Digital Twin-based Anomaly Detection in Cyber-physical Systems

Qinghua Xu
Simula Research Laboratory
University of Oslo
Fornebu, Norway
qinghua@simula.no
0000-0001-8104-1645

Shaukat Ali
Simula Research Laboratory
Fornebu, Norway
shaukat@simula.no
0000-0002-9979-3519

Tao Yue
Simula Research Laboratory
Nanjing University of Aeronautics and Astronautics
Nanjing, China
taoyue@ieee.org
0000-0003-3262-5577

***Abstract*—Cyber-Physical Systems (CPS) are susceptible to various anomalies during their operations. Thus, it is important to detect such anomalies. Detecting such anomalies is challenging since it is uncertain when and where anomalies can happen. To this end, we present a novel approach called Anomaly deTection with digiTAl twIN (ATTAIN), which continuously and automatically builds a digital twin with live data obtained from a CPS for anomaly detection. ATTAIN builds a Timed Automaton Machine (TAM) as the digital representation of the CPS, and implements a Generative Adversarial Network (GAN) to detect anomalies. GAN uses a GCN-LSTM-based module as a generator, which can capture temporal and spatial characteristics of the input data and learn to produce realistic unlabeled adversarial samples. TAM labels these adversarial samples, which are then fed into a discriminator along with real labeled samples. After training, the discriminator is capable of distinguishing anomalous data from normal data with a high F1 score. To evaluate our approach, we used three publicly available datasets collected from three CPS testbeds. Evaluation results show that ATTAIN improved the performance of two state-of-art anomaly detection methods by 2.413%, 8.487%, and 5.438% on average on the three datasets, respectively. Moreover, ATTAIN achieved on average 8.39% increase in the anomaly detection capability with digital twins as compared with an approach of not using digital twins.**

***Index Terms*—cyber-physical system, digital twin, machine learning, anomaly detection**

I. Introduction

A Cyber-physical system (CPS) is a complex system consisting of both physical and cyber parts. CPS have become increasingly ubiquitous in critical infrastructures, such as water treatment plants, smart grids, and railways.

Anomaly detection for CPS concerns the identification of unusual behaviors, i.e., anomalies. Such anomalies could result from control elements, network, or physical environment. Also, they may result from hardware and software faults, operator errors, or even misconfigurations of CPS. Multiple attack detection mechanisms have been proposed to tackle these challenges, including techniques based on invariant checking, physical attestation, and fingerprinting. A well-known solution is to build an anomaly detector [1]–[5], in which an underlying machine learning (ML) model is trained on static data from a system to judge when it deviates from its normal behavior. Typically, such a model would be in the form of a neural network, a powerful model to learn and recognize complex patterns of behaviors that CPS exhibit. An anomaly detector's effectiveness can be assessed by subjecting it to a test suite of attacks, and observing whether it can correctly identify anomalous behaviors.

Despite the success of these neural network-based anomaly detectors, most of them still suffer from two major problems. First, most of these models are trained on static log data and cannot keep learning during the operation of CPS. As a result, these models perform best when dealing with known attacks, but suffer from low accuracy when novel attacks occur. Another problem with neural network-based models is that they rely on a large amount of labeled data for training. However, acquiring labels in CPS is expensive generally. Hence, a model that can take advantage of unlabeled data is much more desirable in this situation.

To address the above-mentioned two key problems, we propose a novel approach, called Anomaly deTection with digiTAl twIN (ATTAIN), which takes advantage of unlabeled data and continuously learns during the operation of a CPS i.e., at runtime. Digital twin is an emerging concept consisting of two parts: Digital Twin model and Digital Twin capability. Digital twin model is a virtual replica or live model of the CPS, while digital twin capability is the digital twin's functionality [6]. In our context, the capability of a digital twin is the application of ML algorithms for anomaly detection. We build a digital twin model with historical data and fine-tune it with real-time data, which simulates the CPS with high realism during runtime. We use a Generative Adversarial Network (GAN) to implement the ATTAIN digital twin capability, which can be trained on labeled historical data and unlabeled real-time data with the digital twin model's ground truth label.

We evaluate the effectiveness of our method on three datasets collected from real-world critical infrastructure testbeds: Secure Water Treatment (SWaT) [7]— a multi-stage water purification plant, Water Distribution (WADI) [8]— a consumer distribution network, and Battle of Attack Detection Algorithms (BATADAL) [9]— a dataset designed for an attack detection contest. We demonstrate that our method improves

This work is supported by the security project funded by Norwegian Ministry of Education and Research, and Horizon2020 project ADEPTNESS, Grant Agreement Number: 871319.

the performance of two state-of-the-art anomaly detection methods by 2.413%, 8.487%, and 5.438% on average on SWAT, WADI, and BATADAL, respectively.

The remaining part of this paper is organized as follows. We present background on digital twins in Section II. Section III presents a running example of CPS operation and an attack against actuators. Section IV introduces our proposed ATTAIN in details. Section V shows the design of our empirical evaluation, followed by Section VI where we present the experimental results and analysis. Section VII shows existing research related to our work. Finally, Section VIII summarizes the paper and proposes possible future work.

II. Background

El Saddik [10] defined a concept of *Digital Twin* as *"a digital replica of a living or non-living physical entity"*. In this paper, we focus on building a digital twin of a CPS. Fig 1 shows a high-level view of a digital twin for a CPS in operation, which is referred to as Physical Twin - a commonly used term in the literature. Data from the physical twin is continuously fed to the digital twin. Depending on the CPS, such data may come from its environment, communication medium, and the CPS itself. A digital twin may also be able to take actions on the physical twin. Examples of such actions include preventing unsafe situations or alerting a CPS operator about abnormal behaviors of the CPS. At some situations, one may not be able to intervene in the operation of the CPS.

As shown in Fig 1, a digital twin consists of two parts, i.e., digital twin model and digital twin capability. The *digital twin model* refers to the digital representation of the CPS. This representation could be in the form of heterogeneous models (e.g., software, hardware, communication models). These models can be constructed in different ways. In some instances, we may have CPS design models that could be used and improved incrementally during the operational digital twin based on live data, thereby representing the most up-to-date state of the underlying CPS. In this paper, we build such models initially from historical data followed by the continuous improvement of them with live data in our context. The *digital twin capability* refers to the functionality of a digital twin. Depending on the context, a digital twin can provide various capabilities such as predictions of non-functional properties, uncertainty detection, and failure prevention. In our context, we focus on the anomaly detection capability that detects anomalies during the operation of a CPS. The digital twin capability can interact with the digital twin model, e.g., to perform simulations on it. Similarly, the digital twin model can interact with the capability, e.g., to get feedback. This bidirectional relationship is shown as a dotted arrow between the digital twin model and the capability in Fig 1. In this paper, a digital twin model provides ground truth labels to improve the anomaly detection capability of ATTAIN.

