Digital Twin-based Anomaly Detection with Curriculum Learning in Cyber-physical Systems

QINGHUA XU, Simula Research Laboratory, Norway and University of Oslo, Norway
SHAUKAT ALI, Simula Research Laboratory, Norway
TAO YUE*, Simula Research Laboratory, Norway

Anomaly detection is critical to ensure the security of cyber-physical systems (CPS). However, due to the increasing complexity of attacks and CPS themselves, anomaly detection in CPS is becoming more and more challenging. In our previous work, we proposed a digital twin-based anomaly detection method, called ATTAIN, which takes advantage of both historical and real-time data of CPS. However, such data vary significantly in terms of difficulty. Therefore, similar to human learning processes, deep learning models (e.g., ATTAIN) can benefit from an easy-to-difficult curriculum. To this end, in this paper, we present a novel approach, named digitaL twin-based Anomaly deTecTion wIth Curriculum lEarning (LATTICE), which extends ATTAIN by introducing curriculum learning to optimize its learning paradigm. LATTICE attributes each sample with a difficulty score, before being fed into a training scheduler. The training scheduler samples batches of training data based on these difficulty scores such that learning from easy to difficult data can be performed. To evaluate LATTICE, we use five publicly available datasets collected from five real-world CPS testbeds. We compare LATTICE with ATTAIN and two other state-of-the-art anomaly detectors. Evaluation results show that LATTICE outperforms the three baselines and ATTAIN by 0.906%-2.367% in terms of the F1 score. LATTICE also, on average, reduces the training time of ATTAIN by 4.2% on the five datasets and is on par with the baselines in terms of detection delay time.

CCS Concepts: • **Software and its engineering → Maintaining software**; • **Security and privacy → Intrusion detection systems**; • **Computer systems organization → Embedded systems**; **Sensors and actuators**; **Embedded and cyber-physical systems**; • **Computing methodologies → Neural networks**; **Supervised learning by classification**; **Online learning settings**.

Additional Key Words and Phrases: cyber-physical system, digital twin, curriculum learning, deep learning, anomaly detection

1 INTRODUCTION

Cyber-physical systems (CPS) have been deployed in many applications [36]. These systems are also becoming more complex, heterogeneous, and integrated, i.e., consisting of multiple systems to provide rich functionalities, which expose CPS to broader threats. Such threats can cause significant damage to CPS. Therefore, anomaly detection, which can potentially detect these threats in advance, is a crucial task in the domain of CPS security.

Multiple anomaly detection mechanisms for CPS have been proposed [4, 15, 18]. These traditional mechanisms include invariant checking, physical attestation, and fingerprinting [15]. Recently, deep learning-based anomaly detectors are intensively studied due to their success in various domains [35], such as image classification and

*Corresponding author

Authors' addresses: Qinghua Xu, Simula Research Laboratory, Oslo, Norway and University of Oslo, Oslo, Norway, qinghua@simula.no; Shaukat Ali, Simula Research Laboratory, Oslo, Norway, shaukat@simula.no; Tao Yue, Simula Research Laboratory, Oslo, Norway, tao@simula.no.

1049-331X/2023/2-ART $15.00
https://doi.org/10.1145/3582571

natural language processing. Typically, anomaly detectors are built as deep learning models and trained on static data collected from a CPS. Despite the success of these neural network-based anomaly detectors, most of them still suffer from two significant challenges. First, most of these models are trained on static log data and cannot keep learning during the operation of CPS. Consequently, these models perform the best when dealing with known attacks but suffer from low accuracy when facing novel attacks. Second, neural network-based models rely on a large amount of labeled data for training, which is known to be expensive in CPS [18]. Comparably, a model that can take advantage of unlabeled data is, therefore, much more desirable and appreciated in practice.

To tackle the above-mentioned challenges, in our previous work, we proposed ATTAIN [63], which takes advantage of unlabeled data and continuously learns during the operation of a CPS, i.e., at runtime. ATTAIN achieved new state-of-the-art results on benchmark datasets by taking advantage of both historical and real-time data introduced in chronological order. Especially, training on live data keeps improving the predictive performance and efficiency of the digital twin, allowing it to make more accurate and earlier predictions of anomalies.

Recently, the latest research on Curriculum Learning (CL) demonstrates that re-organizing the training data as a curriculum can boost the performance of deep learning models [10, 11, 27, 28, 51]. CL is inspired by a key observation from human learning processes: humans acquire knowledge via performing a series of tasks, usually from easy to difficult tasks. In essence, human learning is typically organized as curricula. Similarly, machine learning algorithms can also benefit from CL. According to extensive research in this area, CL is proven to be effective in various machine learning domains, such as computer vision and natural language processing [10, 25, 50, 62, 70]. However, few works focus on exploring CL with time-series data from CPS, which is inherently chronologically ordered [28]. Most existing works utilize sequential deep learning models to process these time-series data directly, without making any changes to the order of training data [34].

In this paper, we extend ATTAIN by employing CL and form the following new contributions:

- We designed **a generic framework** that combines CL with ATTAIN. Next, we developed novel difficulty measurers and adapted CL to CPS time series data. In particular, we proposed two types of difficulty measurers: predefined and automatic difficulty measurers, which focus on the property of samples and are based on the context of the samples, respectively.
- We conducted **extensive empirical study** with five case studies, among which, as compared to ATTAIN, two case studies are newly added. These two case studies further demonstrate the generalizability of LATTICE over diverse datasets.
- We performed **extensive analyses with statistical tests** to evaluate LATTICE. We first evaluate LATTICE with coarse-grained and fine-grained effectiveness metrics, followed by investigating the effectiveness of CL and its components and the efficiency of LATTICE. Moreover, we discussed **the plausible reasons for the improvement**, including taking advantage of predefined difficulty measurer, automatic difficulty measurer, and CL's optimization strategy.

In total, we evaluate the cost-effectiveness of LATTICE on five datasets collected from real-world critical infrastructure testbeds: Secure Water Treatment (SWaT) [40]— a multi-stage water purification plant, Water Distribution (WADI) [1]— a consumer distribution network, Battle of Attack Detection Algorithms (BATADAL) [53]— a dataset designed for an attack detection contest, PHM challenge 2015 dataset [26] and Gas Pipeline Dataset [41]. We demonstrate that LATTICE improves the performance of the state-of-the-art anomaly detection methods by 0.906%, 2.363%, 2.712%, 2.008%, 2.367%, respectively, in terms of the F1 score. We also evaluate the time efficiency of LATTICE with Unit Training Time (UTT) and Detection Delay Time (DDT). Experiment results show that LATTICE improves UTT by 4.2% and DDT by 0.2% on average.

The remaining part of this paper is organized as follows: Section 2 introduces the background of CL, digital twin, and Generative Adversarial Networks (GAN). We present related works in Section 3 a running example in Section 4. Section 5 introduces LATTICE in detail. Section 6 shows our empirical evaluation, followed by Section 7 where we present the experimental results and analysis. Section 9 shows possible threats to validity in our experiments. Section 8 provides the overall discussions. Finally, Section 10 summarizes the paper and proposes potential future work. A replication package of the experiments is provided here [1].

2 BACKGROUND

In this paper, we combine two main techniques: CL and digital twin. We first present a general CL framework in Section 2.1, while in Section 2.2 we present the concept of digital twin and its components. Inside digital twin, we use GAN as the backbone for our model. In Section 2.3, we introduce the basic structure and key functions of GAN.

2.1 Curriculum learning

CL was first introduced into machine learning models by Selfridge et al. [50]. In that paper, CL was used to address the cart pole controlling task, which is a classic problem in robotics. This work later inspires many researchers to explore CL for various tasks such as grammar learning [31, 31, 46] and language learning [6]. Bengio et el [7] first proposed the concept of CL in the domain of language learning. Below is the definition of *Curriculum* and CL.

Definition 2.1. *A curriculum is a sequence of training criteria over T training steps:* $C =< Q_1, ..., Q_t, ..., Q_T >$. *Each criterion* Q_t *is a reweighting of the target training distribution* $P(z)$*:*

$$Q_t(z) \propto W_t(z)P(z) \qquad \forall example\ z \in training\ set\ D \tag{1}$$

under the condition that the entropy of the distribution and weights, for any example, increase. Also, in the final steps, all examples are uniformly sampled.

With the definition of *Curriculum*, Bengio et el. [7] defined CL as follows.

Definition 2.2. *Curriculum learning is a training strategy that trains a machine learning model with a curriculum defined above.*

Most of the studies on CL follow this concept [7, 27, 44, 59]. Wang et. al. [58] summarized these works and proposed a general framework of CL as "Difficulty Measurer + Training Scheduler". The difficulty measurer decides the relative "easiness" of each data sample, while the training scheduler decides the sequence of data subsets throughout the training process, based on the judgment of the difficulty measurer. In this paper, we follow this framework and define our own difficulty measurer and training scheduler.

2.2 Digital Twin

El Saddik [13] defined the 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 a physical twin - a commonly used term in the literature [15, 66, 67]. The 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 action on the physical twin. Examples of such action include preventing unsafe situations or alerting a CPS operator about abnormal behaviors.

As shown in Fig 1, a digital twin consists of two parts: the digital twin model (DTM) and the digital twin capability (DTC). The *DTM* refers to the digital representation of the CPS. This representation could be in the form of heterogeneous models corresponding to different components such as software, hardware, and communication.

[1] https://github.com/xuqinghua-China/tosem/tree/master

We treat these components as black boxes and build the DTM with a data-driven approach. These models can be represented as state machines, machine learning models, etc. The *DTC* refers to the functionality of a digital twin. Depending on the context, a digital twin can provide various functionalities, such as predictions of non-functional properties, uncertainty detection, and failure prevention. In our context, we focus on the anomaly detection capability that detects abnormal patterns exhibited by sensors and actuators and usually caused by external attacks. Moreover, the *DTC* interacts with the *DTM*, e.g., to perform simulations on it. Similarly, the *DTM* interacts with the *DTC*, e.g., to get feedback about whether or not the system is under attack. This bidirectional relationship is shown as a dashed line between the *DTC* and the *DTM* in Fig 1.

Traditionally, digital twins are constructed statically with historical data only [13]. However, in some instances, we may have CPS design models that could be used and improved incrementally during the operation of the digital twin based on live data, thereby representing the most up-to-date state of the underlying CPS. In particular, as discussed in our previous work ATTAIN [63], such live data improves both predictive performance and the efficiency of the digital twin. Better predictive performance indicates that the digital twin can make more accurate anomaly predictions, while better efficiency indicates that it takes less time for a digital twin to detect an anomaly.

In this paper, we build the DTM as a timed automaton machine, and the backbone of the DTC is GAN. The DTM provides ground truth labels to improve the anomaly detection capability of LATTICE. As for training, we train such models initially with historical data, followed by their continuous improvement with live data.

