to adversaries who can launch various cyber attacks. Attacks to networked CPS can be deployed on either the sensor network that sends the sensor measurements to the controller or the actuator network that sends the control signals to physical system. In general, attacks can be categorized into two possible classes in [8]: (1) denial of service (DoS) attacks, where the adversary prevents the information from being transmitted or exhausts the resources of the system so that information cannot be processed in a timely manner, and (2) deception attacks, where the adversary sends false information to the controller or actuator. Replay attacks are a special kind of deception attacks, where the adversary records the information in the network at first, and then replaces the actual data with the recorded data to corrupt the integrity of the data.

Recent work demonstrates that successful attacks can lead the system to abnormal behavior,[1] which may significantly undermine the safety of the system [14,20]. Therefore, detection of attacks is paramount to the safe and reliable operation of CPS. Deception attack is one of two categories of attacks on networks of CPSs, and the detection of such an attack has unsurprisingly received considerable attention in the literature [16,21,35,43]. In this paper, we focus on the deception attacks on the sensor and actuator networks of the CPSs. We consider the CPS domain of autonomous driving, and

✉ Feiyang Cai
feiyang.cai@vanderbilt.edu

Xenofon Koutsoukos
xenofon.koutsoukos@vanderbilt.edu

[1] Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA

[1] In this paper, "normal" refers to *the intended behavior of the system without any attack* and "abnormal" refers specifically to *the abnormal behavior due to attacks*.

we present an approach for efficiently and robustly detecting such attacks in real time.

The detection method relies only on sensor observations in contrast to approaches that use additional input signals and need to change the architecture of the system [21,35]. Detection of deception attacks needs to consider time-series observation sequences because point-wise detection methods can only detect very simple deception attacks and can be easily bypassed by more nuanced attacks [5,6]. Although considerable efforts are made for detecting anomalies and attacks in time-series data [20,42], typical approaches focus on low-dimensional data. The problem is challenging for high-dimensional data, for example, from camera and LIDAR sensor measurements in autonomous vehicles. Further, the detection must be performed in real time as observations become available during the system operation.

The first contribution of the paper is a generative model used for detecting anomalies of high-dimensional time-series data. The model is inspired by the world model [19] for learning the long-term behavior of dynamic systems. By training this model with normal data, we can capture the spatial and temporal characteristics of the normal dynamic system behavior. The model has the capacity of predicting future observations by (1) first utilizing the encoder of a variational autoencoder (VAE) [27] to compress a high-dimensional observation into a latent representation, (2) then using a recurrent neural network (RNN) [38] to predict the future encoding based on the compressed latent encoding and historical information, and (3) finally converting the predicted encoding to a predicted observation using the decoder of the VAE. It should be noted that the encoder and decoder are omitted if the dimension of input is of low dimensionality. The expected observation predicted by the model is compared with the actual observation to quantify the nonconformity of actual behavior relative to normal behavior learned during training.

The second contribution is an approach for real-time detection of deception attacks in CPS. It is well-known that RNNs typically lack the ability to capture the long-term dependencies [2]. We propose to recursively use the RNN to predict the observations for multiple time steps in the future. During testing, the expected current observations predicted from multiple steps in the past are compared with the current actual observation, resulting in a series of nonconformity scores. These nonconformity scores are then combined using inductive conformal anomaly detection (ICAD) [30] allowing detection of abnormal behavior in a long sequence. Benefiting from the deep learning generative model, the approach is computationally efficient, thereby enabling real-time detection.

The final contribution of the paper is the comprehensive evaluation for the approach using two simulation case studies, which are (1) an advanced emergency braking system (AEBS) implemented in CARLA simulator [10] and (2) an autonomous car racing example implemented in OpenAI Gym [4], as well as a real-world dataset, which is (3) a secure water treatment (SWaT) dataset [15]. For the AEBS, we focus on a special kind of deception attack, replay attack, which affects the perception component that consumes the high-dimensional image input to estimate the distance to a front obstacle. The adversary has access to the sensor observations during the system operation and uses prerecorded data when the host vehicle is approaching the obstacle in order to deceive the controller and affect the braking, which in turn causes the vehicle to collide with the obstacle. Four attack scenarios with different sensor replay attacks are used to evaluate the proposed approach. The simulation results show that the method can detect three out of four sensor replay attacks with a relatively small number of false positives and negatives, and a short detection delay. For autonomous car racing, a controller is trained to perform the autonomous driving task using the world model [19]. We consider an deception attack to this controller where the adversary modifies the control signal with a malicious command to lead the car off the racing track. The evaluation against such an attack validates the effectiveness of the proposed approach for detecting controller deception attacks using the high-dimensional sensor observations. SWaT testbed is a scaled-down water treatment testbed for cybersecurity research. By spoofing information of sensors and actuators, ten attacks are launched over four days to compromise the normal behavior. By utilizing the low-dimensional sensor inputs, our approach can detect deception attacks in a real-world testbed with a very small number of false positives and false negatives. A VAE-based method that considers only individual frames [5] and an RNN-based method [16] are used for comparison with the proposed approach. The comparative results show that false positives and false negatives of the proposed method are smaller than other methods. Furthermore, we measure execution times of the proposed approach on a powerful cloud machine and an embedded computing device, and the execution times are relatively short, demonstrating that the proposed approach can be used for real-time detection.

The rest of the paper is organized as follows: Section 2 introduces the related work. Section 3 presents the system and threat model, and then formulates the detection problem. Section 4 introduces the generative model used for attack detection, which is a fundamental component in the proposed approach. Section 5 presents the algorithm for detecting deception attacks. Section 6 presents the evaluation results, and Sect. 7 concludes the paper.

2 Related work

2.1 Security of CPS

Attacks on CPS can be summarized as attacks on the actuators, physical plant, and communication networks [8]. The attacks on the networks can be divided into two subcategories: (1) Denial of Service (DoS) attacks, which prevent the sensor readings (or control signals) from being received by the controller (or actuator) or exhaust the system resources so that the transmitted information cannot be processed timely; (2) deception attacks, which use malicious information to modify the original normal information in networks. Deception attacks on the sensor and control signals are investigated in [34] and [32].

In [15], it is shown that a water treatment testbed could not function properly due to deception attacks launched on the actuator and sensor networks. In [41], DoS attacks demonstrated the potential to disrupt the stability of the power grid. To mitigate these threats, a significant amount of work has been proposed for attack prevention and detection. A robust control system against DoS attacks is proposed in [37]. Using an additional authentication control signal, a method for detecting replay attacks is presented in [21,35]. A Chi-square detector and fuzzy logic-based classifier are used to detect and identify distributed DoS attacks in CPS [36]. Neural networks have been used before to detect attacks on CPS. In [1,13], neural network-based detectors are for replay and deception attacks in power systems. These detection methods are limited to low-dimensional data.

2.2 Out-of-distribution detection in CPS

A related research topic is out-of-distribution detection in learning-enabled CPS since such methods can be used to detect some classes of attacks.

A feature space partitioning tree (FSPT) is used to split the input feature space into multiple partitions and to identify those partitions where the training samples are insufficient in [18]. The method can be used for testing whether the data are out of distribution which is equivalent to checking whether this input instance comes from these data-lacking feature space partitions. In [5,7], deep learning-based models, such as VAE and deep support vector data description (deep SVDD), are used to efficiently quantify the difference between the test example and training data set, enabling real-time detection. The VAE-based method can be significantly improved by considering the spatiotemporal correlation of motion in sequences and optimizing the prior distribution in the latent space [12].

VisionGuard (VG) is proposed to detect adversarial examples by checking if the output of a target classifier on a given input image changes significantly after feeding it a transformed version of the image [24].

Although such methods can be used for detecting out-of-distribution and adversarial data, they cannot be applied for detecting deception attacks in CPS since they focus on anomalous examples. In sensor replay attacks, for example, all the frames can be normal, and it is required to consider sequences of high-dimensional observations. Deep learning architectures can also be employed to detect the abnormal behavior of sequences in CPS, such as RNN [16,22], convolutional neural network (CNN) [28], and generative adversarial network (GAN) [31]. The general idea of these methods is to use the network to model the normal behavior of sequences, and the anomalies can be identified by comparing the expectation with the actual test input.