III. Running Example from the SWaT Testbed

To better illustrate our method, we present a running example in Table I. In this table, we use sensor and actuator data

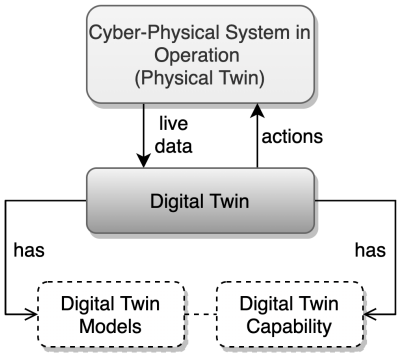

Fig. 1. Digital Twin for Cyber-Physical System

from the first stage of the SWaT testbed. Sensors include Flow Indicator Transmitter (FIT101) and Level Indicator Transmitter (LIT101), while actuators consist of Pumps (P101, P102), and Moving Valve (MV101). Sensor values are continuous, and actuator values are discrete (0 for opening; 1 for opened; 2 for closed). Formally, we use u to denote a observations in a time point, consisting of sensor and actuator values, as shown in Equation 1

$$u = [u_{s1}, u_{s2}, u_{s3}, ...u_{a1}, u_{a2}, u_{a3}, ...] \quad (1)$$

where u_{sj} represents a value for the j^{th} sensor and u_{ak} represents the value for the k^{th} actuator. We define u^i to be the system state at the i^{th} time point, and U^i to be the sequence of states before the i^{th} time point.

An attack against MV101 is also shown in this table. The system starts at 10:00:00, pumping water from outside into the tank. As the water level (LIT101) in the tank grows above the upper limit, MV101 should be closed to prevent a water overflow. However, an attacker forces MV101 to be opened regardless of the readings from LIT101. This attack starts at 10:29:14 and ends at 10:44:53, causing real water overflow damages to the plant. Our goal in this paper is to detect this attack in advance, before it causes actual damages, with the help of U^i.

IV. Approach

In Section IV-A, we provide the overview of ATTAIN, followed by the discussions on the digital twin model generation (Section IV-B and the digital twin capability (Section IV-C).

TABLE I
Running Example

Timestamp	FIT101	LIT101	MV101	P101	P102	Label
10:00:00	2.43	522.84	2	2	1	Normal
10:00:01	2.45	522.88	2	2	1	Normal
...	...	...	...	...	...	...
10:29:13	2.44	816.84	2	1	1	Normal
10:29:14	2.49	817.67	2	1	1	Attack
10:29:15	2.54	817.94	2	1	1	Attack
...	...	...	...	...	...	...
10:44:53	6e-4	869.72	1	2	1	Attack

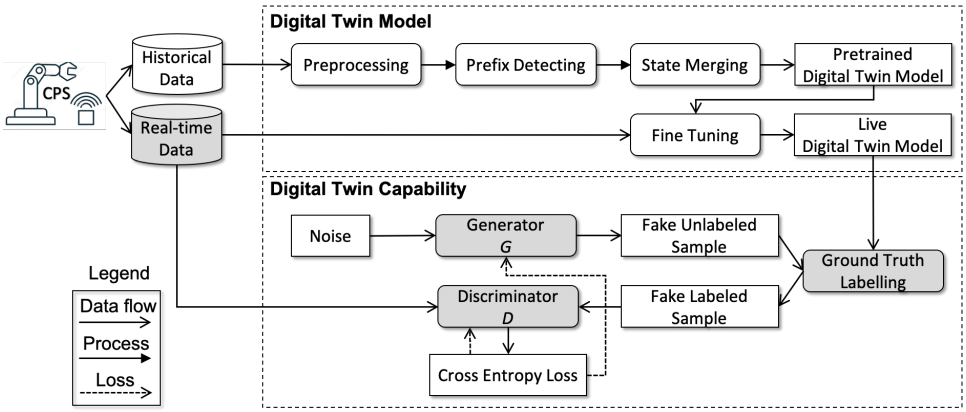

Fig. 2. Overview of ATTAIN: Anomaly deTection with digiTAl twIN.

A. Overview

We propose a novel approach called Anomaly deTection with digiTAl twIN (ATTAIN), which continuously learns from new data and improves its anomaly detection performance using a mix of labeled and unlabeled adversarial data. ATTAIN has two different segments: *Digital Twin Model* and *Digital Twin Capability*.

Fig 2 shows the overview diagram of our proposed method ATTAIN. Two types of data need to be collected before training: sensor and actuator values. Data is acquired both from the past, i.e., *Historical Data*) and in real-time (i.e., *Real-time Data*) from an operational CPS.

Learning digital twin models is a standard state prediction process based on historical and real-time data. It is initially learned as a Timed Automaton Machine from the historical data statically. Though this pre-trained digital twin model tends to simulate the real CPS with high realism, it is limited to capturing only known behaviors. Therefore, this model needs to be further improved with real-time data, allowing it to evolve along with its physical counterpart at runtime. Both the pre-training and online training processes utilize the OTALA [11] algorithm, a multi-stage algorithm with the following steps: pre-processing, prefix detection, state merging. We extend this algorithm to learn probabilistic real-time automaton.

The digital twin capability, i.e., the anomaly detector, in this case, is trained on real-time data only. Generative Adversarial Network (GAN) is used as the backbone framework in our anomaly detector. GAN consists of a generator and a discriminator. The generator learns to generate adversarial samples, mapping from a latent space to the distribution of input data. The discriminator learns to distinguish among real attack, real normal, adversarial normal, and adversarial attack samples. The anomaly detector is trained in a supervised learning fashion, yet unlabeled adversarial samples are also utilized. To that end, our method calculates a ground truth label as training signals by comparing real sensor and actuator values and predicted sensor and actuator values generated by the digital twin model. This ground truth label is then used for the cross entropy loss calculation. Backpropagation is applied in both the generator and discriminator such that the generator produces better adversarial samples, while the discriminator becomes more skilled at flagging samples from different categories.

B. Digital Twin Model Generation

A digital twin model is often a behavioral model. Such a model is usually represented as a statechart or a finite automaton, and mostly created manually by domain experts [12], [13]. However, there exist approaches on the automated construction of Deterministic Finite automaton (DFA) such as the ones reported in [11], [14]. Along with this line, we propose to use Timed Automaton to represent our automatically generated digital twin model. In the automaton theory, a timed automaton is a finite automaton extended with a finite set of real-valued clocks, which can better model real-time systems. Below, in Section IV-B1 we introduce the formalism of Timed Automaton, and in Section IV-B2 we discuss the Online Timed Automaton Learning Algorithm (OTALA) in details [11].