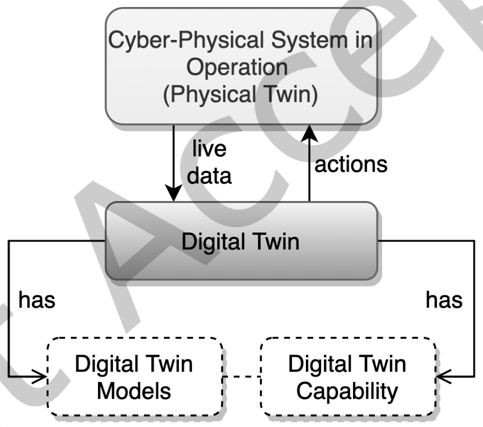

Fig. 1. Digital Twin for Cyber-Physical System

2.3 Generative Adversarial Networks

As mentioned in Section 2.2, we utilize GAN as the backbone of DTC. GAN was invented by Ian Goodfellow in [19] and has been successful in performing many data generation tasks [24] e.g., image generation [5] and text generation [71]. These successful applications demonstrate GAN's generalizability and inspire us to adapt it to the CPS security domain.

A typical GAN has independent models: generator (G) and discriminator (D) (see the GAN structure in Figure 2). 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 $D(u)$ assessing whether the given sample u is from the real data set or 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 the goal of D is the opposite. Thus, G learns to generate more realistic adversarial samples while D continuously improves its capability of distinguishing them from real samples.

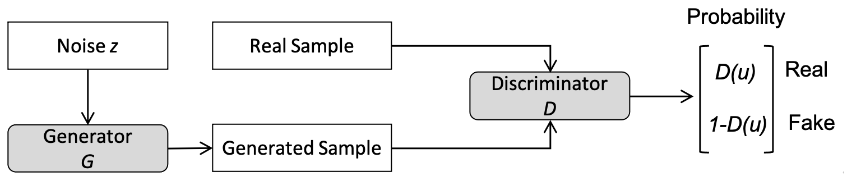

Fig. 2. Generative Adversarial Network Structure

3 RELATED WORK

In Section 3.1, we present related works about CPS anomaly detection. In Section 3.2, we present the research progress of CL and how to incorporate CL and digital twin.

3.1 CPS Anomaly Detection

3.1.1 Traditional Anomaly detection. Anomaly detection is a popular research topic in the domain of CPS security. Many traditional anomaly detection methods attempt to learn normal states of CPS with predefined rules(e.g., frequency limit), state estimation (e.g., the Kalman filter), and statistical models (e.g., Gaussian model, histogram-based model) [36]. However, a disadvantage of these methods is the requirement for domain knowledge or data distribution of normal data. On the other hand, machine learning approaches have proved successful in various domains by leveraging data instead of domain knowledge to improve performance. Following this research line, our method adapts machine learning techniques to the CPS domain.

3.1.2 Deep Learning-based Anomaly Detection. 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 [36]. Jonathan et al. [18] 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., [8, 30, 65]) 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. [8] and Wu et al. [61] used Recurrent Neural Network (RNN) to take the output of the Convolutional Neural Network (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. Such methods prove the effectiveness of deep learning techniques in various domains. We follow this research line and use LSTM to capture temporal characteristics of CPS data while utilizing Graph Convolution Network (GCN) instead of CNN to capture their spatial characteristics better. However, deep learning methods work best when trained on large amounts of labeled data, which can be costly to acquire in the CPS domain. Therefore, we take advantage of a digital twin for data augmentation.

3.1.3 Digital Twin-based Anomaly Detection. A few papers have proposed to use digital twins for anomaly detection. Eckhart & Ekelhart [13] 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. Given 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 is available from operating systems as LATTICE does. The reason is that 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 [14] 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 [15]. 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. Also, their intrusion detector depends on the digital twin model to simulate the correct behavior of a CPS. 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 cannot detect complicated attack patterns, such as attacks with delayed effects and attacks targeting multiple access points simultaneously. LATTICE, however, mitigates this challenge by using Timed Automaton as the digital twin model and GCN as the GAN generator to capture temporal characteristics.

3.2 Curriculum Learning

Based on the framework proposed in [58], existing CL methods can be divided into predefined CL and automatic CL.

In predefined CL, the difficulty measurer and training scheduler are completely designed based on prior human knowledge without any data-driven models or algorithms involved. The difficulty measurers of a predefined CL need to be manually designed based on characteristics of specific data. Since CL was originally designed for computer vision and natural language processing tasks, most predefined difficulty measurers are related to images or text data, such as complexity [44, 59], diversity [6, 27], and noise estimation [9, 10]. Unlike predefined difficulty measurers, predefined training schedulers are usually task/data agnostic. In general, predefined training schedulers can be divided into two types: discrete and continuous training schedulers. The most popular discrete scheduler is called Baby Step [7]. The Baby Step algorithm first divides data into multiple buckets based on the difficulty score of each sample. The training process starts with the easiest bucket and slowly includes harder buckets after several training epochs. Other discrete schedulers, including One-Pass [7] and modified Baby Step [27], are also widely adopted due to their demonstrated simplicity and effectiveness.

Despite its success, predefined CL is limited to domain knowledge and prior information requirements, which might not be accessible in certain application contexts. Also, predefined CL stays fixed during training, is incapable of adapting to potential novel scenarios during operation. Therefore, automatic CL has attracted much attention recently. Self-paced Learning (SPL) [32, 56] is a primary branch of automatic CL. The intuition behind SPL originates from human learning, where a student teaches her/himself, and controls the content, method, time, and length of the study. For instance, Kumar et al. [32] proposed an SPL approach that trains the model at each iteration with the easiest subset based on the model's current performance, i.e., the examples with the lowest training losses. Another type of automatic CL is Transfer Teacher (TT) [21, 60, 62]. Unlike SPL, which depends on the student to teach itself, TT takes another pre-trained model as the teacher and transfers "knowledge" to the student model. SPL has a risk of uncertainty at the beginning of training if the student model is not mature enough to teach itself, while TT reduces this risk by inviting a mature teacher to help the student model for assessment. We follow this research line and use DTM as the teacher model and DTC as the student model, allowing LATTICE to adapt its curricula automatically and continuously.

4 RUNNING EXAMPLE FROM THE SWAT TESTBED

To better illustrate our method, we present a running example in Table 1. In this table, we use sensor and actuator data from the first stage of the SWaT testbed. In this testbed, sensors include a flow indicator transmitter (FIT101) and level indicator transmitter (LIT101), while actuators consist of pumps (P101, P102) and a 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 an observation at a time point, consisting of sensor and actuator values, as shown in equation 2:

$$u = [u_{s1}, u_{s2}, u_{s3}, ...u_{a1}, u_{a2}, u_{a3}, ...], \tag{2}$$

where u_{sj} represents a value for the j^{th} sensor and u_{ak} represents the value for the k^{th} actuator. We define $u^i \in R^n$ 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.

Table 1. Running Example

Timestamp	FIT101	LIT101	MV101	P101	P102	Label	Difficulty	# of Batch
10:00:00	2.43	522.84	2	2	1	Normal	0.1	1
10:00:01	2.45	522.88	2	2	1	Normal	0.1	1
...	...	...	...	...	...	...	...	...
10:29:13	2.44	816.84	2	1	1	Normal	0.4	23
10:29:14	2.49	817.67	23	1	1	Attack	0.5	23
10:29:15	2.54	817.94	23	1	1	Attack	0.5	23
...	...	...	...	...	...	...	...	...
10:44:53	6e-4	869.72	1	2	1	Attack	0.4	22

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 damage to the plant. Our objective is to detect such attacks in advance before they cause actual damages, with the help of U^i. We aim to make both coarse-grained and fine-grained predictions. For coarse-grain predictions, we calculate the number of attacks detected. For fine-grained predictions, we predict the labels for each data sample. For instance, we want to predict the label column shown in Table 1 for each given time step.

We also show the assigned difficulty score and batch number for CL. Both of them are calculated with LATTICE, which we will introduce in detail in Section 5. Intuitively, we define difficulty score to be a number between 0.0 and 1.0, indicating the level of difficulty this sample presents. To be more specific, the higher the difficulty score is, the more difficult this sample is. We assign a new batch number for each sample and change the order based on these scores. Formally, we use d_i and b_i to denote the difficulty score and batch number for sample i. For instance, the first sample in Table 1 has a difficulty score of 0.1 ($d_1 = 0.1$) and its batch number is 1 ($b_1 = 1$). This means that the first sample is relatively easy for the model to learn, and it will be included in the first batch of the training dataset. In this work, we aim to predict the label column at each time point.

5 APPROACH

LATTICE follows the general CL framework proposed in [58]. As we discussed in Section 2, the main idea of CL is about training models from easier data to harder data. Therefore, a general CL design consists of *difficulty* *measurer* and *training scheduler*. Difficulty measurer decides the relative "difficulty" of each sample, while *training scheduler* arranges the sequence of data subsets throughout the training process based on the judgment of *difficulty measurer*. As shown in Figure 3, all the training examples are sorted by the *difficulty measurer* from the easiest to the hardest and passed to the *Training Scheduler*. Then, at each training epoch t, the *training scheduler* samples a batch of training data from the relatively easier examples and sends it to the *ATTAIN* for training. With progressing training epochs, the *Training Scheduler* decides when to sample from harder data. As shown in the running example (Table 1), the Difficulty column presents the difficulty scores given by the *difficulty measurer*, indicating the relative "difficulty" of each sample. For instance, the sample data at 10:00:00 is assigned a difficulty score of 0.9, i.e, $score(u^{10:00:00}) = 0.9$, while the difficulty score of the sample at 10:29:12 is 0.5, i.e, $score(u^{10:29:12}) = 0.5$. This tells that sample $u^{10:00:00}$ is relatively harder for the model to learn. With these difficulty scores, the training scheduler decides which samples should be included in each batch. The general principle is that easy samples should be included first. After calculation, the training scheduler assigns new batch numbers for $u^{10:00:00}$ (batch number=23) and $u^{10:19:12}$ (batch number=2). In the following section, we will present more details about the *difficulty measurer*, *training scheduler* and the extension to ATTAIN in Section 5.1, Section 5.2, and Section 5.3, respectively.

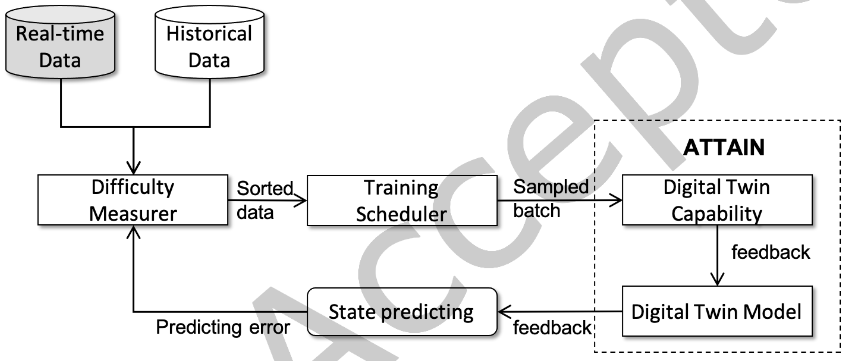

Fig. 3. Overview of LATTICE

5.1 Difficulty Measurer

Wang et al. [58] pointed out that it is difficult to find the best combination of difficulty scorer and training scheduler for a specific task except for performing an exhaustive search, which is often impossible to do for a complex problem. Thus, we did not perform an exhaustive search. However, we investigated various options of difficulty scorers and selected the best-performed ones. Particularly, we use two different types of difficulty measurers, namely predefined and automatic difficulty measurers. Predefined difficulty measurers are designed based on human prior knowledge with no data-driven models or algorithms involved, while automatic difficulty measures are learned by data-driven models or algorithms. Predefined difficulty measurers have been proven to be effective in various tasks and have been popularly used in multiple domains due to their simplicity [58] (also see Section 3.2). However, automatic difficulty measurers require less expert knowledge and can interact with current models to adapt to new data.