2.3 Inductive conformal anomaly detection

Inductive conformal anomaly detection (ICAD) is a general framework for anomaly detection with a well-calibrated false alarm rate [30]. Its key idea is to evaluate if a new test example conforms to the training data using a nonconformity measure (NCM) to indicate how different an example is from the empirical training data. ICAD is split into offline and online phases. At the offline phase, the training data set is divided into a proper training data set and a calibration data set. For each data in the calibration set, the NCM is used to compute the nonconformity score relative to the proper training set. During the online phase, a p-value can be computed as the fraction of calibration data with nonconformity scores greater or equal to the nonconformity score of a test example. If this p-value is smaller than a predefined threshold ϵ, such a test example will be classified as a conformal anomaly. Many different NCMs have been proposed based on kernel density estimation and k-nearest neighbor [40] and [29]. For detection of anomalous trajectories, the Hausdorff distance and sub-sequence local outlier factor are used for defining sequence-wise NCMs in [30] and [23]. However, such methods are not computationally efficient for detecting anomalies in high-dimensional time-series data. Our previous works [5–7] utilize learning models, such as VAE and deep SVDD, to efficiently compute the nonconformity measure. Besides, by incorporating multiple examples and leveraging a martingale test [11], the robustness of the detection can be improved. This work follows the idea but still fundamentally differs from the previous work; in that *this work is for sequence-wise anomalies, whereas previous works are for point-wise anomalies.* Therefore, the learning model used for computing the nonconformity measure in this work is different, and the detection pipeline needs to be adapted for the detection of anomalies in time series.

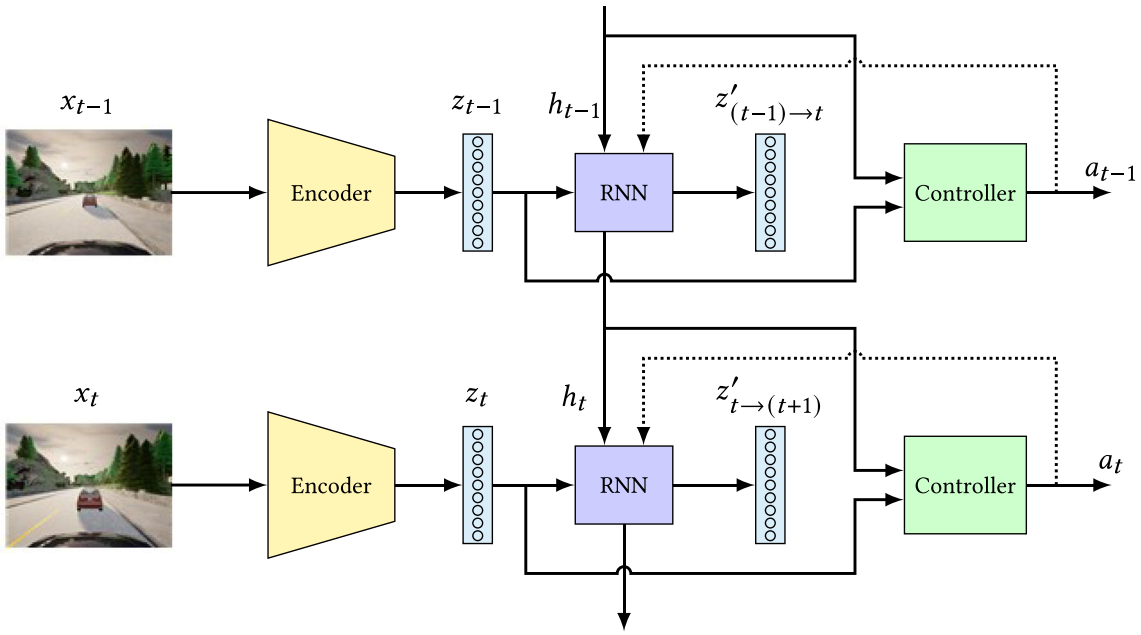

Fig. 1 Architecture of the world model [19]

2.4 World model

Extensive work in the literature attempts to learn a good dynamics model of the system to benefit the training of reinforcement learning policies [9,33,39]. Gaussian process models and Bayesian neural networks are employed to learn the system dynamic in [33] and [9], respectively, and then many trajectories sampled from the models are used to train an agent for a specific task. However, these methods are only applicable to low-dimensional observations. An RNN is leveraged to learn the dynamic model to facilitate the task of the reinforcement learning controller [39]. RNNs used to model high-dimensional time series often have millions of parameters, which bottlenecks the agent's training [19]. The world model is inspired by the human cognitive system that learns an abstract representation of both spatial and temporal aspects of a complex physical environment and is proposed in [19]. By using compressed features, a simple reinforcement learning controller can be trained to perform the required task. Such a world model has been shown to effectively learn the dynamic model and facilitate the training of agents through evaluations in the OpenAI gym [4].

The architecture of the world model is shown in Fig. 1, and it contains two components, Vision (V) and Memory (M). The vision model is a VAE, which encodes the current high-dimensional observation x_t into the low-dimensional latent representation z_t. The memory model is an RNN, which takes the low-dimensional encoding z_t and the historical information embedded in the hidden states of RNN h_t, to predict the latent encoding $z'_{t\to t+1}$ in the next time step. We should note that in the proposed generative model, such $z'_{t\to t+1}$ is utilized to reconstruct an input as the prediction of the next time step, which is different from the world model where $z'_{t\to t+1}$ is not used.

In [19], a simple single-layer linear controller C is trained separately from V and M to perform a specific task. It should be noted that the control signal can be used when the memory RNN makes the predictions for the future, which allows the memory M to take into account the influence of the control signal on the dynamic behavior.

An important property of the world model is that after training, it can be used to imagine a virtual environment, called *dream environment*, by recursively running the RNN to predict the latent encodings for future steps. This can be illustrated using Fig. 1 where instead of using the compressed representations z_t of the actual observation x_t, the prediction from the last time step $z'_{(t-1)\to t}$ can be used as the input to the RNN to predict $z'_{(t-1)\to(t+1)}$ for the next step. In this fashion, the controller can be trained within this dream environment by only using a single seed observation instead of an actual environment.

3 Problem formulation

In order to formulate the problem, let us consider an autonomous vehicle, whose simplified architecture is illustrated in Fig. 2. The sensors which may include inertial measurement unit (IMU), global positioning system (GPS), camera, and LIDAR observe the states of the autonomous

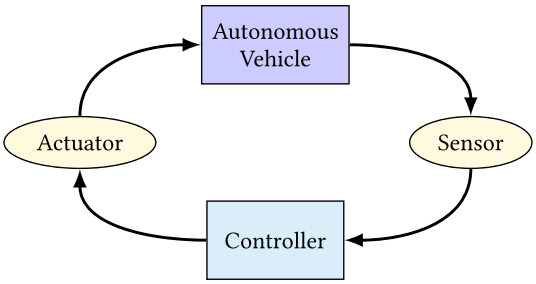

Fig. 2 A typical CPS architecture

vehicle and the environment and feed the information to the controller through the sensor network. In order to realize a specific task, the controller interprets the sensor measurements, makes control decisions following predefined control logic, and sends the control commands, such as gas, brake, and steer signals, to the actuator through the actuator network. In response to the control commands, the states of the vehicle change, along with the environment of the vehicle, and the sensors are used again to close the operation loop of the system.

Although the system may be well-designed and tested, the sensor and actuator networks can still be vulnerable to cyber attacks when the system operates in the real world. A successful attack may lead to abnormal behavior of the system and greatly undermine the safety of the system. Therefore, detecting cyber attacks on CPS is of great significance, for example, to switch to a backup system or human intervention.

In this paper, we focus on deception attacks on the sensor networks and actuator networks and consider the problem of efficiently and robustly detecting such attacks using only the sequence of sensor observations. An alternative method could be used to design detection methods that explore between sensors that observe the environment and vehicle sensors such as the IMU and GPS. However, such methods need to assume that some sensors are not attacked and require fusion of sensor measurements which may increase the attack surface. For this reason, the objective in the proposed approach is to detect attacks using only the time-series sensor observations. We assume that the adversary gains the access to the sensor network and actuator network and has capabilities of collecting and tampering with the information transmitted in networks.[2] For the controller deception attacks, it is also assumed that the sensor network and the control network are not attacked at the same time since in this case, the adversary will be able to completely hide the effects of the attack. Moreover, the detection approach should be able to scale to high-dimensional data, e.g., images from camera and point cloud data from LIDAR sensor, as they are increasingly used in modern CPS, such as autonomous vehicles.

During system operation, the time-series observations become available sequentially. At each time step, the objective of the detection is to quantify how strange the time series up to the current step is relative to the normal system behavior. Online detection requires the algorithm to use only the actual observation sequence that is not complete.

4 Generative model for detection

In this section, we propose a sequential generative model that is used for the detection of anomalies in time-series data, with a special focus on the high-dimensional sequences. The proposed model is inspired by the world model for learning the temporal and spatial representations of physical environments [19]. The model contains a VAE to encode the spatial information into a latent space representation and an RNN to encode the temporal dependencies into the hidden state. The idea is to compute a representation of the normal dynamic system behavior by training this generative model using normal data. We should note that the VAE can be omitted when the observations are low-dimensional. For convenience, only the high-dimensional inputs are used in the introduction below.