1) Timed Automaton Formalism: A Timed Automaton is a tuple $A = (U, T, \delta)$, where

- U is a finite set of states. In our context, each state $u \subseteq U$ is defined as an observation in a time point, which is a vector consisting of sensor values and actuator values $u = [u_{s1}, u_{s2}, u_{s3}, ...u_{a1}, u_{a2}, u_{a3}, ...]$.
- T is a finite set of transitions, where $T \subseteq U \times U$. For example, for a transition $< u, u' >$, $u, u' \in U$ are the source and destination states.
- δ is a transition timing constraint $\delta : T \rightarrow I$, where I is a set of probability distribution functions with regard to time, which denotes how much time is needed for a certain transition.

2) Learning Algorithm: OTALA was introduced by Maier et. al. [11] as a generic algorithm. We extend it for our purpose. According to Maier's definition, a transition is triggered by events and finished when the timing constraint clock runs up. However, in the case of CPS, events are not observable and transitions are not deterministic. Therefore, we extend OTALA by dropping the concept of events and introducing probability distribution functions as time constraints. Our method described in Algorithm 2 works as follows: For each

Algorithm 1: Overview Algorithm

Input: Historical Observations $H = h_0, ..., h_{n-1}$ and Real-time Observation $U = u_0, ..., u_{n-1}$ where h_i and u_i are IO vectors

Output: Digital Twin Model Timed Automaton A, Digital Twin Capability Anomaly Detector C

```
   /* pretraining digital twin model                                    */
1  A=TimedAutomaton(); C=GAN();
2  for all h_i ∈ H do
3  |  prefixDetection(A,h_i);
4  |  stateMerging(A,h_i);
5  end
   /* online training                                                   */
6  for all u_j ∈ U do
   |  // refine digital twin model & calculate ground truth of attack or not
7  |  prefixDetection(A,u_j);
8  |  stateMerging(A,u_j);
9  |  DTMPredictedActuatorValues=A.predict(u_j);
10 |  realAcutatorValues=subset(u_j);
11 |  distance=hammingDistance(DTMPredictedActuatorValues,realActuatorValues);
12 |  isAttackGroundTruth=True if distance > threshhold else False;
   |  // train anomaly detector
13 |  noise=uncertaintyGenerator(u_j,randomSeed);
14 |  adversarialSample=C.GANGenerator(noise);
15 |  GANPredictedProbabilty=C.GANDiscriminator(concat(u_j,adversarialSample));
16 |  if isAttackGroundTruth then
17 |  |  groundTruthLabel=[[0,1,0],[0,0,1]];
18 |  else
19 |  |  groundTruthLabel=[[1,0,0],[0,0,1]];
20 |  end
21 |  loss=CrossEntropyLoss(groundTruthLabel,GANPredictedProbabilty);
22 |  loss.backpropagation();
23 |  updateParameters(C);
24 end
```

real-time observations, we first perform a prefix detection to find if it has been observed before (lines 2 - 9). If such an observation is observed before, it is checked if the current transition exists (line 4). A new transition is added or existing transition information is updated (lines 7 - 8) accordingly. Otherwise, a new state and new transition are added, while time constraints are updated (line 13). After each step, it is checked whether the learning converged to the final automaton (lines 18 - 20) and if it is converged, the identified automaton is returned. The learning is considered to be converged if the number of states, number of transitions, and timing constraints remain unchanged for a certain period of time.

C. Digital Twin Capability

The core part of our proposed approach ATTAIN is a Generative Adversarial Network (GAN). GAN's capability of producing adversarial samples significantly increases the volume of training data. It consists of two independent models: generator (G) and discriminator (D). Considering an input noise variable z, the goal of G is to generate a new adversarial sample $G(z)$ that comes from the same distribution as of u. On the other hand, the discriminator model (D) returns the probability $D(u)$ that the given sample u is from real data set rather than generated by G. The ultimate goal of G is to maximize the probability that D would mistakenly predict generated data as a real one, and for D the goal is to do the opposite. Thus, G learns to generate more realistic samples while D continuously improves its capability of distinguishing adversarial samples from real samples. Here we modify this vanilla GAN for our purpose. We calculate a ground truth label with the help of digital twin model and denote each sample with more fine-grained labels: real normal, real attack, adversarial normal, and adversarial attack. In this case, the discriminator is a 4-category classifier, which is trained with much more data compared to most existing anomaly detectors.

1) Generator: As introduced in the previous section, sensor and actuator values in a CPS are time-series scalar data. Therefore, for the generator, we use an LSTM and a GCN module to capture temporal and spatial characteristics of the input data, respectively.

Let $u[i]$ be the ith sensor or actuator values of the system. We consider values within a predefined time window as

Algorithm 2: OTALA Algorithm (Σ, U)

Input: Real-time Observations $U = u_0, ..., u_{n-1}$ where u_i is vector of sensor and actuator values
Output: Timed Automaton A

```
while newObservationExists() do
    /* prefix detection                          */
    for all s ∈ S do
        if u(t)==u(s) then
            stateExists=True if transitionExists() then
                updateTransition(t);
                updateTimeConstraint(t);
            else
                createNewTransition(currentState, t(j),s);
                updateTimeConstraint(currentState);
            end
        end
        currentState=s
    end
    /* state merging                             */
    if !stateExists then
        snew = createNewState()
        createNewTransition(currentState, t(j), snew)
        currentState = snew
    end
    if learningConverged() then
        return A
    end
end
return A
```

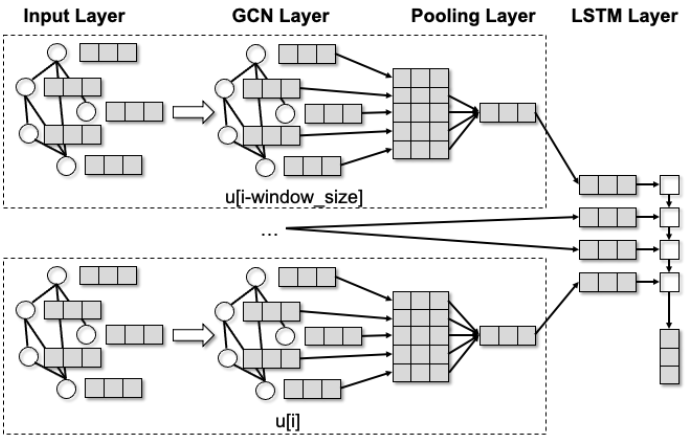

Fig. 3. Model Structure of the GAN Generator

input, e.g., $u[i], u[i-1], ..., u[i-window\_size]$. The overview structure of our generator is shown in Fig 3.