5.1.1 Predefined Difficulty Measurer. Predefined difficulty measurers are usually designed with the help of domain knowledge. Researchers have manually designed various difficulty measurers based on data characteristics of specific tasks [16, 43, 45, 52, 54]. We adapt these works to CPS data and propose our own predefined difficulty measurers s_{pdm}. As shown in Equation 3, s_{pdm} is calculated with four different types of domain knowledge:

complexity (s_{comp}), diversity (s_{div}), noise (s_{noi}), and system vulnerability (s_{vul}). Each of them is normalized and then all are summed.

$$s_{pdm} = normalize(s_{comp}) + normalize(s_{div}) + normalize(s_{noi}) + normalize(s_{vul}) \tag{3}$$

Complexity stands for the structural complexity of data samples, such as the dimension and input space of data. Most of the time, the number of sensors and actuators remains unchanged during the operation of CPS, while attackers can potentially change sensor and actuator values into new ones. This increases the input size of data samples, which subsequently increases the complexity. Therefore, we calculate the complexity of each data sample as in Equation 32

$$s_{comp} = size(sensors\ and\ actuators\ input\ space) \tag{4}$$

Diversity stands for the distributional diversity of data samples. A data sample is considered to increase the diversity of the dataset if the frequency of this data sample is low. A higher diversity potentially indicates that data samples are distributed in more types. In this paper, the diversity difficulty measurer adopts the naive Bayesian assumption and calculates the probability of this data sample as in Equation 5. Under this assumption, we consider actuators to be independent of each other and use frequency to estimate this probability. The probability of this data sample is factorized as the product of the frequency of each actuator value.

$$s_{div} = \prod_{i=0}^{n} frequency(u_i) \tag{5}$$

Noise is about the noise level of data samples. A data sample tends to be noisy if its value deviates from the context. The noise difficulty measurer calculates this deviation as the standard scor, as shown in Equation 6, where $\mu$and σ denote the expected value and standard deviation of data samples from the context.

$$s_{noi} = \frac{u_i - \mu_i}{\sigma} \tag{6}$$

Vulnerability determines how vulnerable the system is to attacks. We define vulnerability as the distance to a known attack since attackers tend to invade the system when it is vulnerable and we hypothesize that samples around time points of attacks are more vulnerable to attacks. Consequently, samples with higher vulnerability should be assigned higher difficulty scores because of the increased complexity, diversity, and noise. We argue that all these three characteristics increase when attacks happen or are about to happen, as explained below:

- **Complexity**. Typically during attacks, values of sensors and actuators become unstable and often deviate from normal ranges. This increases the input space while the input dimension remains the same (the sum of sensors and actuators). We assume that samples with larger input space are more complicated to be trained on.
- **Diversity**. This increase comes from new data patterns introduced by attackers, which tend to manipulate the values of actuators and sensors for their intended purposes. Such a manipulation inevitably increases the diversity of data, thereby increasing the difficulty of model training on this data.
- **Noise**. Given that our approach relies on DTM to give ground truth labels of real-time data, the confidence level of this labeling process is lowered when attacks happen. In other words, labels given by DTM become noisy. Many deep learning models can be induced to fit these noisy label distributions instead of real data distribution. Subsequently, these models tend to make inaccurate predictions around time points of attacks, hence the high difficulty score.

In the context of our running example in Section 4, an attack was introduced at 10:29:14, which implies that the complexity, diversity, and noise of samples around $10:29:14$ tend to increase. As a result, samples around $u^{10:19:14}$ are hard to be trained on and hence obtain high difficulty scores.

We formally define a distance-based vulnerability difficulty measurer s_{vul} based on the above-discussed hypothesis as shown in Equation 7: s_{vul} is the time distance d^i between current sample u^i and closest attack sample u^c. As in the running example, the closest attack to sample $u^{10:00:00}$ is $u^{10:29:14}$. Therefore, s_{vul} is calculated as $d^i = |timeDistance(u^{10:29:14} - u^{10:00:00})|$. To enable calculations with other difficulty measures in the future, we further scale it to be within 0 and 1 with the min-max normalizer as in Equation 8, where min and max denote the minimum and maximum values of time distance in the training samples, respectively. Finally, we use this scaled distance as difficulty score s^i_{vul} for sample u^i as in Equation 8:

$$d^i = |timeDistance(u^c - u^i)| \tag{7}$$

$$s_{vul} = \frac{d^i - min}{max - min} \tag{8}$$

Noticeably, the $timeDistance$ function should be carefully designed to preserve temporal characteristics of anomaly detection data, which is usually in the form of sequences. To that end, we use a sliding window mechanism for this function. In particular, we define hyperparameter s_{window} to denote the size of this sliding window, within which samples share the same distance. Let d^i_o and d^c_o to be the original time distances for sample u^i and sample u_c. We calculate the window distance for any given sample as in Equation 9:

$$timeDistance(u^i) = floor(\frac{d^i_o - d^c_o}{s_{window}}) \tag{9}$$

5.1.2 Automatic Difficulty Measurer. Despite the simplicity and effectiveness of predefined difficulty measurers, they have some essential limitations [58]. First, a predefined difficulty measurer remains unchanged during runtime, i.e., being unable to adapt to new data generated from CPS in operation. Second, a predefined difficulty measurer requires a good grasp of domain knowledge, which can be quite expensive and time-consuming in practice. Last but not least, the definition of difficulty for humans and machines can be quite different; what humans assume to be easy can be quite difficult for machines to comprehend. This discrepancy of decision boundaries between humans and machines causes challenges for experts to define difficulty scores manually.

To alleviate these problems, various automatic difficulty measurers have been developed and explored in the literature, including self-paced learning [32], transfer teacher [70], reinforcement learning teacher [20], and other automatic difficulty measurers [25, 47, 55]. Inspired by these methods, we modify difficulty scores automatically with prediction errors, which are critical indicators of CPS uncertainty. Substantial work has been conducted in the literature, demonstrating the importance of handling uncertainties in CPS security and safety [22, 23, 38, 64, 68, 69]. In our context, we focus on prediction errors of DTM, which is pretrained on historical data. DTM in ATTAIN simulates corresponding CPS with high realism. Therefore, higher prediction errors of DTM indicate higher noise levels of labels produced by DTM. As mentioned in Section 5.1.1, training deep learning models with noisy data is more difficult. Therefore, we assume samples with higher prediction errors should be assigned with higher difficulty scores. Based on this assumption, we define the following two types of automatic difficulty measurers: Hamming Distance-based Measurer (HDM, Definition 5.1) and Cross Entropy-based Measurer (CEM, Definition 5.2). Hamming distance is commonly used to calculate the difference of two strings of equal length, while cross-entropy loss estimates uncertainty by comparing real distribution and prediction. We are aware that there are other distance-based and entropy-based metrics, we, however, argue that hamming distance and cross-entropy are commonly used and representative. In the future, we will explore other options.

Definition 5.1. *HDM.*

Let $\hat{u}_{DTM}$ be the predicted state for the current time point. We define HDM score s_{HDM} as in equation 10:

$$s^i_{HDM} = \frac{\sum_{k=1}^{n} \hat{u}^i_k + u^i_k}{n} \tag{10}$$

where $\hat{u}^i_k$ denotes the kth element of predicted state vector at time point i. As in the running example, $u_3^{10:00:00}$ denotes the third element of sample $u^{10:00:00}$, which is the value of MV101 (2).

Definition 5.2. *CEM.*
We define CEM score s_{CEM} as in equation 11:

$$s^i_{CEM} = -\sum_{k=1}^{n} softmax(u)^i_k \times log(softmax(\hat{u})^i_k) \tag{11}$$

5.1.3 Combined Difficulty Measurer. As previously mentioned in Section 5.1.1 and Section 5.1.2, predefined and automatic difficulty measurers have both advantages and disadvantages. Predefined difficulty measurers can incorporate expert knowledge into learning processes, but it is costly in practice. On the other hand, automatic difficulty measurers can self-adjust during training, which saves considerable time and cost. However, automatic difficulty measurers depend only on training losses and prediction errors acquired from data and discard expert knowledge. Therefore, we propose to combine these two paradigms by introducing prior expert knowledge into the automatic learning process.

We define a hyperparameter λ ($0 < \lambda < 1$) to control the influence level of the predefined difficulty measurer (prior knowledge). The combined measurer is defined as in equation 12 and equation 13:

$$s^i_{cb1} = s^i_{HDM} + \lambda s^i_{PDM} \tag{12}$$

$$s^i_{cb2} = s^i_{CEM} + \lambda s^i_{PDM} \tag{13}$$

5.2 Training Scheduler

As for the training scheduler, we also explored several options including baby steps [44], one pass [44], and root function continuous scheduler [10]. However, we investigated the performance of these training schedulers. Results from this investigation show the superiority of the combination of the proposed difficulty scorer and baby step training scheduler. Baby step scheduler first distributes the sorted data into buckets from easy to hard and starts training with the easiest bucket. After a fixed number of training epochs or convergence, the next bucket is merged into the training subset. Finally, after all the buckets are merged and used, the whole training process either stops or continues with several extra epochs. Note that at each epoch, the scheduler usually shuffles both the current buckets and the data in each bucket and then samples mini-batches for training (instead of using all data at once).

5.3 ATTAIN Extension

In addition to implementing CL, we also extended ATTAIN by introducing gated GCN. The top part of Fig. 4 shows the overview of 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 [39]

Algorithm 1: Baby Step Training Scheduler

```
Input: $D$: training dataset; $C$: the difficulty measurer
Output: $M^*$: the optimal model.
1 $D' = sort(D, C)$;
2 $D^1, D^2, ..., D^k = D'$ where $c(d_a) < c(d_b), d_a \in D^i, d_b \in D^j, \forall i < j$;
3 $D^{train} = \emptyset$;
4 for $s=1...k$ do
5     $D^{train} = D^{train} \cup D^s$;
6     while not converged for $p$ epochs do
7         $train(M, D^{train})$
8     end
9 end
```

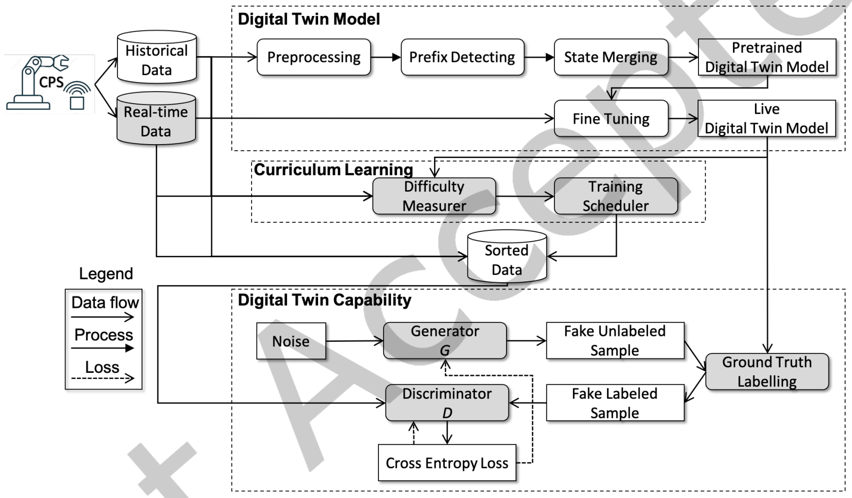

Fig. 4. Details of ATTAIN Extension

algorithm, a multi-stage algorithm with the following steps: pre-processing, prefix detection, and state merging. We extend this algorithm to learn probabilistic real-time automaton.