The main advantage of the proposed model is that it allows predicting observations for future steps. Specifically, given the observation (image) x_{t-1}, the encoder of the VAE encodes the high-dimensional data point as a Gaussian distribution over a latent space parameterized by mean μ_{t-1} and standard deviation σ_{t-1}. A point z_{t-1} is sampled from this distribution, which is the low-dimensional representation of the original input. Then, the RNN consumes such a latent representation z_{t-1} and the historical information h_{t-1} to predict the latent representation $z'_{(t-1)\to t}$ in the next step t. This part is similar to the world model. However, the world model does not utilize the predicted low-dimensional representation $z'_{(t-1)\to t}$ but feeds the hidden state of the RNN h_{t-1} into the reinforcement learning controller, which is shown in Fig. 1. On the contrary, our proposed generative model uses this representation $z'_{(t-1)\to t}$ to reconstruct the original image $x'_{(t-1)\to t}$. The expected observation $x'_{(t-1)\to t}$ can be predicted based on the previous observation x_{t-1} as illustrated in the top of Fig. 3. Similar to the world model, the influence of the control signals can be considered by feeding the signals to the RNN to predict the next state.

The proposed generative model predicts the observations for multiple steps in the future similarly to the dream environment in the world model. In Fig. 3, after getting the $z'_{(t-1)\to t}$ in the first inference of the RNN, the RNN continues to predict the next latent representation $z'_{(t-1)\to(t+1)}$ by using the virtual encoding $z'_{(t-1)\to t}$ instead of the actual

[2] It should be noted that the observations (control signals) can be either from historical data that has been occurred in the network or modified arbitrarily by an attacker.

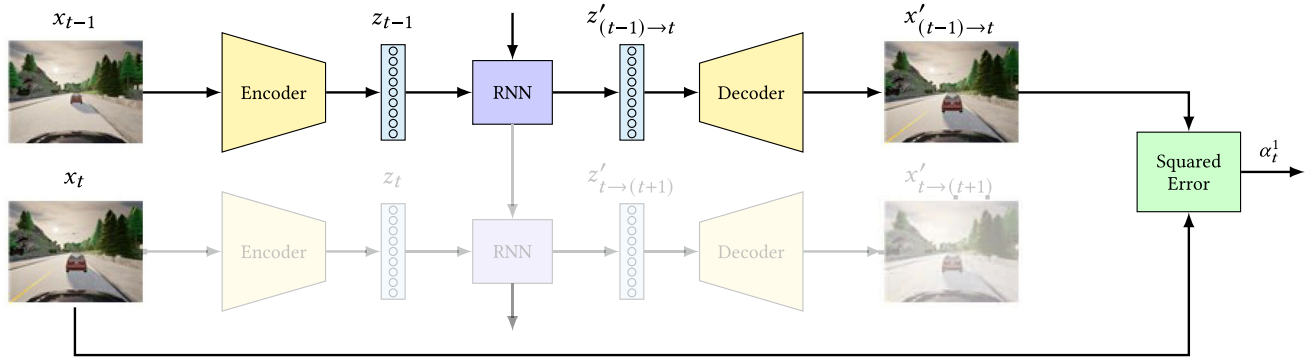

Fig. 3 Illustration of 1-step nonconformity measure using generative model

encoding z_t. Recursively, the RNN can be used to compute m latent encodings $(z'_{(t-1)\to t}, \ldots, z'_{(t-1)\to(t+m-1)})$. Using the decoder, these encodings are converted to a series of predicted observations $(x'_{(t-1)\to t}, \ldots, x'_{(t-1)\to(t+m-1)})$. In this manner, the model can generate sequential data by only using a single seed input.

We should emphasize that it is also possible to include the control signals to make predictions for multiple steps. The control signals should be recursively generated by the controller to feed to the RNN for predicting the next latent encoding. However, the actual observations are not available during the prediction process, and therefore, it is not applicable to use the observation directly in the controller. To overcome this limitation, the controller can use a low-dimensional representation encoded by the VAE as the input, similar to the controller in the world model [19].

The training procedure of the generative model consists of the following steps. After collecting the training data set, the VAE is trained first by minimizing the reconstruction error and the Kullback–Leibler (KL) divergence between the prior and posterior [27]. Next, we compute and store the latent representations for the training data by feeding them into the trained VAE model. The last step is to train the RNN using the sequences of the latent representations of the training data. The objective is to predict the probability distribution of the latent representation in the next time step. In order to model the probability distribution of the latent representation, a mixture density network (MDD) [3] is connected after the RNN as the output layer as in the training for the world model [19]. After training, the model can generate a sequence of consecutive images from a single input, and the sequence is very similar to the actual sequence. Further, the model can predict the sequence of observations for N steps, and this prediction can be repeated at every time step resulting in N predictions corresponding to a specific time step in the future from N different time steps in the past.

5 Attack detection

The detection method is based upon inductive conformal anomaly detection (ICAD), which requires a suitable nonconformity measure (NCM) defined to measure the difference between a test example and the training data set. In this section, we first introduce a novel NCM that utilizes the proposed generative model. Then, based on this NCM, we present an ICAD algorithm for detecting deception attacks.

5.1 Nonconformity measure

Let us consider a set of normal sequences denoted by $\Gamma_{\text{train}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$, where $\mathcal{X}_i : i = 1, \ldots, n$ is a time series of data points $(x_1^i, \ldots, x_{l_i}^i)$ with variable length l_i. During runtime, a sequence of test observations $(x_1^{n+1}, \ldots, x_{l_{n+1}}^{n+1})$ arrive to the system one by one. At time step t, the objective is to first quantify how different the time series up to t $(x_1^{n+1}, \ldots, x_t^{n+1})$ is from the set of normal sequences Γ_{train} used for training, and then raise an alarm if the test sequence deviates considerably from the sequences used at design time.

Most of the NCMs used in ICAD, such as k-nearest neighbor (k-NN) NCM [40] and VAE-based NCM [5], however, focus on point-wise detection. Namely, they aim to check if a single test instance x_t conforms to the training data set Γ_{train}. Such methods lack the capacity of capturing the temporal dependencies of time-series data and cannot be used for detection of deception attacks.

As discussed in Sect. 4, we propose a generative neural network model which aims to learn compressed spatial and temporal representations of the physical system and environment and can be used to predict future observations. By training the generative model using the normal sequences Γ_{train}, the model captures the distribution of complex high-dimensional observation sequences. Therefore, if the actual

observations do not conform to the predictions from the generative model, the sequence of observations can be regarded as anomalous behavior. Following the description of the generative model in Sect. 4 and Fig. 3, we compute the squared error between the current actual observation x_t and predicted current observation from the last time step $x'_{(t-1)\to t}$ to compute the nonconformity, which is termed as 1-step NCM.

Although an RNN is used in the proposed model to take into account dependence on information at the previous step, it is very hard to learn long-term dependencies, especially for high-dimensional sequences. The 1-step NCM defined above can be used only across two adjacent time steps, and it is insensitive to attacks where the time-series sequence is changed gradually and slowly. In order to overcome this limitation, the generative model is used to predict the observations for multiple time steps in the future. The difference between the actual observation and the expected observation that is predicted from multiple steps in the past can be used to monitor long-term dependencies. The detailed description for predicting observations for multiple steps can be found in Sect. 4. We denote the generative model as $\mathcal{P}$, and the process of predicting observation $x'_{(t-k)\to t}$ after k-steps based on the observation x_{t-k} at $t-k$ can be described as

$$x'_{(t-k)\to t} = \mathcal{P}(x_{t-k}, k),$$

and the k-step NCM at time t can be naturally defined as

$$\alpha_{t,k} = ||x_t - x'_{(t-k)\to t}||^2. \tag{1}$$

It should be noted that the control signal can be incorporated in the prediction process by feeding the control signal into the RNN. Therefore, using the same definitions of NCM, the NCM can capture the anomalies in the sequence of observations caused by malicious control signals.

5.2 Detection algorithm

After the definition of the NCM based on the proposed generative model, we introduce the detection algorithm. The algorithm is divided into offline and online phases as described below.

5.2.1 Offline phase

During the offline phase, the training set of sequences $\Gamma_{\text{train}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ is split into a proper training set $\Gamma_{\text{proper}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ and a calibration set $\Gamma_{\text{calibration}} = \{\mathcal{X}_{m+1}, \ldots, \mathcal{X}_n\}$. The proper training set Γ_{proper} is used to train the generative model.

The next step is to compute the nonconformity scores for the sequences in the calibration set. In order to model the long-term dependencies, the difference between the actual current observation and the expected current observation predicted from multiple steps earlier is incorporated into the computation of the nonconformity. For each observation x_t^i in the calibration set from the sequence $\mathcal{X}_i : i \in \{m+1, \ldots, n\}$, we compare the observation with N expected observations predicted from 1 to N steps earlier. Therefore, we compute N different nonconformity scores by using 1-step, 2-step,..., N-step NCM [Eq. (1)]. It is worth noting that the choice of N depends on the dynamic evolution of the system. For a slowly evolving system, a large N should be chosen in order to be able to capture a significant change within N steps, which will be beneficial for detection. Moreover, for each observation x_t^i in the calibration data set, we predict the observations for the next N steps $\{x'_{t\to t+1}, x'_{t\to t+2}, \ldots, x'_{t\to t+N}\}$. Although these observations are not used at the current time step, they are available for the calculation of nonconformity scores at the next N time steps.