To get a spatial representation $spatial[i]$ for each time step, first, the input is fed into a multi-layer network, consisting of an *Input Layer*, a *GCN Layer*, and a *Pooling Layer*. Second, the spatial representation of all time steps within the window is fed into an *LSTM module* to learn temporal characteristics. The final output of the generator is adversarial samples $fake[i]$ containing both temporal and spatial characteristics.

- **Input Layer.** Let $u[i]$ be the raw input values in the ith time point, consisting of actuator values $a[i]$ and sensor values $s[i]$, which are discrete values and continuous values, respectively. For discrete values, we encode them into one-hot vectors as shown in Equation 2, while for continuous values, we expand them to 3-dimensional vectors, adding their upper limits and lower limits as two additional dimensions as shown in Equation 3. $u[i]$ is made up by concatenating $a[i]$ and $s[i]$ as in Equation 4

$$a'[i] = oneHot(a[i]) \tag{2}$$

$$s'[i] = concat(s[i], upper\_limit, lower\_limit) \tag{3}$$

$$u'[i] = concat(a'[i], s'[i]) \tag{4}$$

- **GCN Layer.** GCN Layer captures interdependent relationships among sensors and actuators. Unlike CNN, GCN takes a specification graph as well as sensor and actuator values as input. In this graph, each actuator and sensor is viewed as a node, while edges are drawn where there is a connection between these two nodes according to the CPS process. For example, in the SWAT testbed, P101 is connected to LIT101 because P101 changes its value based on the readings of LIT101. Let g be the graph we acquired from the CPS specification, which contains a set of edges E and a set of nodes V as in Equation 5. GCN takes graph g and vector $u'[i]$ as input as shown in Equation 6.

$$g = (E, V) \tag{5}$$

$$gcn_{out}[i] = GCN(g, u'[i]) \tag{6}$$

- **Pooling Layer.** In the pooling layer, all the GCN outputs collapse into one vector, as shown in Equation 7 below:

$$spatial[i] = maxPooling(gcn_{out}) \tag{7}$$

where $spatial[i]$ is the spatial representation vector of timestep i.

- **LSTM Layer.** In the LSTM Layer, spatial representations from different time steps are concatenated together as input as shown in Equation 8. LSTM learns temporal features by calculating hidden states between each time step as in Equation 9. The last hidden state $h[i]$ is used as our final representation $fake[i]$ for time step i as shown in Equation 10.

$$input_w = concat(spatial[i : i - window\_size]) \tag{8}$$

$$h[i] = LSTM(input_w) \tag{9}$$

$$fake[i] = h[i] \tag{10}$$

2) Discriminator: The discriminator distinguishes input from adversarial normal, adversarial attack, real attack, and real normal. It is a 4-category classification task. We first transform the input to the output space with linear transformation followed by applying a *tanh* activation function. After that, we calculate a ground truth label using the predicted sensor and actuator values from the digital twin model. With this ground truth label, we calculate cross entropy loss as training signals for our model. Cross Entropy loss is commonly used loss function for classification tasks [15]. It works by penalizing high confident false predictions, increasing the model's generalization ability.

- **Transformation.** This linear layer takes input from both generators and real samples. This layer applies a linear transformation and an activation function consecutively as shown in Equation 11, 12 and 13.

$$input[i] = concat(real[i], fake[i]) \tag{11}$$

$$linear_{out}[i] = Linear(input[i]) \tag{12}$$

$$out[i] = tanh(linear_{out}[i]) \tag{13}$$

- **Ground Truth Calculation**. We calculate the ground truth for every adversarial sample from the generator, and further label it as a adversarial attack or adversarial normal data. We first use digital twin model to predict the current state as in Equation 14. With predicted and real state vector, we then calculate the hamming distance, which is compared with the threshold value δ afterward as shown in Equation 15. For each sample j from the input, we assign a label indicating whether or not it is a normal state as in Equation 16.

$$\hat{a}[i] = DigitalTwinModel(fake[i]) \tag{14}$$

$$d = hamming(\hat{a}[i], a[i]) \tag{15}$$

$$label[j] = \begin{cases} [1,0,0,0], & \text{if sample } j \text{ is fake \& } d > \delta \\ [0,1,0,0], & \text{if sample } j \text{ is fake \& } d <= \delta \\ [0,0,1,0], & \text{if sample } j \text{ is real normal} \\ [0,0,0,1], & \text{if sample } j \text{ is real attack} \end{cases} \tag{16}$$

- **Classification Layer**. We use cross entropy loss for the training as in:

$$likelihood[i] = softmax(out[i]) \tag{17}$$

$$loss = CrossEntropyLoss(label[i], likelihood[i]) \tag{18}$$

V. Empirical Evaluation Design

Section V-A presents the three research questions that we would like to answer. Section V-B presents the case studies we used for experimentation, followed by evaluation metrics (Section V-C). Section V-D provides parameter settings of the experiments, and Section V-E details the experiment execution process.

A. Research Questions

In our experiment, we are interested in answering the following three research questions (RQs).

- **RQ1:** How effective is our anomaly detector as compared to the literature?
- **RQ2:** How realistic is our digital twin model?
- **RQ3:** Is the mechanism of using digital twin effective in detecting anomalies as compared to not using it?

With RQ1, we aim to compare the performance of our approach with existing approaches from the literature. RQ2 is important to answer since a digital twin model is a timed automation machine, aiming at realistically simulating its physical counterpart. Intuitively, the more realistic this model is, the better performance the digital twin capability will demonstrate. Given that we are using the digital twin model to improve anomaly detection, thus, it is important that we assess whether using the digital twin model payoffs at the end and thus we define RQ3.

B. Case Studies

We evaluate ATTAIN with three CPS case studies, namely Secure Water Treatment (SWaT) [7], Water Distribution (WADI) [8], and Battle Of The Attack Detection Algorithms (BATADAL) [9].

SWaT is a CPS testbed for water treatment. Its main functionality is about producing filtered water through a series of stages. The testbed consists of 25 sensors and 26 actuators. The SWaT dataset was produced from the testbed during its operation. The dataset has 946722 samples, each of which has 51 attributes, i.e., sensor measurements and actuator states. Each sample was labeled with *normal* or *attack*.

The second case study is the WADI data produced from the WADI testbed – an extension to the SWaT testbed. The WADI testbed consists of 42 sensors and 61 actuators. The dataset has 1221372 samples, each of which has 103 attributes, i.e., sensor measurements and actuator states. The main functionality of the WADI testbed is about the secure distribution of water. The dataset generated from the WADI testbed has two week's normal operation data, whereas it has two days' attacks data.

The third case study is the BATADAL dataset, which is an extension of the WADI dataset and the attacks were designed for an attack detection competition. The dataset consists of 26 sensors and 17 actuators and has 12938 samples in total.