In this case, the DTC, i.e., the anomaly detector, is trained only on real-time data, where GAN is used as the backbone framework. GAN consists of a generator and a discriminator. The generator learns to generate adversarial samples, mapping from a latent space to input data distribution. 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 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.

5.3.1 Digital Twin Model Generation. A digital twin model is often a behavioral model. Such a model is usually represented as a state chart or a finite automaton and is mostly created manually by domain experts [37, 49]. However, there exist approaches on the automated construction of Deterministic Finite automaton such as the ones reported in [2, 39]. 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.

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; and
- δ is a transition timing constraint $\delta : T \longrightarrow I$, where I is a set of probability distribution functions with regard to time, which denotes how much time is needed for a certain transition.

We use OTALA as a learning algorithm, which was introduced by Maier et. al. [39] as a generic algorithm. We further 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. In our example (Table 1), the state vector at 10:00:00 is $u_0 = [2.43, 522.84, 2, 2, 1]$. And at 10:00:01, we observe a new state $u_1 = [2.45, 522.88, 2, 2, 1]$ which has not been observed before. Therefore we add u_1 into the states set U and add one transition from u_0 to u_1 into the transition set T. Then the algorithm continues to take another input at the next second.

5.3.2 Digital Twin Capability. The core part of Digital Twin Capability 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). G learns to generate more realistic samples while D continuously improves its capability of distinguishing adversarial samples from real samples (as also discussed in Section 2.3). Here we modify this vanilla GAN for our purpose. We calculate a ground truth label with the help of the 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.

In this paper, we extend ATTAIN by replacing GCN in the generator with Gated GCN, while the discriminator stays unchanged. Gating mechanisms have been effective in RNNs such as Gated recurrent units and LSTM. Gating mechanisms have been proven to be effective in fitting more complicated data. They control the information flow through their recurrent cells. In the case of GCN, these gated units control the domain information that flows to pooling layers. The model must be robust to change in domain knowledge and should be able to generalize well across different domains. For instance, in our example (Table 1), LIT101, FIT101, and P101 are connected. ATTAIN requires domain knowledge of SWaT to acknowledge this connection, while LATTICE can learn this during training automatically. In the following part of this section, we elaborate on the structure of the generator.

As described in the running example, let $u[i]$ be the ith sensor or actuator values of the system. We consider values within a predefined time window as input, e.g., $u[i], u[i-1], ..., u[i - window\_size]$. 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 14, 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 15. $u[i]$ is made up by concatenating $a[i]$ and $s[i]$ as in equation 16:

$$a'[i] = oneHot(a[i]) \tag{14}$$

$$s'[i] = concat(s[i], upper\_limit, lower\_limit) \tag{15}$$

$$u'[i] = concat(a'[i], s'[i]) \tag{16}$$

- **GCN Layer.** GCN Layer captures interdependent relationships among sensors and actuators. In ATTAIN, GCN takes a specification graph as well as sensor and actuator values as the input. In this graph, each actuator and sensor is viewed as a node, while an edge is drawn when there is a connection between these two nodes according to the CPS process. However, specification graphs are not always available, which motivates us to extend GCN with Gated GCN to learn graph edges automatically. As shown in the running example, we find the connection between MV101, P101, LIT101, and FIT101 in the specification graph. Therefore the corresponding edge weight of these four nodes will be set as 1 in ATTAIN, while other edges such as edges between P101 and P102 will be set as 0. However, in LATTICE, the weight matrix will be initialized randomly at the beginning and updated during training. We formally define this gated GCN Layer as follows.
Let $P[i]$ be the graph edge weight matrix at time point i. We first calculate a new candidate matrix $P_c[i]$ as in equation 17, where W_c and b_c are weight and bias matrices, respectively.

$$P_c[i] = tanh(P[i]W_c + b_c) \tag{17}$$

We then calculate the control matrix P_w as in equation 18:

$$P_w[i] = sigmoid(P[i]W_w + b_w) \tag{18}$$

Finally, we update edge matrix E for the current time point as in equation: 19

$$E = P_w[i] \times P_c[i] \tag{19}$$

Consequently, we have an updated graph g as in equation 20. Gated GCN takes graph g and vector $u'[i]$ as input as shown in equation 21:

$$g = (E, V) \tag{20}$$

$$gcn_{out}[i] = GatedGCN(g, u'[i]) \tag{21}$$

- **Pooling Layer.** In the pooling layer, all the GCN outputs collapse into one vector, as shown in equation 22 below:

$$spatial[i] = maxPooling(gcn_{out}) \tag{22}$$

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 23. LSTM learns temporal features by calculating hidden states between each time step as in Equation 24. The last hidden state $h[i]$ is used as our final representation $fake[i]$ for time step i as shown in equation 25:

$$input_w = concat(spatial[i : i - window\_size]) \tag{23}$$

$$h[i] = LSTM(input_w) \tag{24}$$

$$fake[i] = h[i] \tag{25}$$

6 EXPERIMENT DESIGN AND EXECUTION

Section 6.1 presents the research questions (RQs) that we would like to answer. Section 6.2 presents the case studies we used for experimentation, followed by the evaluation metrics (Section 6.3). Section 6.4 provides parameter settings of the experiments, and Section 6.5 details the experiment execution process.

6.1 Research Questions

In our experiment, we are interested in answering the following four RQs:

- **RQ1:** How effective is our anomaly detector as compared to ATTAIN and the other two baselines from the literature?
- **RQ2:** How effective is it for introducing CL in DTC?
- **RQ3:** Is LATTICE on par with the baselines in terms of required training time?
- **RQ4:** Is LATTICE on par with the baselines in terms of detection delay time?

With RQ1, we aim to compare the effectiveness of our approach with existing approaches from the literature (baselines). RQ2 focuses on evaluating the improvement brought by CL when it comes to DTC. In addition, we perform an ablation study to check the effectiveness of each difficulty measurer inside CL. RQ3 is designed to demonstrate the efficiency of LATTICE in terms of training time, whereas RQ4 attempts to show how quickly LATTICE can detect an anomaly.

6.2 Characteristics of the Datasets

We evaluate LATTICE with five CPS datasets, namely Secure Water Treatment (SWaT) [40], Water Distribution (WADI) [1], Battle Of The Attack Detection Algorithms (BATADAL) [53], PHM challenge 2015 dataset [26] and Gas Pipeline Dataset [41]. Table 2 provides key characteristics of these datasets.

Table 2. Characteristics of the Datasets. N_{sensor}, $N_{actuators}$, N_{train_attack}, N_{test_attack} and $\bar{N}_{attack\_len}$ denote the number of sensors, number of actuators, number of attacks in the training dataset, number of attacks in the test dataset, and the average length of attacks, respectively. The total number of samples is N.

Dataset	N_{sensor}	$N_{actuators}$	N_{train_attack}	N_{test_attack}	$\bar{N}_{attack\_len}$	N
SWaT	25	26	33	8	109	946722
WADI	42	61	12	3	665	1221372
BATADAL	26	17	5	2	70	12936
PHM 2015	4 - 16	4	9284-292064	2322-73016	2	79916-1152651
Gas Pipeline	16	7	51011	12753	4	236179

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. There are 41 attacks in total. We split these 41 attacks into training (33) and testing (8) datasets. The average length of an attack, i.e., the number of samples in the attack, is 109 in this dataset.

The second case study is the WADI data produced from the WADI testbed – an extension to the SWaT testbed. The main functionality of the WADI testbed is the secure distribution of water. The dataset generated from the WADI testbed has two weeks' normal operation data, whereas it has two days' attack data. This testbed consists of 42 sensors and 61 actuators. There are 15 attacks in total. We split these 15 attacks into two sets: 12 for training and 3 for testing. The average length of an attack is 665 in this dataset.

The BATADAL dataset is an extension of the WADI dataset. The attacks were designed for an attack detection competition. This testbed consists of 26 sensors and 17 actuators. There are 7 attacks in total, which are divided into 5 for training and 2 for testing. The average length of an attack is 70.

The PHM challenge 2015 dataset focuses on the operation of plants and the capability to detect failure events of the plants in advance. This dataset is a collection of data from 70 plants and the number of sensors and actuators varies from plant to plant. Therefore, we show the ranges (4-16 sensors and 4 actuators) of this dataset in Table 2. The number of attacks also varies from plant to plant. We split these attacks into two groups: 9284-292064 for training and 2322-72016 for testing. The average length of an attack is around 2, much smaller as compared to SWaT, WADI, and BATADAL.

The gas pipeline dataset was collected from a gas pipeline testbed, which transports gas from one place to another. There are 6376441 attacks in total, which are split into two groups: 51011 for training and 12753 for testing. The average length of an attack is 4.

6.3 Evaluation Metrics and Statistical Tests

This section presents the evaluation metrics used herein. For RQ1-RQ3, we introduce three metrics (precision, recall, and F1 score) for effectiveness evaluation in Section 6.3. In Section 6.3.2, we define a novel metric called Unit Training Time (UTT) for answering RQ4. In Section 6.3.3, we propose Detection Delay Time (DDT) for answering RQ4. In Section 6.3.4, we briefly introduce the employed statistical tests.

6.3.1 Metrics for effectiveness. LATTICE aims to provide practical information about anomalies to CPS. Such information includes a general indication of whether the target CPS is under attack and detailed information about the attack, e.g., starting time and duration. Correspondingly, we propose two types of metrics for RQ1-RQ3, namely coarse-grained and fine-grained effectiveness metrics.

Coarse-grained effectiveness metric. This metric provides general information about an attack, i.e., the existence of the attack. One practical usage scenario of LATTICE is that it can detect the existence of an attack without much precise information such as the exact starting and ending time for this attack. Inspired by this scenario, we propose *Anomaly Coverage Rate (ACR)* as in Equation 26

$$ACR = \frac{N_{detected}}{N_{total}} \tag{26}$$

where $N_{detected}$ denotes the number of attacks detected and N_{total} denotes the total number of attacks. We consider an attack detected if half of the attack instances are correctly classified. ACR takes a value between 0 and 1, where a higher value indicates a higher number of attacks are successfully detected, and vice versa.