Because the prediction accuracy will decrease as the number of prediction steps k increases, the nonconformity score computed by comparing with a prediction from a longer past is very likely to be greater than the one computed by comparing with prediction from a shorter past. The nonconformity scores of the calibration data are used for comparing with nonconformity scores of the test data during online phase. Therefore, for fair comparison, after computing the N nonconformity scores for each observation in the calibration data, the nonconformity scores are clustered into N different sets based on the number k of steps between the actual and predicted observations. Therefore, the k-th calibration set $\mathcal{C}_k$ contains all the nonconformity scores using the k-step NCM, that is, $\mathcal{C}_k = \{\alpha^i_{t,k} : (i,t) = \{m+1, \ldots, n\} \times \{1, \ldots, l_i\}\}$. Each cluster of nonconformity scores is sorted and stored for use during the online phase. The detailed algorithm for the offline phase is shown in Algorithm 1.

5.2.2 Online phase

During the online phase, we consider a sequence of observation $\{x_1^{n+1}, \ldots, x_{l_{n+1}}^{n+1}\}$ arriving at the detector one by one. At time step t, for the observation x_t^{n+1}, we compute N nonconformity scores $\{\alpha_{t,1}^{n+1}, \ldots, \alpha_{t,N}^{n+1}\}$ in the same way as calibration data, i.e., by computing N squared errors between the actual observation x_t^{n+1} and the expected observations $\{x'^{n+1}_{(t-N)\to t}, \ldots, x'^{n+1}_{(t-1)\to t}\}$ predicted from time steps $t-N$ to $t-1$. Using N nonconformity scores improves detection by considering long-term dependencies and also improves robustness.

Next, for each nonconformity score $\alpha_{t,k}^{n+1} : k \in \{1, \ldots, N\}$, we compute its corresponding p-value as the fraction of nonconformity scores in k-th calibration set $\mathcal{C}_k$ that are greater or equal to $\alpha_{t,k}^{n+1}$. This p-value can be expressed as

Algorithm 1 Offline phase of detecting deception attacks.

Input: a training set of sequences $\Gamma_{\text{train}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$, number of calibration sequences $n-m$; number of observations predicted from past N;
Output: N calibration sets of nonconformity scores $\{\mathcal{C}_i\}_{i=1}^N$
1: Split the training set of sequences $\Gamma_{\text{train}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ into a proper training set of sequences $\Gamma_{\text{proper}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_m\}$ and a calibration set of sequences $\Gamma_{\text{calibration}} = \{\mathcal{X}_{m+1}, \ldots, \mathcal{X}_n\}$
2: Train the generative model $\mathcal{P}$ (a VAE and a RNN) using the proper training set of sequences Γ_{proper}
3: **for** $i = m+1$ to n **do**
4: **for** $t = 1$ to l_i **do**
5: ▷ Compute the N nonconformity scores
6: **for** $k = 1$ to N **do**
7: $\alpha_{t,k}^i = ||x_t^i - x'^i_{(t-k)\to t}||^2$
8: **end for**
9: ▷ Predict the observations for next N time steps
10: **for** $k = 1$ to N **do**
11: $x'^i_{t\to(t+k)} = \mathcal{P}(x_t^i, k)$
12: **end for**
13: **end for**
14: **end for**
15: Construct N calibration sets of nonconformity scores $\{\mathcal{C}_i\}_{i=1}^N$, where $\mathcal{C}_k = \{\alpha_{t,k}^i : i = m+1, \ldots, n; t = 1, \ldots, l_i\}$

$$p_{t,k} = \frac{|\{\alpha_k \in \mathcal{C}_k | \alpha_k \geq \alpha_{t,k}^{n+1}\}|}{|\mathcal{C}_k|}.$$

It is possible that $\alpha_{t,k_1}^{n+1} < \alpha_{t,k_2}^{n+1}$ for $k_1 < k_2$ since the observation predicted from $t-k_2$ may not be as accurate as the one predicted from $t-k_1$. By clustering the nonconformity scores of the calibration data based on the number of prediction steps k used in their computation, we do not need to compare the actual observations with predictions made at different time steps which may be of different quality. Merging these nonconformity scores, for example, using average is not appropriate and will result in loss of information since the p-value for α_{t,k_1}^{n+1} could be larger than the p-value for α_{t,k_2}^{n+1}. Since the nonconformity scores for the calibration data are sorted in the offline phase, the calculation of p-values can be accelerated by using a binary search algorithm.

The next step is to use a martingale to test if there are many small p-values in the set of $\{p_{t,1}, \ldots, p_{t,N}\}$ [11]. The martingale can be defined as

$$M_t = \int_0^1 \prod_{k=1}^N \epsilon p_{t,k}^{\epsilon-1} d\epsilon.$$

If there are many small p-values, M_t will be very large indicating a sequence of observations that are not conformal to the training data. By incorporating the control signal into the prediction process, the NCM can measure the anomalies caused by malicious control signals, and our detection method can thus detect the controller deception attacks.

A stateful detector test defined by a cumulative SUM (CUSUM) procedure is applied to the sequence of martingale values in order to detect the anomaly robustly similar to Cai and Koutsoukos [5]. The detailed algorithm of the online phase can be found in Algorithm 2.

Algorithm 2 Online phase of detecting deception attacks.

Input: a test sequence $\mathcal{X}_{n+1}$, where each sequence $\mathcal{X}_i = (x_1^i, \ldots, x_{l_i}^i), i = 1, \ldots, n+1$; N calibration sets of nonconformity scores $\{\mathcal{C}_i\}_{i=1}^N$; number of observations predicted from past N; threshold τ and parameter δ of CUSUM detector
Output: Boolean variable $Anom_t$
1: **for** $t = 1$ to l_{n+1} **do**
2: ▷ Compute the N nonconformity scores and p-values
3: **for** $k = 1$ to N **do**
4: $\alpha_{t,k}^{n+1} = ||x_t^{n+1} - x'^{n+1}_{(t-k)\to t}||^2$
5: $p_{t,k} = \frac{|\{\alpha_k \in \mathcal{C}_k | \alpha_k \geq \alpha_{t,k}^{n+1}\}|}{|\mathcal{C}_k|}$
6: **end for**
7: ▷ Compute the martingale value
8: $M_t = \int_0^1 \prod_{k=1}^N \epsilon p_{t,k}^{\epsilon-1} d\epsilon$
9: ▷ CUSUM procedure
10: **if** $t = 1$ **then**
11: $S_t = 0$
12: **else**
13: $S_t = \max(0, S_{t-1} + M_{t-1} - \delta)$
14: **end if**
15: $Anom_t \leftarrow S_t > \tau$
16: ▷ Predict the observations for next N time steps
17: **for** $k = 1$ to N **do**
18: $x'^{n+1}_{t\to(t+k)} = \mathcal{P}(x_t^{n+1}, k)$
19: **end for**
20: **end for**

6 Evaluation

In this section, we evaluate the proposed approach for detecting (1) sensor replay attacks using an advanced emergency braking system (AEBS) implemented in the CARLA simulator [10], (2) controller deception attacks using an autonomous car racing example implemented in OpenAI Gym [4], and (3) deception attacks using a real-world secure water treatment (SWaT) dataset [15]. The experiments are performed on a Ubuntu Linux virtual machine with a 48-core CPU and an RTX 6000 GPU, which is provided by Chameleon Cloud Platform [25] supported by the National Science Foundation.

6.1 Advanced emergency braking system

Advanced emergency braking system (AEBS) is a typical CPS, whose objective is to detect an approaching obstacle and safely stop the host vehicle [5]. A perception component receives the image captured by an onboard camera and uses a convolutional neural network to estimate the distance to a front obstacle. The distance, along with the velocity of the host vehicle, is used by a reinforcement learning controller to

generate an appropriate brake force to stop the host vehicle avoiding a potential collision.

6.1.1 Experimental setup

We collect 100 episodes by varying the position of the obstacle, the distance to the obstacle, and the velocity of the host vehicle, on a fixed road segment with the same obstacle. The length of each episode is 199 frames, with a total of 19, 000 images in the training dataset. Besides, the regression target, distance to the obstacle, is nearly uniformly distributed between $0\,m$ and $50\,m$ so that the training dataset is almost balanced. The data are split into two partitions: 80 episodes are used as the proper training set and 20 as the calibration set. We collect 100 additional episodes that are used as normal test sequences.

The detection method utilizes a generative model consisting a VAE and an RNN, which are trained using the proper training set. We use similar network architectures and training hyperparameters as in [19]. Since the controller is not considered in this experiment, the RNN in the generative model uses only the latent representations from the VAE and does not include the controller signals in its input. We compare the proposed method against a VAE-based method that considers only individual frames presented in [5].