C. Evaluation Metrics

Following the similar research on anomaly detection [16]–[18], we adopt the three commonly used evaluation metrics, namely precision, recall, and F_1 score, to evaluate our anomaly detector. These metrics are generally considered to be better than other metrics such as accuracy when dealing with imbalanced datasets like the three datasets used in our experiments. Precision is defined in Equation 19

$$precision = \frac{TP}{TP + FP} \tag{19}$$

where TP and FP stand for True Positive and False Positive, respectively. The recall is defined in Equation 20

$$recall = \frac{TP}{TP + FN} \tag{20}$$

where FN denotes False Negative, and F_1 is defined in Equation 21

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{21}$$

To answer RQ2 (aiming to evaluate how realistic our digital twin model is), we consider two metrics:

- **Hamming Distance.** Hamming distance is commonly used for comparing vectors with equal length. The higher the hamming distance between predicted and actual actuator values is, the more realistic our digital twin model is.
- **Outlier Detection Accuracy.** Outlier Detection Accuracy is derived from the hamming distance and it demonstrates the realism of the digital twin model by showing its capability in downstream tasks such as outlier detection. We compare the hamming distance between predicted and actual values at a certain time point with a given threshold. If the hamming distance is larger than the threshold value, this sample is considered to be an outlier. We then calculate the accuracy of outlier detection. For an ideal digital twin model, the predicted values and real values should be identical. In this case, the hamming distance is zero and the outlier detection accuracy is 100%.

(1) hamming distance between predicted and actual actuator values, and (2) outlier detection accuracy. Hamming distance is commonly used for comparing vectors with equal length. The higher the hamming distance between predicted and actual actuator values is, the more accurate that our digital twin model is. Another metric for evaluation is the outlier detection accuracy, which is defined by comparing the hamming distance with a predefined threshold to detect outliers.

D. Parameter Settings

To find optimal parameter settings, we split each dataset into 10 chunks sequentially. The last chunk was used as the held-out testing data, while the first 9 chunks were used as the training data (i.e., training dataset). We performed a 9-fold cross validation on the training dataset with F1 score as an evaluation metric. We set the hyperparameters in our model based on the validation result. We trained the GAN with a batch size of 10 (1 from real-time data, 9 from historical data). The hidden dimension of the neural network was set as 100, and Rectified Linear Unit (ReLU) was used as the activation function. As for the GCN layer, we used a gated GCN module and set the number of layers to 2.

E. Experiment Execution

In this paper, neural network layers were built with the PyTorch framework [19], and the GCN layer was constructed with the PyTorch Geometric (PyG) framework [20]. All the experiments were carried out on Google collaboratory notebooks, with Intel(R) Xeon(R) CPU at 2.00GHz, 12 GB system memory and for GPU, Tesla V100-SXM2 with 16GB memory. To eliminate the effect of randomness, we repeated all the experiments 10 times, and the average results were reported in this paper. For the SWaT dataset, we followed previous works [17] by ignoring the first 16000 records because the state of the CPS tends to be highly unstable during that period of time after it is started.

VI. Results and Analysis

Section VI-A, Section VI-B, and Section VI-C present the results for RQ1, RQ2, and RQ3 respectively. The overall discussion is presented in Section VI-D and the threats to validity in Section VI-E.

A. Results and Analysis for RQ1

Table II presents the comparison of ATTAIN with the state-of-the-art approaches, i.e., the LSTM-CUSUM anomaly detector [21] and the MAD-GAN anomaly detector [22], with respect to precision, recall, and F1 score. Both LSTM-CUSUM and MAD-GAN are designed for CPS anomaly detection and more details about their models can be found in the related work section. We can observe from Table II that ATTAIN outperforms LSTM-CUSUM and MAD-GAN for almost all metrics on all the three datasets. For SWaT, MAD-GAN achieved a slightly better precision, while ATTAIN outperforms it in recall and F1 score by 5.003% and 2.413%, respectively. For the WADI and BATADAL datasets, MAD-GAN has the highest recall. However, in terms of the precision and F1 score, ATTAIN outperformed the other two methods by a large margin, at least 5.184% and 5.438%, respectively, on the WADI and BATADAL datasets.

B. Results and Analysis for RQ2

Figure 4 reveals changes in the hamming distance during the training of the digital twin model, which measures its realism. It is clearly shown in the figure that the hamming distance decreases by a large margin after the digital twin model has been trained with 80,000 and more samples, reaching around 0.5 at the end. After the training of the digital twin model is converged, we define an outlier detector to demonstrate its outlier detection accuracy. A sample is considered to be an outlier or an anomaly if the hamming distance between the real value and a predicted value exceeds a threshold (e.g., 0.5 in our experiment). This threshold value is chosen by experimenting on the validation datasets. We calculated the outlier detection accuracy using labels provided by the datasets, and the results on SWaT, WADI, and BATADAL are 0.82, 0.69, and 0.74, respectively. Considering that our current outlier detector is a very simple model and we did not train the digital twin model for outlier detection, achieving this level of accuracy is already encouraging. In the future, we plan to devise a better outlier detector and train the digital twin model to further improve the outlier detection accuracy.

TABLE II
EXPERIMENT RESULTS OF COMPARING ATTAIN WITH THE STATE-OF-THE-ART APPROACHES (RQ1 AND RQ2)

	SWaT			WADI			BATADAL		
Model	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
LSTM-CUSUM	0.90701	0.67721	0.77544	0.61402	0.69763	0.65959	0.65773	0.72105	0.68794
MAD-GAN	0.96136	0.94228	0.95172	0.43212	0.95274	0.59456	0.52991	0.96213	0.68342
ATTAIN(without signals)	0.92210	0.95402	0.93779	0.52473	0.78219	0.62784	0.55318	0.77451	0.64540
ATTAIN	0.95994	0.99231	0.97585	0.66586	0.84411	0.74446	0.72238	0.76341	0.74232

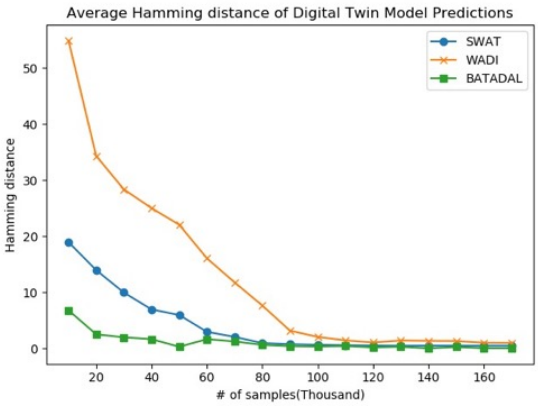

Fig. 4. Hamming distances of the Digital Twin Models (RQ3)