Fine-grained effectiveness metric. Although the coarse-grained effectiveness metric can assess LATTICE's ability to detect anomalies in general, details of attacks are neglected by ACR. This motivates us to evaluate LATTICE with instance-wise metrics. Given the binary classification nature of anomaly detection, we use three standard classification metrics [8, 30, 65]: precision, recall, and F1 score. In our context, precision is the percentage of correctly detected anomaly instances among all the instances that are predicted as anomalies, while recall is the percentage of correctly detected anomaly instances among all the anomaly instances. The F1 score is the harmonic mean of precision and recall.

Formally, precision is defined in Equation 27:

$$precision = \frac{TP}{TP + FP} \tag{27}$$

where TP and FP stand for True Positive and False Positive, respectively. The recall is defined in Equation 28:

$$recall = \frac{TP}{TP + FN} \tag{28}$$

where FN denotes False Negative. F_1 is defined in Equation 29:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{29}$$

6.3.2 Unit Training Time (UTT).. We observe that the training time of LATTICE depends not only on the model itself, but also on the complexity of the CPS. Therefore, to evaluate the model's efficiency, we propose UTT, a CPS complexity-agnostic metric. Specifically, we first identify the complexity-sensitive execution time tt and divide it by the CPS complexity as in Equation 30.

$$utt = \frac{tt}{S} \tag{30}$$

Training time tt denotes the required convergence time of the model (Equation 31). We assume that the model converges when losses reach the minimum. In other words, losses from two neighboring batches should have minimal discrepancy (smaller than a threshold value δ, i.e. $\delta = 1e - 4$).

$$tt = time(convergence) - time(start) \tag{31}$$

S stands for the complexity of the CPS. Since there is no out-of-box complexity metric for our purpose, we propose the Metric of Complexity (MoC), which estimates the complexity of CPS from three aspects: CPS itself, attack, and dataset, as formalized in Equation 32.

$$S = S_{CPS} + S_{ATT} + S_{DAT} \tag{32}$$

CPS complexity. CPS complexity consists of static and automatic complexity (Equation 33). Static complexity entails a CPS's internal structure, e.g., the number of sensors and actuators. Such structure decides the dimension of LATTICE's input data in a fixed manner. However, not all the components in this structure are active during training. Hence, we propose automatic complexity, which identifies only active components during training.

$$S_{CPS} = S_{CPS.static} + S_{CPS.automatic} \tag{33}$$

For *static complexity*, we calculate the number of sensors and actuators in the CPS and normalize it as $z_{n\_s/a}$, allowing this metric to be calculated along with other metrics regardless of the scale of CPS. Moreover, we make a crucial observation that value changes in actuators usually induce changes in sensor values. Such correlation indicates dependencies between sensors and actuators. Therefore, we calculate sensors and actuators' ratio $z_{r\_s/a}$ in the CPS in addition to the absolute number $z_{n\_s/a}$. Finally, we add $z_{r\_s/a}$ and $z_{n\_s/a}$ together as the static complexity of CPS as shown in Equation 34.

$$S_{CPS.static} = z_{n\_s/a} + z_{r\_s/a} \tag{34}$$

However, not all the sensors and actuators are active during the process. As in the running example (Table 1), FIT101, LIT101, MV101, and P101 are only active in Phase 1 of the water treatment process and become inactive in other phases. Therefore, we propose *automatic complexity* for calculating only active sensors and actuators at runtime as in Equation 35. Similar to static complexity, we calculate both the number and ratio of active sensors and actuators. We then add the normalized number $z_{n\_s/a}$ and ratio of $z_{r\_s/a}$ together as in Equation 35.

$$S_{CPS.automatic} = d_{n\_s/a} + d_{r\_s/a} \tag{35}$$

Attack complexity. Attack complexity is computed based on the attack's information, e.g., the number of attacks in total. Formally, we calculate the attack complexity as in Equation 36, where $z_{n\_n/a}$ denotes the normalized number of attacks; $z_{r\_n/a}$ denotes the ratio of attack and normal samples; and z_{a_type} denotes the normalized number of different types of attacks.

$$S_{ATT} = z_{n\_n/a} + z_{r\_n/a} + z_{a\_type} \tag{36}$$

Dataset complexity. Dataset complexity reflects the scale of a given CPS dataset. Size is a good indicator of the scale. Therefore, we normalize the dataset size as z_{dat_size} and include it in the computation of dataset complexity. Another scale's indicator that we found is concept drift. Concept drift is a phenomenon where the target variable's statistical properties (e.g., mean/variance values of sensors) change over time [57]. As a result, dataset complexity increases in the presence of concept drift. According to Scheffel et al [48], this phenomenon is prevalent in the CPS domain, which motivates us to include concept drift in dataset complexity computation. To calculate concept drift, we take inspiration from Adaptive Sliding Window (ADWIN) [12] and propose a novel concept drift complexity metric S_{dat_drift}. In conclusion, dataset complexity consists of information about dataset size and concept drift as in Equation 37.

$$S_{DAT} = z_{dat\_size} + S_{dat\_drift} \tag{37}$$

In detail, concept drift complexity is calculated as in Equation 38. s_p and s_q are two random samples selected, respectively from two separate windows W_i and W_j (Equation 39). Let sample size be N, we calculate the KL divergence of each pair of samples and calculate the average value for all the N pairs.

$$S_{dat\_drift} = \sum_{p=1,q=1}^{N,N} KL(s_p, s_q)/N \tag{38}$$

$$s_p, s_q = sample(W_i), sample(W_j) \tag{39}$$

6.3.3 Detection delay time (DDT). As for DDT, we aim to assess how good LATTICE is at detecting anomalies at early stages to prevent further damages. To this end, we count false negative samples N_{FN} at the beginning of an attack which consists of N samples. We then calculate DDT by dividing N_{FN} by N as in Equation 40.

$$s_{ddt} = \frac{N_{FN}}{N} \tag{40}$$

DDT indicates the delay of the assessed method when trying to detect an anomaly. It takes a value between 0 and 1. A lower DDT value means that the assessed method can detect attacks at an earlier stage.

6.3.4 Statistical Testing. Due to the inherent randomness of our approach, we employ statistical testing to answer RQ1-RQ4 to determine whether the improvements are statistically significant.

In this paper, we use **Mann-Whitney U test**, which, unlike $t-$test, makes no assumption on the underlying data distribution. We test all the pair-wise comparisons in each RQ. In general, to compare *Method A* and *Method B*, we run each method 30 times, as suggested in [3]. The null hypothesis is that there is no statistical difference between the two methods. If the null hypothesis is rejected, we conclude that *Method A* and *Method B* are not equivalent.

Mann-Whitney U test results reveal the significant difference between *Method A* and *Method B*, whereas the magnitude of this difference is unknown. Such magnitude can be assessed with an effect size. In this paper, we use **Vargha and Delaney's A12** , which also requires no knowledge of the underline data distribution. An A12 value ranges from 0 to 1. If $A12 = 0.5$, it means that the results are obtained by chance. If $A12 > 0.5$, it means *Method A* has a higher chance of getting better results than *Method B*, and vice versa.

Concretely in this paper, RQ1 and RQ4 involve comparisons between LATTICE (*Method A*) with the baselines (*Method B*). RQ2 involves comparisons between LATTICE (*Method A*) and LATTICE-DTM (*Method B*). RQ3 performs the ablation study by comparing LATTICE (*Method A*) with LATTICE-PDM, LATTICE-CEM, LATTICE-HDM,LATTICE-CEM-HDM, and LATTICE-CL (*Method B*).

6.4 Parameter Settings

We use cross-validation to automatically select the hyperparameters of LATTICE. Cross-validation is a commonly used hyperparameter tuning technique in machine learning [29]. The general idea is to train machine learning models with different parameter settings and find out optimal ones that yield the best performance. Specifically for each hyperparameter set, we split each dataset into training and testing datasets. The training dataset is then further split into ten chunks. We select one chunk as a validation dataset each time, while the remaining nine chunks are used as training datasets. We perform ten such validation processes and calculate an average F1 score as a conclusion. We compare each hyperparameter set's F1 score and find the optimal set based on the comparison result.

We divide LATTICE's hyperparameters into two groups: respectively for ATTAIN and CL. We show our optimal hyperparameters found by cross-validation as follows:

- **ATTAIN hyperparameters.** We set the batch size of input data as 64. 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.
- **CL hyperparameters.** We set the threshold value of the baby step algorithm to 0.8. We use the min-max scaler to calculate the difficulty measurer. Min and max values are calculated with the historical data we have for now. We plan to substitute this with more generic scalers in the future when needed.

6.5 Experiment Execution

In this paper, neural network layers were built with the PyTorch framework [42], and the GCN layer was constructed with the PyTorch Geometric (PyG) framework [17]. 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 30 times, and the average results were reported in this paper. For the SWaT dataset, we followed previous works [30] 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.

7 RESULTS AND ANALYSIS

Section 7.1-Section 7.4 present the results for RQ1-RQ4, respectively.

7.1 Results and Analysis for RQ1

Table 3 shows the **ACR** results of LATTICE and the baselines (i.e., LSTM-CUSUM [18], MAD-GAN [34], and ATTAIN [63]). Both LSTM-CUSUM and MAD-GAN were designed for CPS anomaly detection (see Section 3 for more details). ATTAIN is our previous work, which is extended in this paper by introducing CL. We evaluate these methods with ACR on the five public datasets (Section 6.2). Anomalies in SWaT, WADI, and BATADAL datasets have a longer duration and fewer occurrences than the PHM 2015 and Gas Pipeline datasets. We can observe that all the methods achieve 100% on SWaT, WADI, and BATADAL, which means that they never miss any anomaly. Contrarily, ACR values on the PHM 2015 and Gas Pipeline dataset are lower due to the anomalies' shorter duration and higher occurrences. LATTICE achieves the highest ACR values on these two datasets, demonstrating its superior capability at detecting anomalies in a coarse-grained manner.

Dataset	LSTM-CUSUM	MAD-GAN	ATTAIN	LATTICE
SWaT	1	1	1	1
WADI	1	1	1	1
BATADAL	1	1	1	1
PHM2015	0.79	0.82	0.87	0.87
Gas Pipeline	0.77	0.81	0.85	0.91

Table 3. ACR results of LATTICE and the baselines

Figure 5 shows the fine-grained effectiveness metric results of LATTICE. Three boxplots in each row demonstrate the precision, recall, and F1 score on a specific dataset. We can observe that LATTICE outperforms LSTM-CUSUM and MAD-GAN by a large margin except for recall on WADI and BATADAL. LATTICE manages to further improve ATTAIN in all five datasets. For the SWaT dataset, the improvement is slight, but we argue that this is because the precision, recall, and F1 scores are already high, and there is not much room for further improvement at the first place. Also, we can observe that, as compared to the baselines, the results of LATTICE have smaller variances, indicating that LATTICE is more certain about its prediction.

As mentioned in Section 6.3, we repeated all the experiments 30 times and performed statistical tests. Table 4 shows the results of comparing LATTICE with each baseline. We consider $p - value < 0.01$ as a significant difference. In terms of precision, LATTICE is significantly better than LSTM-CUSUM (5/5), MAD-GAN (5/5), and ATTAIN (4/5). The minimum effect size for all the comparisons is 0.701. In terms of recall, LATTICE is significantly better than LSTM-CUSUM (5/5), MAD-GAN (3/5), and ATTAIN (4/5), while the minimum effect size is 0.771. Regarding the F1 score, LATTICE is significantly better than all the baselines for all the datasets with the minimum effect size of 0.713.