6.1.2 Replay attack 1

The sensor replay attack occurs when an adversary (1) collects the sensor observations and (2) replays the collected data at different time instants of an episode. Using a replay attack, the adversary can spoof the camera images with images recorded earlier from the same camera. The objective of the adversary is to cause the predicted distance to the obstacle to be larger than the actual distance. In this case, the controller will be deceived to apply a soft brake force even when the host vehicle is very close to the obstacle, which will eventually cause the host vehicle to collide with the obstacle.

We use 100 episodes of normal sequences and 100 episodes of sequences under attack to evaluate the detection method. In the sequences under attack, the video segment that is recorded between the time period t_0 to $t_0 + 40$ is replayed in a forward order at time step t_1. The time instants t_0 and t_1 are randomly sampled from $\{5, \ldots, 30\}$ and $\{80, \ldots, 100\}$, respectively.

In order to characterize the detection performance, we report the false positives, false negatives, and the average detection delay by considering different values for the prediction horizon N and CUSUM parameters σ and τ that are shown in Table 1. The results show that the proposed method can detect such a sensor replay attack with a few false positives and false negatives, and a very short delay. This is because a sequential generative model is utilized to compute the nonconformity measure, where an RNN can learn the temporal characteristics of the normal dynamic system behavior. Such a nonconformity measure can capture the anomalies in sequence-wise. Once the attack occurs, test observations do not conform with the expected observations predicted by the generative model, which will trigger the alarm of the proposed method. In contrast, the VAE-based detector [5] utilizes a reconstruction-based nonconformity measure to capture the point-wise anomalies. Such a nonconformity measure is defined as the error between the reconstructed input from VAE model and the original input. In our evaluation, the observations for individual frames are in the same distribution of the training dataset and can be reconstructed by the VAE quite well. Therefore, the test episode will not trigger the alarm of the VAE-based detector. The VAE-based method performs considerably worse than the proposed method exhibiting large number of false negatives and large delay of detection.

We show the simulation results for a specific sequence under attack in Fig. 4 where we plot the predicted distance to the obstacle, p-values, and stateful detector S-values for the VAE-based and proposed method. In the episode shown, the replay attack starts at time step 96 corresponding to 4.8 s from the beginning of the episode (with a sampling rate of 50 ms). The frames used in the attacks are from the same distribution as the training data set, and therefore, the VAE-based method that uses individual frames cannot detect the

Table 1 False positives, false negatives, and detection delay for detecting sensor replay attack I in AEBS

Types	N, δ, τ	False positive	False negative	Average delay (frames)
VAE-based	10, 1, 147	14/100	82/100	12.44
	10, 0, 243	17/100	81/100	14.21
	20, −2, 282	26/100	71/100	12.38
	20, 3, 260	14/100	83/100	10.82
Proposed	10, −2, 64	16/100	2/100	1.19
	10, −2, 72	14/100	2/100	1.67
	20, −2, 57	14/100	0/100	0.83
	20, −1, 35	12/100	0/100	0.27

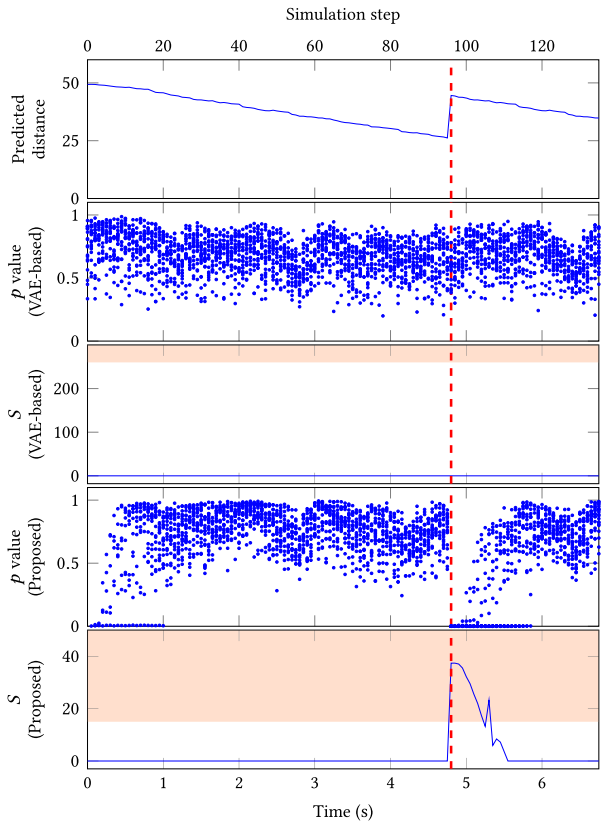

Fig. 4 An episode under the sensor replay attack I in AEBS (detector parameter: (1) VAE-based: $N = 20$, $\delta = 3$, $\tau = 260$; (2) proposed: $N = 20$, $\delta = 13$, $\tau = 15$)

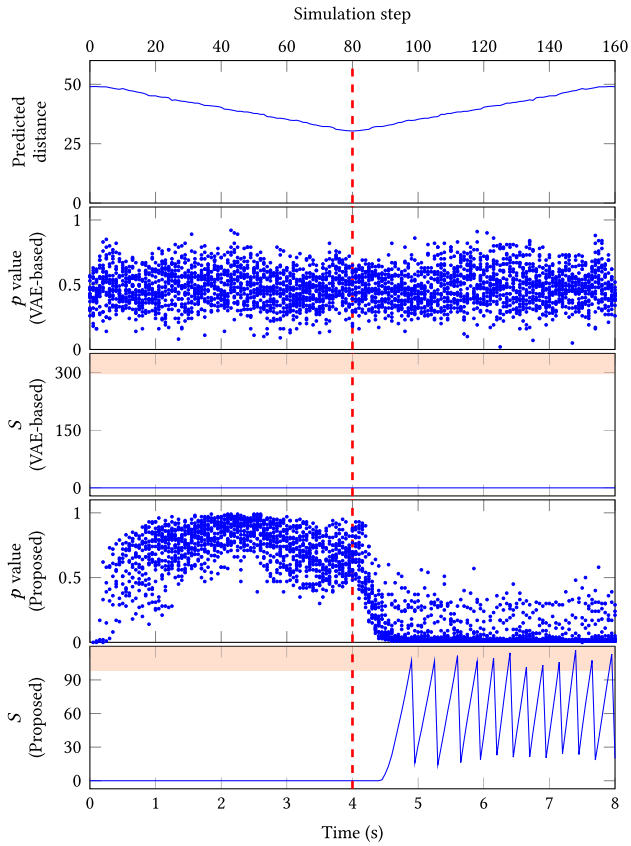

Fig. 5 An episode under the sensor replay attack II in AEBS (detector parameter: (1) VAE-based: $N = 20$, $\delta = 27$, $\tau = 296$; (2) proposed: $N = 20$, $\delta = 7$, $\tau = 98$)

attack and trigger an alarm. Since the proposed method uses a model that captures the temporal dependencies of the observation sequences, the p-values become very small once the attack occurs and can be detected using the approach.

6.1.3 Replay attack II

In the experiment described above, the replay attack results in abrupt changes in the observation sequences that can be easily detected by the proposed approach. In this experiment, we consider a type of replay attack that changes the observations gradually and slowly. In this scenario, at the time step t_1, an adversary starts to replay the collected from time t_0 in reverse, that is, from the camera point of view, the vehicle starts moving backward. Suppose the normal sequence is denoted as $(x_1, \ldots, x_{t_1-1}, x_{t_1}, x_{t_1+1}, \ldots, x_l)$. Using this attack, this sequence becomes $(x_1, \ldots, x_{t_1-1}, x_{t_1}, x_{t_1-1}, \ldots, x_1)$, where t_1 is randomly selected in $\{80, \ldots, 100\}$. We collect 100 episodes with this replay attack for evaluation.

Figure 5 illustrates the detection process. It is not surprising that the VAE-based detector trained on individual frames cannot detect this type of attack. In the proposed approach, the p-values start to decrease when the attack is deployed, and the detector generates alarms indicating that the sequence is abnormal. Table 2 reports the false positives, false negatives, and average detection delay for the two methods. Note that for some combinations of N and detector parameters σ and τ, the detector is insensitive, and all test episodes will be identified as negatives. Therefore, the average delay cannot be computed and labeled as "N/A" in the table. The results demonstrate that the approach can detect the slowly changing replay attack with a small number of false positives and false negatives. However, the number of falses for detecting the replay attack II is larger than the replay attack I because in replay attack II, the changes at the observation sequences are very gradual. The performance of the detector for a large time horizon (e.g., $N = 20$) is improving since it takes a longer time for such an attack to change the observation sequence.