C. Results and Analysis for RQ3

In ATTAIN, signals from the digital twin model are used to calculate a ground truth label for the digital twin capability, and in turn the ground truth label is used to guide the training process of GAN (Fig. 2). To prove that this ground truth label is critical in the training, we conducted experiments to compare the methods of ATTAIN with signals and the ATTAIN without signals from the digital twin model. Without these signals, ATTAIN becomes a vanilla GAN model, i.e., the simplest model to distinguish adversarial samples from the real ones. To get optimal parameters, we re-tuned the GAN model. As a result, the batch size is changed to 20, while the hidden size decreases from 100 to 32. As shown in Table II, the fifth and sixth rows denote the results of ATTAIN with and without signals from the digital twin model, respectively. ATTAIN with signals from the digital twin model improves the F1 score by more than 10% on the SWaT and BATADAL datasets when compared with ATTAIN without signals. We remark that for the BATADAL dataset, the number of normal samples and attacks samples are both the lowest among the three datasets, making data augmentation critical in improving the performance of the detectors. However, only our ATTAIN with signals can increase the data volume by a large margin. Therefore, the poor performance of ATTAIN without signals is reasonable. The comparison shows that signals from the digital twin model are vital in improving the overall performance of ATTAIN.

D. Discussion

ATTAIN outperforms the state-of-art models on the three benchmark datasets for most of the cases. We argue that we achieve this good performance due to the following reasons.

First, capturing temporal and spatial features of CPS is critical in making predictions. Many LSTM-based models were proposed to learn temporal characteristics of CPS, while several CNN-based models have been presented to learn spatial features [23]. CNN was initially designed to learn potential features of images, which can be considered as functions in the Euclidean space. However, actuators and sensors are connected in a Non-Euclidean fashion, contradicting the nature of CNN. Therefore, we use GCN instead of CNN, since GCN is known for its capability of capturing Non-Euclidean structures.

Second, ATTAIN achieved good performance since we used GAN for data augmentation. Existing benchmarks suffer from the sparsity of attack data. However, manually acquiring labeled data, especially attack data in the CPS security domain is expensive. We mitigate this problem by using a GAN framework, thereby increasing the number of attack data by generating our own attack data. A newly expanded dataset with ATTAIN improves its performance by a large margin.

Third, we design our model to be able to learn continuously at runtime. Existing effort mainly focuses on developing new models to improve detecting performance. However, the deployment of these methods has not been studied yet. In particular, there is barely any mechanism for updating trained models to prevent new attacks. However, for a deployed CPS, real-time data can be obtained all the time during its operation. Therefore, we design the digital twin model of ATTAIN as an online learning model, taking full advantage of newly obtained real-time data. Otherwise, a CPS would be vulnerable to attacks that were not included in the training dataset.

E. Threats to Validity

We identify the following threats to the validity of our experiments. First, our experiments were performed on three testbeds scaled down from real operating CPS. Without conducting experiments with real CPS, we cannot draw a sound conclusion on the performance of ATTAIN. Similarly, at this stage, more experiments are definitely required to make any claim on generalizing ATTAIN to other CPS. However, we would like to point out that, in our context, to conduct experiments with real CPS, they need to be deployed, as ATTAIN needs to get connected to them during their operations and obtain their operating data at runtime. Setting up this kind of experiment is complicated, which requires simulators, setting

up communications between a digital twin (i.e., ATTAIN) and its corresponding CPS, etc.

Second, same as for other existing research works (e.g., [21][[17]]), in our experiments, we only kept the stable data of the SWaT dataset. However, attacks are possible to take place during the starting stage of a CPS. Ignoring this part of the data renders the CPS vulnerable at the start, and it is likely that ATTAIN would produce more false-positives at the starting phase of a CPS. In the future, ATTAIN needs to be extended, by for instance developing a strategy particularly for detecting anomalies of the starting phase of a CPS, and then switching to the current implemented strategy.

Third, currently ATTAIN adopted several commonly used techniques such as ReLU, for activation function and max pooling for dimension reduction. There are opportunities to further improve the performance of ATTAIN with alternative techniques such as leaky ReLU [24] and average pooling [25]. In the future, we will design dedicated experiments to assess the effect of alternative techniques on the performance of ATTAIN. Moreover, we adopted the hamming distance as the metric for evaluating the realism of the digital twin model. However, we are aware of other metrics such as the Levenshtein distance [26]. In the future, we will experiment with different metrics for measuring the quality of digital twin models.

Fourth, in ATTAIN, the digital twin model is represented with Timed Automaton machine. We are aware that there are other methods that can be used to represent digital twin models of CPS, such as the Dependent Markov Chain [27]. When needed, in the future, we will pursue opportunities to equip ATTAIN with different digital twin modeling solutions, and evaluate their cost-effectiveness in terms of supporting the digital twin capability of ATTAIN.

VII. Related Work

We present the related work from three aspects: anomaly detection for CPS (Section VII-A, digital twin enabled anomaly detection (Section VII-B), and GAN-based anomaly detection (Section VII-C).

A. CPS Anomaly Detection

Anomaly detection is a popular research topic in the domain of CPS security. There exist several techniques for detecting anomalies in CPS. As reported in [28], methods based on rules (e.g., frequency limit), state estimation (e.g., the Kalman filter), and statistical models (e.g., Gaussian model, histogram-based model) have been proposed to learn normal states of CPS [29]. However, these methods usually require domain knowledge of experts, or need to know underlying normal data distributions. Machine learning approaches do not rely on domain-specific knowledge. However, they usually require a large quantity of labeled data, e.g., for classification-based methods. Also, such labeled data cannot capture properties such as spatial-temporal correlation [23], which are specific to CPS.

To this end, deep learning-based anomaly detection methods have been proposed to identify anomalies in CPS, by exploring various neural network architectures to detect attacks in different CPS domains [29]. Jonathan et al. [21] introduced Long Short Term Memory Networks (LSTM) to capture temporal characteristics of time series data. LSTM was used as a predictor to model the normal behavior of the CPS and subsequently, CUSUM was used to identify abnormal behaviors. Several approaches (e.g., [16]–[18]) adopt a convolutional layer as the first layer of a neural network to obtain correlations of multiple sensors in a sliding time window. Further, extracted features are fed to subsequent layers to generate output scores. Canizo et al. [16] and Wu et al. [30] used RNN to take the output of the CNN layer and form the prediction layer. Moreover, both methods used datasets from real industrial plants and applied precision, recall, F1, and ROC as the evaluation metrics. We follow this research line and use LSTM to capture temporal characteristics of CPS data, while utilizing GCN instead of CNN to better capture their spatial characteristics.