Conclusion: RQ1

We conclude that LATTICE outperforms the baselines in terms of both the coarse-grained (i.e., ACR) and fine-grained effectiveness metrics. Statistical test results show that this improvement is significant, and LATTICE is more likely to yield better results. In short, our anomaly detector is more effective than the baseline methods in the literature.

7.2 Results and Analysis for RQ2

The introduction of CL is the major contribution of LATTICE. To better understand CL's influence, we evaluate the effectiveness of CL and perform an ablation study in this section. **CL effectiveness.** To evaluate CL's effectiveness, we compare LATTICE and LATTICE-CL, which removes CL from LATTICE. Figure 6 presents the box plots of LATTICE and LATTICE-CL. We see a great performance decline when removing CL from LATTICE. LATTICE-CL's variance is also larger than LATTICE, indicating higher prediction uncertainty.

We also report the statistical test results in Table 5. The last column compares LATTICE with LATTICE-CL. CL's improvement on all five datasets in precision, recall, and F1 score is significant ($p - value < 0.05$), except for recall on WADI. The corresponding effect sizes are also strong ($minimum = 0.6944$), indicating that LATTICE has a higher probability of yielding better results than LATTICE-CL.

Ablation study In LATTICE, we explore three types of difficulty measurers: PDM, CEM, and HDM. PDM is a predefined difficulty measurer, while CEM and HDM are automatic difficulty measurers focusing on entropy and vector distance, respectively. We argue that each difficulty measurer contributes to the performance improvement

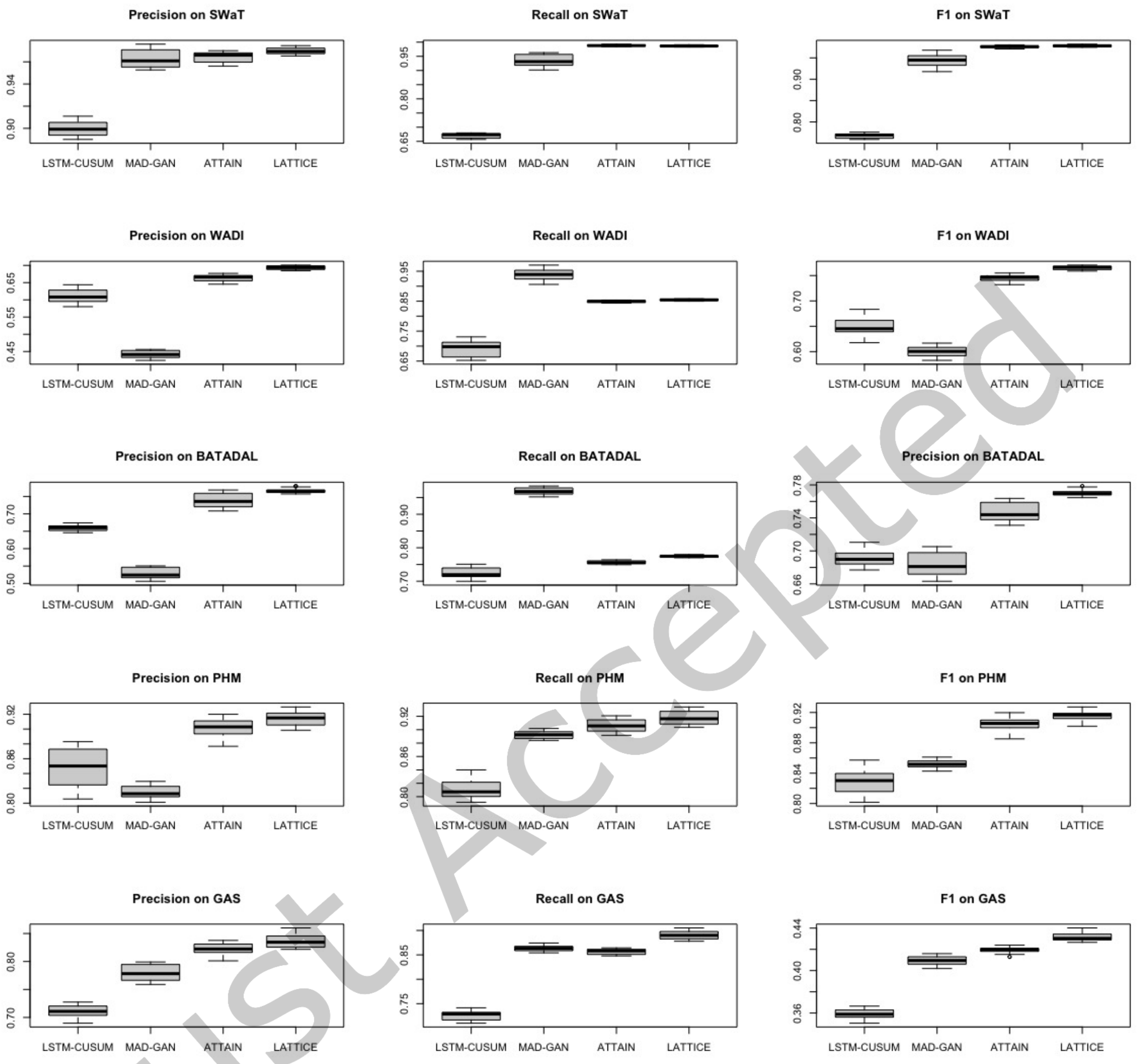

Fig. 5. Effectiveness boxplots of LATTICE and the baselines

of LATTICE. Therefore it is valuable to assess the effectiveness of each difficulty measurer. To that end, we perform an ablation study of each difficulty measurer.

- **PDM effectiveness.** LATTICE-PDM removes the PDM difficulty measurer from LATTICE. Figure 6 shows that removing PDM decreases recall on all five datasets. The precision on SWaT, BATADAL, and PHM 2015 also decreases while the precision on WADI and Gas Pipeline increases. Mann-whitney U-test results (Table 5) show that the decreases are significant ($p - value < 0.05$) except for recall on WADI. The effect size of the F1 score is very strong on SWaT, BATADAL, and PHM 2015 datasets.

Table 4. Statistical test results of comparing LATTICE and the baselines

Datasets	Metrics	Testing	LSTM-CUSUM	MAD-GAN	ATTAIN
SWaT	Precision	p-value	<1e-4	9.518e-4	**0.1294**
		A12	1.0	0.701	0.846
	Recall	p-value	<1e-4	<1e-4	**0.0523**
		A12	1.0	1.0	0.336
	F1	p-value	<1e-4	<1e-4	1.13e-2
		A12	1.0	1.0	0.713
WADI	Precision	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	1.0	1.0
	Recall	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	0	0.875
	F1	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	1.0	1.0
BATADAL	Precision	p-value	<1e-4	<1e-4	1.684e-4
		A12	1.0	1.0	0.886
	Recall	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	0	1.0
	F1	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	1.0	1.0
PHM2015	Precision	p-value	<1e-4	<1e-4	2.833e-4
		A12	1.0	1.0	0.791
	Recall	p-value	<1e-6	<1e-4	1.886e-4
		A12	1.0	1.0	0.771
	F1	p-value	<1e-6	<1e-4	3.79e-6
		A12	1.0	1.0	0.872
Gas Pipeline	Precision	p-value	<1e-4	<1e-4	1.886e-4
		A12	1.0	1.0	0.811
	Recall	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	1.0	1.0
	F1	p-value	<1e-4	<1e-4	<1e-4
		A12	1.0	1.0	1.0

- **CEM effectiveness.** LATTICE-CEM removes the CEM difficulty measurer from LATTICE. We can observe from Figure 6 that removing CEM decreases precision, recall, and F1 score in all cases. Mann-whitney U-test results demonstrate these decreases' significance ($p - value < 0.05$) except for precision on the gas pipeline dataset. The effect sizes are strong with a minimum of 0.7189 (F1 score on gas pipeline dataset).
- **HDM effectiveness.** LATTICE-HDM removes the HDM difficulty measurer from LATTICE. Similar to LATTICE-CEM, LATTICE-HDM also suffers a decline in precision, recall, and F1 score in all the cases. Mann-whitney U-test results demonstrate these decreases' significance ($p - value < 0.05$) except for precision on the gas pipeline dataset. The effect sizes are strong with a minimum of 0.7211 (recall score on PHM 2015 dataset).
- **DTM effectiveness.** LATTICE-CEM-HDM removes the CEM and HDM difficulty measurers, which are computed with DTM information. The comparison between LATTICE and LATTICE-CEM-HDM shows the

effectiveness of introducing DTM in CL. We can observe from Figure 6 that LATTICE-CEM-HDM further decreases precision, recall and F1 score in all the cases. All the p-values are smaller than 0.05 except for recall on WADI datasets, while the corresponding effect sizes are strong (*minimum* = 0.7389).

Conclusion: RQ2

We conclude that CL is effective in improving the effectiveness of DTC, and each difficulty measurer (i.e., PDM, CEM, or HDM) contributes to this improvement.