6.1.4 Stuck sensor attack

The sensor may get stuck due to an attack. In this experiment, we simulate a scenario where the attack results in the camera sensor getting stuck which can be viewed as a particular type of replay attack. We assume that the camera sensor is stuck at time step t_1, where t_1 is randomly sampled in $\{80, \ldots, 100\}$

Table 2 False positives, false negatives, and detection delay for detecting sensor replay attack II in AEBS

Types	N, δ, τ	False positive	False negative	Average delay (frames)
VAE-based	10, 1, 148	14/100	78/100	38.68
	10, 0, 227	17/100	75/100	38.08
	20, −3, 0	100/100	0/100	0.04
	20, 27, 296	0/100	100/100	N/A
Proposed	10, −2, 64	16/100	4/100	20.90
	10, −2, 72	14/100	4/100	21.17
	20, −2, 57	14/100	0/100	17.68
	20, −1, 35	12/100	3/100	16.33

and therefore, the observation sequence can be denoted as $(x_1, \ldots, x_{t_1}, \ldots, x_{t_1})$.

We plot the p-values and the detector S-values for both two detection approaches in Fig. 6. For the VAE-based method, the p-values are between 0 and 1, and S-values are almost 0 through the whole episode implying that it cannot be used for detection of such a sequence. In contrast, the proposed method can detect such sequence with a very small delay. Once the camera gets stuck, the p-values start to decrease approaching 0, and the S-value increases and exceeds the threshold of the CUSUM detector. We also report the false positives, false negatives, and average detection delay in Table 3. The number of falses for this scenario is larger than the two previous attacks because there is no considerable change in the observations.

6.1.5 Slowdown sensor attack

To have a better understanding of the proposed approach, we adopt an attacker's mindset and design an attack trying to reduce the effectiveness of the approach. We consider a slowdown sensor attack that reduces the transmission speed of the sensor network. In this scenario, at the time step t_1, the transmission speed is reduced by half, that is, the observation sequence can be denoted as $(x_1, \ldots, x_{t_1}, x_{t_1}, x_{t_2}, x_{t_2}, \ldots, x_l, x_l)$, where t_1 is randomly sampled in $\{80, \ldots, 100\}$. From the camera point of view, the vehicle slows down.

By varying the prediction horizon N and CUSUM parameters σ and τ, we report the false positives, false negatives, and average detection delay for the proposed method in Table 4. It can be seen from the table that the the number of false negatives reaches 76 out of 100 test episodes, which demonstrates that the proposed method becomes ineffective against such a slowdown sensor attack.

6.2 Autonomous car racing

We evaluate the approach for detecting controller deception attacks in an autonomous car racing example from OpenAI

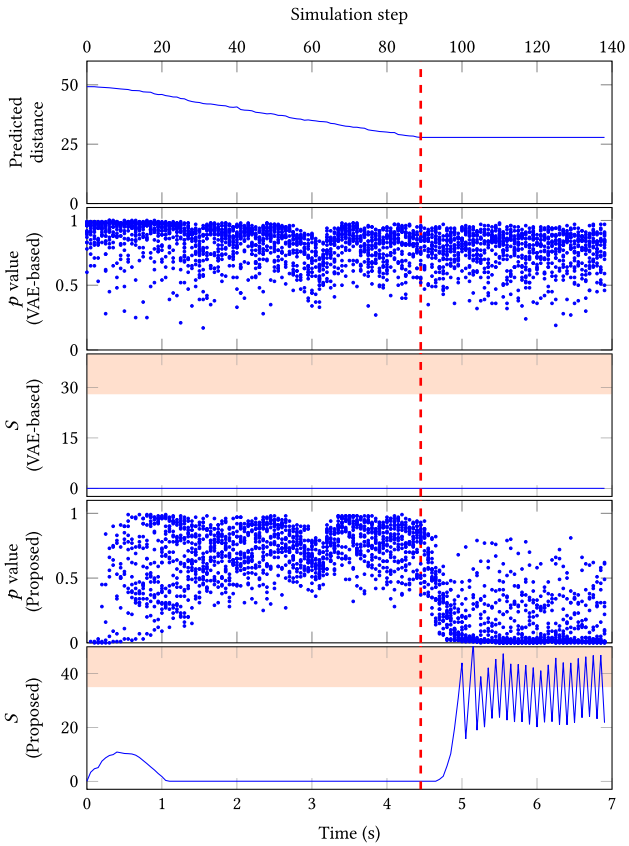

Fig. 6 An episode under the stuck sensor attack in AEBS (detector parameter: (1) VAE-based: $N = 20$, $\delta = 0$, $\tau = 28$; (2) proposed: $N = 20$, $\delta = -1$, $\tau = 35$)

Gym [4]. The task here is to use the pixel inputs from the top-down camera looking at a racing environment to learn a controller for the throttle, brake and steer signal so that the autonomous car follows the track. In [19], a world model is trained to extract the spatial and temporal representation of the autonomous car racing environment, and a reinforcement controller is trained by using the compressed features from the world model. It should be noted that in this example, the control signal along with the latent space representation is used as inputs to the RNN in the generative model to predict

Table 3 False positives, false negatives, and detection delay for detecting stuck sensor attack in AEBS

Types	N, δ, τ	False positive	False negative	Average delay (frames)
VAE-based	10, 12, 210	0/100	100/100	N/A
	10, 23, 4	0/100	100/100	N/A
	20, −3, 98	64/100	25/100	11.70
	20, 0, 28	58/100	30/100	9.73
Proposed	10, −2, 64	16/100	21/100	17.08
	10, −2, 72	14/100	23/100	17.89
	20, −2, 57	14/100	16/100	16.00
	20, −1, 35	12/100	18/100	14.53

Table 4 False positives, false negatives, and detection delay for detecting slowdown sensor attack in AEBS

Types	N, δ, τ	False positive	False negative	Average delay (frames)
Proposed	10, 2, 110	12/100	77/100	26.96
	10, 1, 173	14/100	76/100	23.50
	20, 7, 200	8/100	76/100	26.80
	20, 8, 132	8/100	76/100	24.38

the future states and observations. Therefore, we can evaluate if the approach can detect anomalies caused by the deception attacks on the control signal using only the observation sequence from the top-down camera.

We use the same network architecture and training process as in [19] to train the world model including the controller. Using the trained model, we collect 900 episodes by randomizing the starting position of the car. Each episode contains 1000 frames, and a total of 900, 000 images are collected. Eight hundred of these episodes are used as calibration data, while 100 episodes are used for testing as normal sequences. As discussed in Sect. 5, the control signals must be incorporated in the detection method because they are used in the RNN to predict the future observations.

6.2.1 Experimental results

In order to generate the observation sequences under attack, the original control signal is replaced with a full gas, zero brake, and full opposite steer control signal at a random time step $t \in \{50, \ldots, 79\}$, which will cause the car to drive off the track. We collect 100 episodes that are used for testing positive abnormal sequences. We evaluate both the VAE-based and the proposed method using 100 episodes with normal sequences and 100 episodes with abnormal sequences. The attack parameters are selected to evaluate how fast we can detect an attack with catastrophic consequences.

6.2.2 Experimental setup

We plot the detection results for an abnormal sequence in Fig. 7 for both approaches. In this episode, the attack starts at time step 64, and the car then turns sharply to the left off the track. In Fig. 8, we plot predicted observations and the actual observations for the following 20 steps forward from time step 59. The generative model predicts the car will behave normally and follow the track, but due to the attack, the car will drive off the track. Such abnormal behavior of the car deviates significantly from what is expected by the generative model. Therefore, the deviations between the expected and actual observations become large and can be captured by the proposed method, where the proposed method can cap-

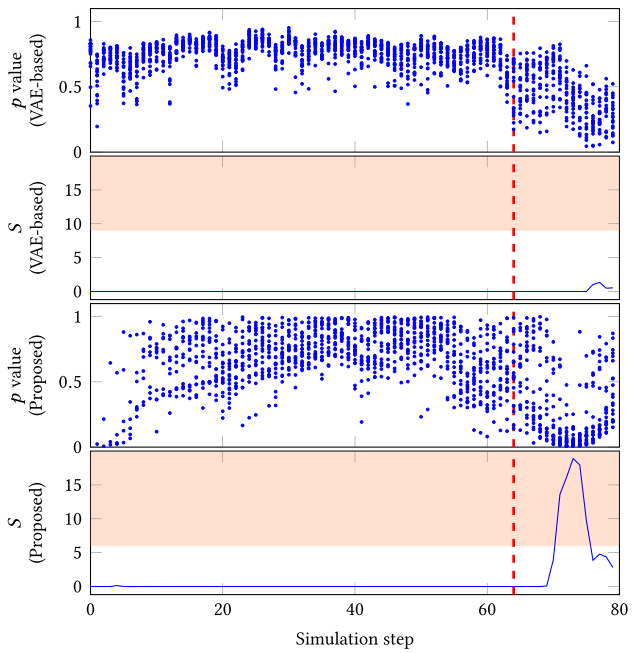

Fig. 7 An episode under the controller deception attacks in autonomous car racing example (detector parameter: (1) VAE-based: $N = 20, \delta = 0, \tau = 9$; (2) proposed: $N = 20, \delta = 0, \tau = 6$)

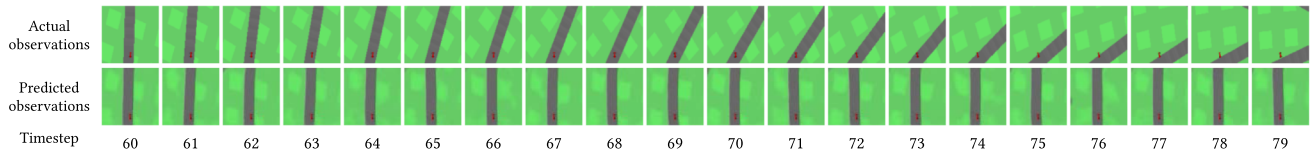

Fig. 8 Comparison between the actual and predicted observations when the control signal is under attack in autonomous car racing example; all the predicted observations are predicted at time step 59

ture such deviations because the p-values become small, and the CUSUM detector generates alarms. Because the observations for individual frames are in the same distribution of the training dataset and can be reconstructed by the VAE model quite well, the test episode will not trigger the alarm of the VAE-based detector. Therefore, the test episode will not trigger the alarm of the VAE-based detector, and the VAE-based detector performs much worse than the proposed method.