B. Digital Twin based Anomaly Detection

Rubio et al. [31] published a survey on anomaly detection techniques and discussed the trend of future industry, clearly stating that digital twins open up new opportunities to this field. To the best of our knowledge, only a few papers have been proposed to use digital twins for anomaly detection, which are discussed below.

Eckhart & Ekelhart [10] built a knowledge-based intrusion detection system with digital twins, which is based on the assumption that a CPS would exhibit certain unusual behavior patterns during an attack. To this end, they proposed rules that the system must adhere to under normal conditions. Based on these rules, they built a simple digital twin, which continuously checks rule violations at runtime. This method achieved a low false-positive rate in anomaly detection tasks. However, these rules need to be predefined and do not evolve or get updated when additional real-time data available from operating systems as ATTAIN does, because their digital twin is simply a static representation of a real system, implying that the digital twin does not evolve/learn when new data are available.

Later on, Eckhart & Ekelhart [32] further improved their digital twin by introducing a passive state replication approach to simulate real systems with real-time data. To demonstrate the viability of the proposed digital twin, Eckhart & Ekelhart (2018b) implemented a behavior-specification-based intrusion detection system. They evaluated the effectiveness by launching a man-in-the-middle and an insider attack against a real CPS [6]. Evaluation results show that the proposed anomaly detector yields a low false-positive rate while being capable of detecting unknown attacks. In their work, the specification of a CPS is used to automatically build the digital twin model, which is a Finite State Machine (FSM). FSM does not consider time constraints for transitions as Timed Automaton does, making it difficult to replicate a CPS with delayed transitions such as SWaT, WADI, and BATADAL. Also, their intrusion

detector depends on the digital twin model to simulate the correct behavior of a CPS and when a mismatch between a state of the digital twin model and the corresponding state of the real operating CPS occurs, the CPS is considered to be under attack. However, mismatches are determined by predefined rules, which is not capable of detecting complicated attack patterns, such as attacks with delayed effects and attacks targeting at multiple access points at the same time. Our approach ATTAIN, however, mitigates this challenge by using Timed Automaton as the representation of the digital twin model and using GCN as the GAN generator to capture temporal characteristics.

C. GAN-based Anomaly Detection

GANs and the adversarial training framework have been successfully applied to model complex and high dimensional distributions of real-world data [33]. Existing GAN-based anomaly detectors can be divided into two categories: unsupervised and semi-supervised.

Schlegl et al. [34] introduced AnoGAN as a supervised anomaly detector. AnoGAN uses a standard GAN, which is trained only on positive samples. After the training, the generator learns to map a latent space representation z to a realistic normal sample $\hat{x} = G(z)$. AnoGAN then trains an encoder to map inversely, from real sample to latent space representation. In the detection phase, first, each sample is mapped back to the latent space and then fed into the generator. Given that the generator only learns how to generate normal samples, when an anomalous sample is encoded, its reconstruction will be non-anomalous. The difference between the input and the reconstructed image, therefore, highlights anomalies. AnoGAN is evaluated on a clinical image dataset, and their experiment results show that it can identify anomalies with the precision of 0.8834. However, AnoGAN suffers from a high computational expense because the generator and encoder are not trained simultaneously and a costly optimization procedure is required at test time.

Zenati et al. [35] proposed the Efficient GAN-Based Anomaly Detection (EGBAD) approach by introducing the BiGAN structure to anomaly detection. EGBAD aims to address the above-mentioned limitation of AnoGAN by adopting the ideas of Donahue et al. [36] and Dumoulin et al. [37] on learning the encoder and generator simultaneously. Doing so enables EGGBAD to compute the anomaly score without optimization steps during the inference as it happens in AnoGAN, reducing the time cost for the convergence of GAN.

Li et al. [22] proposed MAD-GAN, a multi-variate anomaly detector for time series data with GAN. They used LSTM as the base model in both generator and discriminator. The proposed method is a completely unsupervised GAN model, which is well-known for suffering from the "mode collapse" problem. Therefore, their evaluation results reveal a high false positive rate on the WADI dataset.

Fewer works have focused on developing semi-supervised and GAN-based anomaly detectors. Akcay et al. [38] proposed GANomaly - a novel semi-supervised model that can take advantage of both labeled and unlabeled data. They trained a generator network on normal samples to learn their manifold X while at the same time an autoencoder is trained to learn how to encode data in their latent representation efficiently. Employing encoder-decoder-encoder sub-networks in the generator network enables the model to map the input to a lower dimension vector, which is then used to reconstruct the generated output. They conducted experiments in various application domains, and the results demonstrate the efficacy and generalization of their model.

When comparing with these related works, ATTAIN relies on a digital twin model to generate ground truth labels, which consequently takes fake unlabelled samples produced by the generator as the input and generates fake labeled samples for the discriminator. Inspired by [32], ATTAIN builds the digital twin model as a Timed Automaton, which takes time constraints into consideration. As for the GAN-based digital twin capability, we adopt a GCN-LSTM structure to better capture the spatial and temporal characteristics of input data instead of the CNN-RNN structure, which was commonly used in existing anomaly detectors.

VIII. Conclusion and Future Work

In this paper, we presented Anomaly deTection with digiTAl twIN (ATTAIN), a novel digital twin-based approach to address the anomaly detection problem of Cyber-Physical Systems (CPS). ATTAIN consists of a timed automaton-based digital twin model and a GAN-based anomaly detector. The digital twin model provides ground truth labels indicating whether a CPS is operating in a normal state. Doing so allows ATTAIN to take advantage of a large amount of unlabeled real-time data obtained during the CPS operation, and enables ATTAIN to continuously learn along with the operation. We evaluated ATTAIN with three datasets obtained from three real-world CPS testbeds. Experiment results that the performance superiority of ATTAIN in comparison to two state-of-art anomaly detectors, increasing by 2.413%, 8.487%, and 5.438% on SWAT, WADI, and BATADAL, respectively.

In future work, we plan to conduct more experiments on real-world CPS of various domains to evaluate the scalability and generalization of ATTAIN. We will also consider exploring digital twins for more challenging tasks, such as detecting attacks targeting multiple CPS at the same time, which requires developing an integrated digital twin model. We also want to investigate how digital twins can be extended to provide additional advanced analyses during the operation of CPS such as predicting uncertainties and non-functional properties.

Acknowledgement

This work is supported by the security project funded by Norwegian Ministry of Education and Research, and Horizon2020 project ADEPTNESS, Grant Agreement Number: 871319.

REFERENCES

[1] Y. Harada, Y. Yamagata, O. Mizuno, and E. H. Choi, "Log-based anomaly detection of CPS using a statistical method," *Proceedings - 8th IEEE International Workshop on Empirical Software Engineering in Practice, IWESEP 2017*, pp. 1–6, 2017.