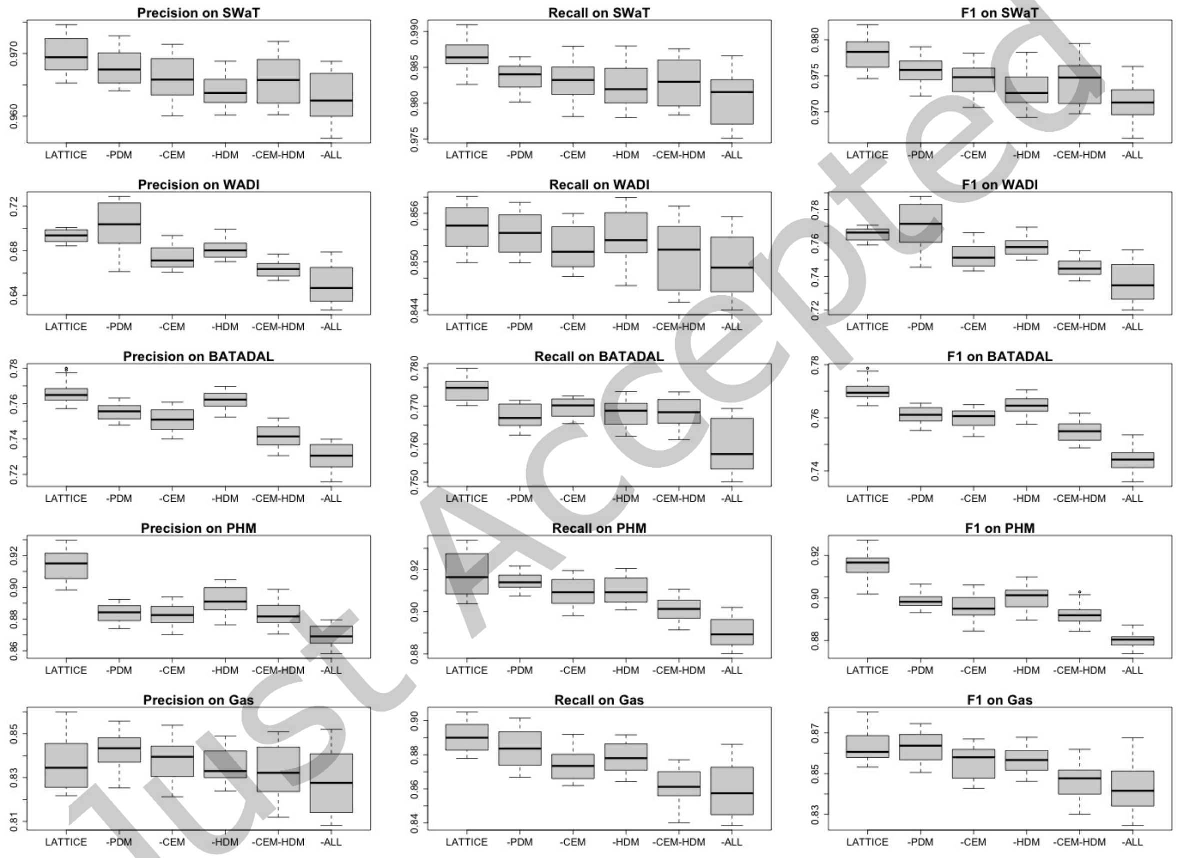

Fig. 6. Boxplots of the ablation study experiment results

7.3 Results and Analysis for RQ3

As we target large-scale real-world CPS, efficiency is crucial for LATTICE. As mentioned in Section 6.3, we use UTT to evaluate the efficiency of LATTICE. Table 6 shows the results of the UTT of each method, along with the complexity of each CPS. We can observe that LATTICE improves ATTAIN by 4.2% ($\frac{9.513-9.114}{9.513}$) in terms of UTT. LSTM-CUSUM takes the least time among all four methods. However, as reported in RQ1-RQ3, LSTM-CUSUM is the least effective method among all. Table 6 further shows that LATTICE takes less time on the SWaT, WADI,

Table 5. Statistical test results of the ablation study

Datasets	Metrics	Testing	-PDM	-CEM	-HDM	-CEM-HDM	-ALL
SWaT	Precision	p-value	0.0128	<1e-4	<1e-4	2e-04	<1e-4
		A12	0.6811	0.78	0.9344	0.7733	1
	Recall	p-value	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.8622	0.87	0.8544	0.7922	1
	F1	p-value	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.81	0.8778	0.94	0.85	1
WADI	Precision	p-value	0.0405	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.3578	0.94	0.8589	1	1
	Recall	p-value	0.2449	0.0024	0.1347	0.0024	0.4161
		A12	0.5611	0.74	0.58	0.7389	0.4444
	F1	p-value	0.0667	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.3722	0.9433	0.8711	1	1
BATADAL	Precision	p-value	<1e-4	<1e-4	0.0054	<1e-4	<1e-4
		A12	0.9444	0.9844	0.6944	1	1
	Recall	p-value	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.9522	0.8711	0.9167	0.8911	1
	F1	p-value	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.9911	0.9978	0.87	1	1
PHM 2015	Precision	p-value	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
		A12	1	1	0.9722	0.9989	1
	Recall	p-value	0.1142	<1e-4	0.0017	<1e-4	<1e-4
		A12	0.5756	0.7567	0.7211	0.9178	0.9978
	F1	p-value	<1e-4	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.9944	0.9922	0.9689	0.9989	1
Gas Pipeline	Precision	p-value	0.0473	0.685	0.6702	0.3184	0.0011
		A12	0.3311	0.4444	0.4856	0.57	0.6944
	Recall	p-value	0.0277	<1e-4	<1e-4	<1e-4	<1e-4
		A12	0.7122	0.9178	0.8389	1	0.9633
	F1	p-value	0.9193	0.001	<1e-4	<1e-4	<1e-4
		A12	0.4989	0.7189	0.7456	0.9456	0.9722

and BATADAL datasets, while it takes more time on the PHM 2015 and Gas pipeline datasets. However, we argue that this increase is small and caused by the anomalies' short duration in these two datasets. Short anomalies present smaller complexity, which can train a simpler model quickly. LATTICE is more complicated, as compared to ATTAIN and MAD-GAN, hence the longer UTT.

Similar to RQ1-RQ3, we also collected the execution time of 30 runs. We calculate the UTT of these runs and performed the Mann-Whitney U-test, the results of which are reported in Table 7. We can observe that LSTM-CUSUM takes significantly less time than LATTICE ($p - value < \alpha$) on all the five datasets, but we exclude it from this comparison due to its low effectiveness. LATTICE takes significantly less time than MAD-GAN and ATTAIN ($p - value < \alpha$) on all the datasets except for the PHM2015 and Gas Pipeline datasets. In terms of A12, the results between LATTICE and ATTAIN are also strong (with the maximum being 0.303) except on the

Table 6. Results of efficiency (measured in UTT) of LATTICE and the baselines

Datasets	Complexity	LSTM-CUSUM	MAD-GAN	ATTAIN	LATTICE
SWaT	0.436	3.050	8.628	8.817	8.514
WADI	0.771	4.901	14.302	15.581	12.134
BATADAL	0.472	4.544	8.280	8.22	7.989
PHM2015	0.180	4.944	5.533	6.117	8.539
Gas Pipeline	0.479	5.282	8.591	8.	8.401
Average	0.468	4.544	9.067	9.513	9.114

PHM2015 dataset (1.0). The results between LATTICE and MAD-GAN are strong (with the maximum being 0.341) except for PHM2015 (1.0) and Gas Pipeline (0.432).

Table 7. Statistical test results of Unit Training Time (UTT) of LATTICE and the baselines

Datasets	Testing	LSTM-CUSUM	MAD-GAN	ATTAIN
SWaT	p-value	<1e-6	8.705e-2	<1e-6
	A12	1.0	0.341	0.0
WADI	p-value	<1e-6	<1e-6	<1e-6
	A12	1.0	0.0	0.0
BATADAL	p-value	<1e-6	3.239e-6	4.968e-5
	A12	1.0	0.122	0.221
PHM2015	p-value	<1e-6	<1e-6	<1e-6
	A12	1.0	1.0	1.0
Gas Pipeline	p-value	<1e-6	**0.3492**	7.296e-4
	A12	1.0	**0.432**	0.303

Conclusion: RQ3

LATTICE generally reduces UTT compared to the baselines. Therefore, we conclude that LATTICE is on par with the baselines in terms of efficiency.

7.4 Results and Analysis for RQ4

RQ4 aims to evaluate how early LATTICE can detect an anomaly. We use DDT as the evaluation metric (Section 6.3.3). Table 8 shows the detection time for LATTICE and the baselines on the SWaT, WADI, and BATADAL datasets. We do not include the PHM 2015 and Gas Pipeline datasets in this RQ because their average anomaly lengths are quite small (2 and 4, respectively). DDT degrades to the coarse-grained effectiveness metric for short anomaly detection since a large DDT is equivalent to missing an anomaly.

We present Mann-whitney U-test results and effect sizes in Table 7. Compared to LSTM-CUSUM, LATTICE's DDT decreases are all significant ($p - value < 0.05$) and all effect sizes are strong ($A12 = 1$). Compared to MAD-GAN, the decreases in WADI and BATADAL are significant ($p - value < 0.05$), and effect sizes are strong ($A12 = 1$ and $A12 = 0.88$ respectively). However, the SWaT dataset's decrease is not significant ($p-value = 0.2621$). Similarly, we observe significant differences for the WADI and BATADAL datasets ($p - value < 0.05$) and no significant differences for the SWaT dataset ($p - value = 0.3931$) when comparing LATTICE and ATTAIN. The effect sizes on WADI and BATADAL are 0.6578 and 0.9456, respectively, indicating that LATTICE is more likely to detect anomalies faster on these two datasets.

Dataset	LSTM-CUSUM	MAD-GAN	ATTAIN	LATTICE
SWaT	0.2%	0.1%	0.1%	0.1%
WADI	6%	5.7%	5.3%	5.1%
BATADAL	4.7%	4.5%	4.5%	4.1%

Table 8. Results of detection delay time

Dataset	Testing	LSTM-CUSUM	MAD-GAN	ATTAIN
SWaT	p-value	<1e-4	0.2621	0.3931
	A12	1	0.6133	0.4711
WADI	p-value	<1e-4	<1e-4	0.0185
	A12	1	1	0.6578
BATADAL	p-value	<1e-4	<1e-4	<1e-4
	A12	1	0.88	0.9456

Table 9. Statistical testing results of detection delay time

Conclusion: RQ4

LATTICE reduces detection delay on all three datasets when compared with the baselines. Therefore, we conclude that LATTICE is on par with the baselines in terms of detection delay time.

8 OVERALL DISCUSSION

As discussed in the previous sections, LATTICE benefits from CL and digital twin, which allow it to be more effective in anomaly detection. Below, we discuss, in detail, the plausible reasons for LATTICE achieving the effectiveness improvement.

Advantage of the predefined difficulty measurers. In Section 5.1.1, we propose a predefined difficulty measurer, consisting of complexity, diversity, noise, and vulnerability. We conducted surveys with these four measurers and found their correlation with the attack labels. We believe capturing this correlation is one of the potential reasons for performance improvement [33]. We use the Spearman correlation coefficient test for studying the correlation of attack labels with the four measurers. This is a non-parametric test, which is calculated based on the rankings of two variables. The correlation coefficient tends to be high if observations from the two variables have a similar rank position and vice versa. The coefficient value ranges between -1 and 1. Figure 7 shows the correlation graph between diversity and the attack labels on the SWaT, WADI, BATADAL, and Gas Pipeline datasets. We make a subset containing at least one complete anomaly for each dataset. Correlations are calculated with these subsets instead of the whole datasets for illustrative purpose. We find that vulnerability (the last column of Figure 7) has the highest correlation with the attack labels, which is expected because vulnerability is calculated based on the time difference with the attack. Among the other three measurers, diversity has the highest correlation. We find higher correlations in the SWaT, WADI, and BATADAL datasets, while the correlations in the PHM 2015 challenge and Gas pipeline datasets are weaker. We believe this is because the

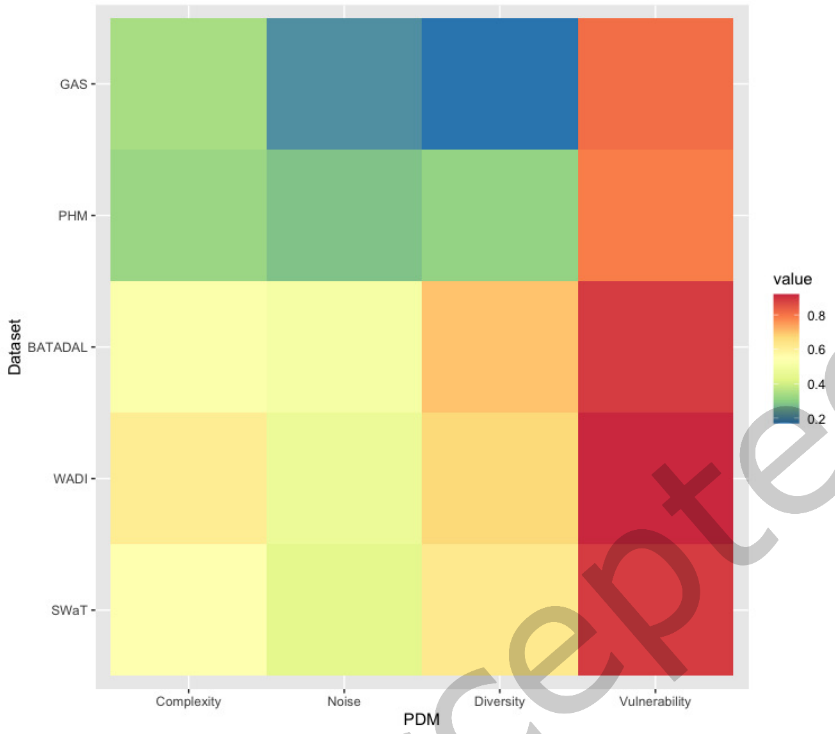

Fig. 7. Spearman Correlations between the attack labels and the difficulty measurers

average attack length is much smaller in these two datasets than in the others. Average attack lengths in the PHM 2015 challenge and Gas pipeline datasets are only 2 and 4, respectively. Spearman correlation can hardly capture correlations for such short attacks. However, the high correlations in SWaT, WADI, and BATADAL are sufficient to support our hypothesis that the diversity is related to the attack labels.