We also report the number of false positives, false negatives, and average detection delay in Table 5 by considering different values of the prediction horizon N and CUSUM parameters σ and τ. From the results, we can see that the number of falses is decreasing with larger prediction horizon. Not surprisingly, the VAE-based method performs much worse than the proposed method.

6.3 Secure water treatment dataset

Secure water treatment (SWaT) testbed is a scaled-down water treatment plant primarily developed for research in the area of cyber security [15]. SWaT consists of a modern six-stage filtration process capable of producing five gallons of filtered water per minute. The SWaT dataset is collected under seven days of continuous normal operation and four days with attack scenarios. The data contains values from all 51 sensors and actuators. Forty-one attacks are launched over four days by spoofing the sensor or actuator information to compromise the normal behavior of the plant, which can be regarded as deception attacks on the sensor and actuator networks. We use the SWaT dataset to demonstrate the effectiveness of our approach for detecting deception attacks in a real-world CPS.

6.3.1 Experimental setup

Our evaluation follows the settings in [16] to detect the cyber attacks in Process 1 (P1) of SWaT testbed. The objective of P1 is to manage the inflow and outflow of the raw water tank, where two sensor measurements (FIT-101, LIT-101) and three control signals (MV-101, P-101, and P-102) are involved. The seven-day continuous sequence with a total of 496, 800 samples are segmented into 4, 968 subsequences (100 samples in each subsequence), 3974 of which are used as the proper training dataset and 994 as the calibration dataset.

We should note that only the five values (FIT-101, LIT-101, MV-101, P-101, and P-102) relating to P1 are used in our experiment, and the VAE is omitted because it is unnecessary to compress such a low-dimensional input. An LSTM with three layers and 100 hidden states is used in our experiment, which is same as the RNN described in [16] for fair comparison. More details about the dataset and the experimental settings can be found in [15] and [16], respectively.

6.3.2 Experimental results

Four consecutive days of data with attack scenarios are utilized as the test data and sequentially fed into the detection algorithm. Of all 41 attacks during four days, only ten are involved in the P1 and are used as positive test samples, and the remaining parts in the dataset are used as negatives.

We illustrate the detection process for one of the ten attacks and plot the p-values and detector S-values in Fig. 9. Such an attack occurs at the timestamp "12/28/2015 10:29:14 AM," whose intent is to overflow the water tank by deceiving the controller to open the inflow motor valve when it

Table 5 False positives, false negatives, and detection delay for detecting controller deception attacks in autonomous car racing example

Types	N, δ, τ	False positive	False negative	Average delay (frames)
VAE-based	10, −2, 53	62/100	29/100	11.63
	10, 0, 9	55/100	39/100	9.68
	20, −3, 98	64/100	25/100	11.70
	20, 0, 28	58/100	30/100	9.73
Proposed	10, −2, 33	27/100	12/100	8.82
	10, −1, 15	22/100	20/100	11.49
	20, −3, 73	12/100	13/100	8.75
	20, 0, 6	7/100	18/100	7.87

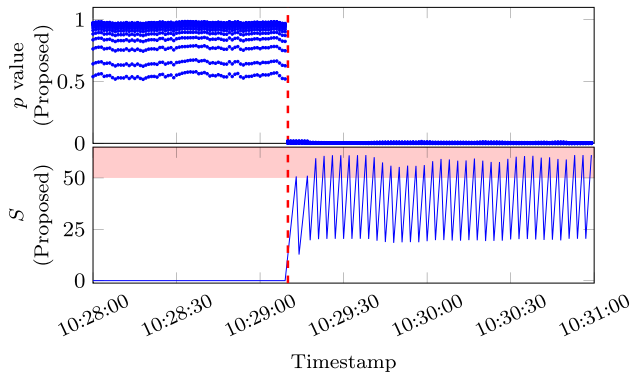

Fig. 9 An episode under the an deception attack in SWaT dataset (detector parameter: Proposed: $N = 20$, $\delta = 30$, $\tau = 50$)

Table 6 False positives and false negatives for detecting deception attacks in SWaT dataset

Types	N, δ, τ	False positive	False negative
Proposed	10, 10, 20	9 in 4 days	0/10
	10, 15, 142	5 in 4 days	2/10
	20, 30, 50	9 in 4 days	0/10
	20, 34, 184	5 in 4 days	2/10

was supposed to be closed. Once the attack is injected, the p-values approach zero instantly and the large detector S-values alarms that the P1 is in an abnormal behavior and may be compromised by an attack.

Table 6 summarizes the results of our approach for identifying attacks in the test sequence by varying the prediction horizon N and CUSUM parameters σ and τ. As can be seen from the table, the proposed approach can detect the deception attacks in P1 with very low false positives and false negatives. Given $N = 20$, $\sigma = 30$, and $\tau = 50$, all 10 attacks are detected and only nine false positives occur within four days. Such a low number of false positives is acceptable concerning four-day run time. Besides, the detection result surpasses that reported in [16], where nine out of the ten attacks are detected, but no specific number of false positives are reported.

6.4 Computational efficiency

In order to characterize the real-time nature of the proposed method, we take the AEBS as an example and measure the execution time of the detection algorithm at each time step of the entire episode. Although the generative model containing the VAE and RNN can be trained offline on a powerful machine, it will be deployed to a real-world CPS and executed on an embedded board during testing. Therefore, we measure execution times not only on a powerful cloud machine, but also on a NVIDIA Jetson TX2 board. Table 7 reports the minimum (min), first quartile (Q_1), second quartile or median (Q_2), third quartile (Q_3), and maximum (max) of the execution times for different values of N on different platforms (cloud machine and Jetson TX2 board). It is reasonable to see that the execution times become longer as N increases, because the generative model needs to predict a longer future when N is larger. The execution times measured on both powerful cloud machines and embedded devices are relatively short, suggesting that our approach can be used for real-time detection.

Table 7 Execution times (mm) for each detection step in AEBS

Platform	N	min	Q_1	Q_2	Q_3	max
Cloud machine	10	65.30	65.83	66.31	66.87	70.50
	20	25.95	26.16	26.27	26.41	30.77
Jetson TX2	10	109.94	127.20	127.98	128.85	131.21
	20	25.95	26.16	26.27	26.41	30.77

7 Conclusions

In this paper, we propose an approach for detecting deception attacks in CPS. The method is based on ICAD framework and utilizes a novel generative model inspired by the world model for efficiently measuring the nonconformity of the high-dimensional observation sequences relative to the normal system behavior, thereby allowing for real-time detection. The evaluation is performed using two simulation case studies of an advanced emergency braking system and an autonomous car racing example, as well as a real-world SWaT dataset.

The evaluation demonstrates the effectiveness of the proposed approach, showing a small number of false alarms in most experiments. We conduct comparative evaluation against a VAE-based method in two simulation case studies. The VAE-based method only considers individual frames and is normally used to detect the anomalies in point-wise. Our approach outperforms the VAE-based method since the proposed approach can capture the temporal dependencies of time-series data. An RNN-based method is also utilized for comparison in SWaT dataset and performs worse than the proposed approach. Furthermore, execution times measured on both power cloud machines and embedded computing devices are very short, enabling real-time detection.

Although the proposed approach can detect attacks with relatively small number of false positives and false negatives, and the false rate is less than 10% in most experiments, it is not enough to guarantee the safety of the system operation. The system should switch to human intervention or enable a backup system when the test is positive. The high false

positive rate will indeed cause frequent intervention by the manual or backup systems, resulting in system instability. Besides, any false negatives or missed attacks can have catastrophic consequences, such as colliding with the obstacle in AEBS or overflowing the water tank in the SWaT testbed. Therefore, reducing the false positives and false negatives of the detector is one of the challenges remaining in our future work.