[2] L. Cheng, K. Tian, and D. D. Yao, "Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks," *ACM International Conference Proceeding Series*, vol. Part F1325, pp. 315–326, 2017.

[3] E. Aggarwal, K. Pattabiraman, M. Karimibiuki, and A. Ivanov, "CORGIDS: A correlation-based generic intrusion detection system," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 24–35, 2018.

[4] Q. Lin, S. Verwer, S. Adepu, and A. Mathur, "TABOR: A graphical model-based approach for anomaly detection in industrial control systems," *ASIACCS 2018 - Proceedings of the 2018 ACM Asia Conference on Computer and Communications Security*, no. March, pp. 525–536, 2018.

[5] Z. He, A. Raghavan, G. Hu, S. Chai, and R. Lee, "Power-grid controller anomaly detection with enhanced temporal deep learning," *Proceedings - 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering, TrustCom/BigDataSE 2019*, pp. 160–167, 2019.

[6] M. Eckhart and A. Ekelhart, *Security and Quality in Cyber-Physical Systems Engineering*, 2019, no. March 2020.

[7] A. P. Mathur and N. O. Tippenhauer, "SWaT: a water treatment testbed for research and training on ICS security," in *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. IEEE, 2016, pp. 31–36.

[8] 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 *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, 2017, pp. 25–28.

[9] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, A. Ostfeld, D. G. Eliades, M. Aghashahi, R. Sundararajan, M. Pourahmadi, M. K. Banks, B. M. Brentan, M. Herrera, A. Rasekh, E. Campbell, I. Montalvo, G. Lima, J. Izquierdo, K. Haddad, N. Gatsis, A. Taha, S. L. Somasundaram, D. Ayala-Cabrera, S. E. Chandy, B. Campbell, P. Biswas, C. S. Lo, D. Manzi, E. Luvizotto Jr, Z. A. Barker, M. Giacomoni, M. F. K. Pasha, M. E. Shafiee, A. A. Abokifa, M. Housh, B. Kc, and Z. Ohar, "The Battle Of The Attack Detection Algorithms: Disclosing Cyber Attacks On Water Distribution Networks," *Journal of Water Resources Planning and Management*, vol. 144, no. 8, p. 4018048, aug 2018.

[10] M. Eckhart and A. Ekelhart, "Securing Cyber-Physical Systems through Digital Twins," *Ercim News*, no. 115, pp. 22–23, 2018.

[11] A. Maier, "Online Passive Learning of Timed Automata for Cyber-Physical Production Systems," 2014.

[12] J. L. Schiff, *Cellular automata: a discrete view of the world*. John Wiley & Sons, 2011, vol. 45.

[13] Q. Ma, B. Burns, K. Narayanaswamy, V. Rawat, and M. C. Shieh, "Network attack detection using partial deterministic finite automaton pattern matching," mar 2011.

[14] B. K. Aichernig, A. Pferscher, and M. Tappler, "From Passive to Active: Learning Timed Automata Efficiently BT - NASA Formal Methods," R. Lee, S. Jha, and A. Mavridou, Eds. Cham: Springer International Publishing, 2020, pp. 1–19.

[15] G. Nasr, E. Badr, and C. Joun, "Cross Entropy Error Function in Neural Networks: Forecasting Gasoline Demand." *FLAIRS Conference*, no. January, pp. 381–384, 2002.

[16] M. Canizo, I. Triguero, A. Conde, and E. Onieva, "Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study," *Neurocomputing*, vol. 363, pp. 246–260, 2019.

[17] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 72–83, 2018.

[18] D. Yao, X. Shu, L. Cheng, and S. J. Stolfo, "Anomaly detection as a service: challenges, advances, and opportunities," *Synthesis Lectures on Information Security, Privacy, and Trust*, vol. 9, no. 3, pp. 1–173, 2017.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[20] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[21] J. Goh, S. Adepu, M. Tan, and Z. S. Lee, "Anomaly detection in cyber physical systems using recurrent neural networks," *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, no. March 2019, pp. 140–145, 2017.

[22] Z. Li, J. Li, Y. Wang, and K. Wang, "A deep learning approach for anomaly detection based on SAE and LSTM in mechanical equipment," *The International Journal of Advanced Manufacturing Technology*, vol. 103, no. 1-4, pp. 499–510, 2019.

[23] P. Schneider and K. Böttinger, "High-performance unsupervised anomaly detection for cyber-physical system networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, 2018, pp. 1–12.

[24] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015.

[25] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[26] A. Andoni and K. Onak, "Approximating edit distance in near-linear time," *Proceedings of the Annual ACM Symposium on Theory of Computing*, pp. 199–204, 2009.

[27] Z. Qu, Q. Xie, Y. Liu, Y. Li, L. Wang, P. Xu, Y. Zhou, J. Sun, K. Xue, and M. Cui, "Power cyber-physical system risk area prediction using dependent markov chain and improved grey wolf optimization," *IEEE Access*, vol. 8, pp. 82 844–82 854, 2020.

[28] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. Di Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174–216, 2019.

[29] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. D. Yao, "Deep Learning-Based Anomaly Detection in Cyber-Physical Systems: Progress and Opportunities," vol. 1, no. 1, pp. 1–29, 2020. [Online]. Available: http://arxiv.org/abs/2003.13213

[30] Z. Wu, Y. Guo, W. Lin, S. Yu, and Y. Ji, "A weighted deep representation learning model for imbalanced fault diagnosis in cyber-physical systems," *Sensors*, vol. 18, no. 4, p. 1096, 2018.

[31] J. E. Rubio, C. Alcaraz, R. Roman, and J. Lopez, "Analysis of intrusion detection systems in industrial ecosystems," *ICETE 2017 - Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, vol. 4, no. Secrypt, pp. 116–128, 2017.

[32] M. Eckhart and A. Ekelhart, "Towards security-aware virtual environments for digital twins," in *Proceedings of the 4th ACM workshop on cyber-physical system security*, 2018, pp. 61–72.

[33] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A Survey on GANs for Anomaly Detection," 2019. [Online]. Available: http://arxiv.org/abs/1906.11632

[34] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10265 LNCS, pp. 146–147, 2017.

[35] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient GAN-Based Anomaly Detection," 2018. [Online]. Available: http://arxiv.org/abs/1802.06222

[36] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial Feature Learning," no. 2016, pp. 1–18, 2016. [Online]. Available: http://arxiv.org/abs/1605.09782

[37] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville, "Adversarially learned inference," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, no. July, 2017.

[38] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "GANomaly: Semi-supervised Anomaly Detection via Adversarial Training," *Lecture Notes in Computer Science (including subseries Lecture Notes in Ar-*

tificial Intelligence and Lecture Notes in Bioinformatics), vol. 11363 LNCS, pp. 622–637, 2019.