The effectiveness of the proposed approach can be reduced by well-crafted attacks. Our evaluation in AEBS validates that a slowdown sensor attack can reduce the effectiveness of the proposed detector. In addition, an adversarial example with small specially human-crafted perturbations can cause false prediction of neural network [17], and spoofing the input data with such an adversarial example can be regarded as an deception attack. Although not confirmed, it is very likely that such an attack can break our detection method since a small perturbation added to the input cannot lead to a large nonconformity score. Making the detector effective against a wider range of attacks is also one of the challenges.

The proposed approach can capture the abnormal inputs that deviate significantly from the expected inputs. Therefore, the approach can be extended to detect abnormal behaviors that are not limited to attacks, such as unintended system behavior caused by component failures. Evaluation on broader scenarios can be the future work, especially on realistic applications with high-dimensional inputs.

Funding The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under Grant Numbers CNS 1739328 and the Defense Advanced Research Projects Agency (DARPA) through Contract Number FA8750-18-C-0089. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, or NSF.

Data availability statement The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Anwar, A., Mahmood, A., Ray, B., Mahmud, M.A., Tari, Z.: Machine learning to ensure data integrity in power system topological network database. Electronics **9**(4), 693 (2020)
2. Bengio, Y. , Frasconi, P., Simard, P.Y.: The problem of learning long-term dependencies in recurrent networks. In: Proceedings of International Conference on Neural Networks, ICNN '88 (1993)
3. Bishop, C.: Mixture density networks. Technical Report (1994)
4. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym. CoRR. arXiv:1606.01540 (2016)
5. Cai, F., Koutsoukos, X.D.: Real-time out-of-distribution detection in learning-enabled cyber-physical systems. In: 11th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS (2020)
6. Cai, F., Li, J., Koutsoukos, X.D.: Detecting adversarial examples in learning-enabled cyber-physical systems using variational autoencoder for regression. In: IEEE Security and Privacy Workshops (2020)
7. Cai, F., Ozdagli, A.I., Koutsoukos, X.D.: Detection of dataset shifts in learning-enabled cyber-physical systems using variational autoencoder for regression. In: 4th IEEE International Conference on Industrial Cyber-Physical Systems, ICPS (2021)
8. Cárdenas, A.A., Amin, S., Sastry, S.: Secure control: towards survivable cyber-physical systems. In: 28th IEEE International Conference on Distributed Computing Systems Workshops (2008)
9. Depeweg, S., Hernández-Lobato, J.M., Doshi-Velez, F., Udluft, S.: Learning and policy search in stochastic dynamical systems with Bayesian neural networks. In: 5th International Conference on Learning Representations, ICLR (2017)
10. Dosovitskiy, A., Ros, G., Codevilla, F., López A., Koltun, V.: CARLA: an open urban driving simulator. In: 1st Annual Conference on Robot Learning, CoRL (2017)
11. Fedorova, V., Gammerman, A.J., Nouretdinov, I., Vovk, V.: Plug-in martingales for testing exchangeability on-line. In: Proceedings of the 29th International Conference on Machine Learning, ICML '12 (2012)
12. Feng, Y., Ng, D.J.X., Easwaran, A.: Improving variational autoencoder based out-of-distribution detection for embedded real-time applications. ACM Trans. Embed. Comput. Syst. (TECS) **20**(5s), 1–26 (2021)
13. Ferragut, E.M., Laska, J., Olama, M.M., Ozmen, O.: Real-time cyber-physical false data attack detection in smart grids using neural networks. In: International Conference on Computational Science and Computational Intelligence (CSCI) (2017)
14. Garip, M.T., Gursoy, M.E., Reiher, P., Gerla, M.: Congestion attacks to autonomous cars using vehicular botnets. In: NDSS Workshop on Security of Emerging Networking Technologies (SENT) (2015)
15. Goh, J., Adepu, S., Junejo, K.N., Mathur, A.: A dataset to support research in the design of secure water treatment systems. In: 11th International Conference on Critical Information Infrastructures Security, CRITIS (2016)
16. Goh, J., Adepu, S., Tan, M., Lee, Z.S.: Anomaly detection in cyber physical systems using recurrent neural networks. In: 18th IEEE International Symposium on High Assurance Systems Engineering, HASE '2017 (2017)
17. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: 3rd International Conference on Learning Representations, ICLR (2015)
18. Gu, X., Easwaran, A.: Towards safe machine learning for cps: infer uncertainty from training data. In: 10th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS, pp. 249–258 (2019)
19. Ha, D., Schmidhuber, J.: Recurrent world models facilitate policy evolution. In: Advances in Neural Information Processing Systems, NeurIPS, vol. 31 (2018)
20. Habler, E., Shabtai, A.: Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages. Comput. Secur. **78**, 155–173 (2018)
21. Hoehn, A., Zhang, P.: Detection of replay attacks in cyber-physical systems. In: American Control Conference, ACC, pp. 290–295. IEEE (2016)

22. Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C.M., Sun, J.: Anomaly detection for a water treatment system using unsupervised machine learning. In: IEEE International Conference on Data Mining Workshops (2017)
23. Ishimtsev, V., Bernstein, A., Burnaev, E., Nazarov, I.: Conformal *k*-NN anomaly detector for univariate data streams. In: Proceedings of Machine Learning Research, vol. 60, pp. 213–227. PMLR (2017)
24. Kantaros, Y., Carpenter, T.J., Sridhar, K., Yang, Y., Lee, I., Weimer, J.: Real-time detectors for digital and physical adversarial inputs to perception systems. In: 12th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS (2021)
25. Keahey, K., Anderson, J., Zhen, Z., Riteau, P., Ruth, P., Stanzione, D., Cevik, M., Colleran, J., Gunawi, H.S., Hammock, C., Mambretti, J., Barnes, A., Halbach, F., Rocha, A., Stubbs, J.: Lessons learned from the chameleon testbed. In: USENIX Annual Technical Conference (2020)
26. Khaitan, S.K., McCalley, J.D.: Design techniques and applications of cyberphysical systems: a survey. IEEE Syst. J. **9**(2), 350–365 (2014)
27. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes. In: 2nd International Conference on Learning Representations, ICLR (2014)
28. Kravchik, M., Shabtai, A.: Detecting cyber attacks in industrial control systems using convolutional neural networks. In: Proceedings of the 2018 Workshop on Cyber-physical Systems Security and Privacy (2018)
29. Laxhammar, R., Falkman, G.: Conformal prediction for distribution-independent anomaly detection in streaming vessel data. In: 1st International Workshop on Novel Data Stream Pattern Mining Techniques (2010)
30. Laxhammar, R., Falkman, G.: Inductive conformal anomaly detection for sequential detection of anomalous sub-trajectories. Ann. Math. Artif. Intell. **74**, 67–94 (2015)
31. Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.: MAD-GAN: multivariate anomaly detection for time series data with generative adversarial networks. In: 28th International Conference on Artificial Neural Networks, ICANN '2019 (2019)
32. Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. ACM Trans. Inf. Syst. Secur. **14**(1), 13:1-13:33 (2011)
33. McAllister, R., Rasmussen, C.E.: Data-efficient reinforcement learning in continuous state-action gaussian-pomdps (2017)
34. Mo, Y., Sinopoli, B.: Integrity attacks on cyber-physical systems. In: 1st International Conference on High Confidence Networked Systems, HiCoNS (2012)
35. Mo, Y., Weerakkody, S., Sinopoli, B.: Physical authentication of control systems: designing watermarked control inputs to detect counterfeit sensor outputs. IEEE Control Syst. Mag. **35**(1), 93–109 (2015)
36. Nizam, F., Chaki, S., Al Mamun, S., Kaiser, M.S., et al.: Attack detection and prevention in the cyber physical system. In: International Conference on Computer Communication and Informatics (ICCCI) (2016)
37. Pang, Z.-H., Liu, G., Dong, Z.: Secure networked control systems under denial of service attacks. IFAC Proc. Vol. **44**(1), 8908–8913 (2011)
38. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Technical report, California Univ. San Diego La Jolla Inst for Cognitive Science (1985)
39. Schmidhuber, J.: On learning to think: algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. arXiv preprint arXiv:1511.09249 (2015)
40. Smith, J., Nouretdinov, I., Craddock, R., Offer, C., Gammerman, A.: Anomaly detection of trajectories with kernel density estimation by conformal prediction. In: International Conference on Artificial Intelligence Applications and Innovations, AIAI (2014)
41. Srikantha, P., Kundur, D.: Denial of service attacks and mitigation for stability in cyber-enabled power grid. In: 2015 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT), pp. 1–5. IEEE (2015)
42. Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D.: Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD (2019)
43. Zhou, M., Zhang, Z., Xie, L.: Permutation entropy based detection scheme of replay attacks in industrial cyber-physical systems. J. Frankl. Inst. **358**(7), 4058–4076 (2021)