Advantage of the automatic difficulty measurers. We successfully adapted CL to time series data by introducing the contextualized difficulty measurer. Most existing CL approaches use context-free difficulty measurers, implying that the difficulty score of each sample is assigned based on its generic characteristics such as diversity, noise level, and intensity. These context-free difficulty measurers are sufficient for classification tasks in the natural language processing and image recognition domains, while the classification of time series data requires consecutive data for reserving chronological characteristics [28, 36]. In this paper, we take advantage of DTM and propose two automatic difficulty measurers: CEM and HDM. CEM and HDM successfully incorporate context information into the difficulty scores of each sample. The ablation study in Section 7.2 shows the improvement brought by these two contextualized difficulty measurers. Figure 6 shows a significant performance drop between LATTICE and LATTICE-CEM-HDM.

Advantage of CL's optimization principle. Another reason for the improvement is the optimization principle of CL. Bengio et al. [6] pointed out that CL can be seen as an optimization strategy for non-convex functions. Such a strategy first optimizes a smoother version of the problem to reveal the global picture and then gradually considers less smoothing versions until the target objective of interest is reached. In our case, the introduction of CL potentially prevents our method from getting stuck at a local optimum. As we can observe in the example

(Table 1), the first attack starts at 10:29:14 after 29 minutes of normal operation. Consequently, we acquire far more normal data than anomaly data, inducing machine learning methods to a local optimum. CL, however, alleviates this problem by re-ordering batches based on difficulty scores. Particularly, difficult samples are fed into our model gradually, presenting a smoother optimization problem. CL enables LATTICE to learn the global picture instead of being stuck at a local optimum while speeding up the whole training process.

9 THREATS TO VALIDITY

We identify four common types of threats to the validity of our experiments, as discussed below.

Conclusion Validity. We evaluate our method with metrics such as precision, recall, F1, UTT, and DDT. Precision, recall, and F1 are commonly used in classification tasks. However, other metrics, such as ROC and false positive rate, could also be adopted for our evaluation. This could pose a threat to the conclusion validity. These metrics are less commonly used and evaluate the effectiveness of models from similar perspectives as precision, recall, and F1. We will include these metrics in the future if needed.

Another threat could be that we performed the statistical tests with the Mann-Whitney U-test on samples with a sample size of 30. We are aware that larger sample sizes are always preferred, which however comes with a cost. In the future, when more resources are available, we will conduct experiments with larger sample sizes.

Construct Validity. We also empirically studied whether LATTICE benefits from CL and digital twin for CPS anomaly detection. To that end, we designed LATTICE with digital twin trained with specific CL strategies. However, we are aware that there are other options for digital twin and CL design. Therefore, more experiments are needed to try out various ways of constructing digital twin and different CL strategies, to know better about the potential of CL and digital twin contributing to the performance of LATTICE.

Internal Validity. We compared LATTICE with three baselines: LSTM-CUSUM, MAD-GAN, and ATTAIN, which are the state-of-art in anomaly detection. But we notice that there are other models that can potentially outperform these baselines, such as variational autoencoder and deep belief networks. In the future, we will choose more representative models and conduct more experiments.

External Validity. Our experiments were performed on five testbeds, which are scaled datasets from real operating CPS. Without conducting experiments with real CPS and on different types of CPS, we cannot generalize the performance of LATTICE. However, we would like to point out that, in our context, to conduct experiments with real CPS, LATTICE needs to get connected to them during their operations and obtain their operating data at runtime. Setting up this kind of experiment is complicated and expensive. We plan to work with our collaborators and apply our method in real-world CPS in the future.

10 CONCLUSION AND FUTURE WORK

LATTICE is a novel method, which combines both digital twin and curriculum learning (CL) to address the anomaly detection challenge of CPS. LATTICE extends our previous work ATTAIN. ATTAIN consists of a timed automaton-based digital twin model and a GAN-based digital twin capability. 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 extended ATTAIN by introducing CL to optimize its training process, which forms LATTICE. We performed extensive experiments with LATTICE on five CPS datasets. Experiment results show the performance superiority of LATTICE in comparison to two state-of-art anomaly detectors and ATTAIN, increasing by 0.906%, 2.363%, 2.712%, 2.008%, 2.367% on the SWAT, WADI, BATADAL, PHM Challenge 2015, and Gas Pipeline datasets, respectively. We also demonstrated the effectiveness of CL and the different difficulty measurers with the ablation study. Finally, we demonstrated that LATTICE is on par with the baselines in terms of the training time and detection delay time.

In the future, we plan to conduct more experiments on real-world CPS of various domains to evaluate the scalability and generalization of LATTICE. We will also consider exploring digital twins for more challenging tasks, such as detecting attacks targeting multiple CPS simultaneously which requires developing an integrated digital twin model.

ACKNOWLEDGEMENT

The SWaT, WADI, and BATADAL datasets were provided by iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design. The research presented in this paper has benefited from the Experimental Infrastructure for Exploration of Exascale Computing (eX3), which is financially supported by the Research Council of Norway under contract 270053. Qinghua Xu is supported by a project funded by the Norwegian Ministry of Education and Research. Shaukat Ali and Tao Yue are supported by Horizon 2020 project ADEPTNESS (871319) funded by the European Commission.

REFERENCES

[1] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. 2017. 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*. 25–28.

[2] Bernhard K Aichernig, Andrea Pferscher, and Martin Tappler. 2020. From Passive to Active: Learning Timed Automata Efficiently BT - NASA Formal Methods, Ritchie Lee, Susmit Jha, and Anastasia Mavridou (Eds.). Springer International Publishing, Cham, 1–19.

[3] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. *Proceedings - International Conference on Software Engineering* (2011), 1–10. https://doi.org/10.1145/1985793.1985795

[4] Ayan Banerjee, Krishna K Venkatasubramanian, Tridib Mukherjee, and Sandeep Kumar S Gupta. 2011. Ensuring safety, security, and sustainability of mission-critical cyber–physical systems. *Proc. IEEE* 100, 1 (2011), 283–299.

[5] Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. 2017. CVAE-GAN: fine-grained image generation through asymmetric training. In *Proceedings of the IEEE international conference on computer vision*. 2745–2754.

[6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/1553374.1553380

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/1553374.1553380

[8] Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. 2019. Multi-head CNN–RNN for multi-time series anomaly detection: An industrial case study. *Neurocomputing* 363 (2019), 246–260.

[9] Xinlei Chen and Abhinav Gupta. 2015. Webly supervised learning of convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision* 2015 International Conference on Computer Vision, ICCV 2015 (2015), 1431–1439. https://doi.org/10.1109/ICCV.2015.168 arXiv:1505.01554

[10] Jaehoon Choi, Minki Jeong, Taekyung Kim, and Changick Kim. 2019. Pseudo-labeling curriculum for unsupervised domain adaptation. *arXiv preprint arXiv:1908.00262* (2019).

[11] Volkan Cirik, Eduard Hovy, and Louis-Philippe Morency. 2016. Visualizing and understanding curriculum learning for long short-term memory networks. *arXiv preprint arXiv:1611.06204* (2016).

[12] Lei Du, Qinbao Song, and Xiaolin Jia. 2014. Detecting concept drift: An information entropy based method using an adaptive sliding window. *Intelligent Data Analysis* 18, 3 (2014), 337–364. https://doi.org/10.3233/IDA-140645

[13] Matthias Eckhart and Andreas Ekelhart. 2018. Securing Cyber-Physical Systems through Digital Twins. *Ercim News* 115 (2018), 22–23.

[14] Matthias Eckhart and Andreas Ekelhart. 2018. Towards security-aware virtual environments for digital twins. In *Proceedings of the 4th ACM workshop on cyber-physical system security*. 61–72.

[15] Matthias Eckhart and Andreas Ekelhart. 2019. *Security and Quality in Cyber-Physical Systems Engineering*. Number March 2020. https://doi.org/10.1007/978-3-030-25312-7

[16] Rasheed El-Bouri, David Eyre, Peter Watkinson, Tingting Zhu, and David Clifton. 2020. Student-teacher curriculum learning via reinforcement learning: predicting hospital inpatient admission location. In *International Conference on Machine Learning*. PMLR, 2848–2857.

[17] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[18] Jonathan Goh, Sridhar Adepu, Marcus Tan, and Zi Shan Lee. 2017. Anomaly detection in cyber physical systems using recurrent neural networks. *Proceedings of IEEE International Symposium on High Assurance Systems Engineering* March 2019 (2017), 140–145. https://doi.org/10.1109/HASE.2017.36

[19] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems* 3, January (2014), 2672–2680. arXiv:arXiv:1406.2661v1

[20] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *international conference on machine learning*. PMLR, 1311–1320.

[21] Guy Hacohen and Daphna Weinshall. 2019. On the power of curriculum learning in training deep networks. *36th International Conference on Machine Learning, ICML 2019* 2019-June (2019), 4483–4496. arXiv:1904.03626

[22] Liping Han, Shaukat Ali, Tao Yue, Aitor Arrieta, and Maite Arratibel. 2022. *Uncertainty-aware Robustness Assessment of Industrial Elevator Systems*. Technical Report.

[23] Liping Han, Tao Yue, Shaukat Ali, Aitor Arrieta, and Maite Arratibel. 2022. Are Elevator Software Robust against Uncertainties? Results and Experiences from an Industrial Case Study. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1331–1342. https://doi.org/10.1145/3540250.3558955

[24] Abdul Jabbar, Xi Li, and Bourahla Omar. 2021. A survey on generative adversarial networks: Variants, applications, and training. *ACM Computing Surveys (CSUR)* 54, 8 (2021), 1–49.

[25] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*. PMLR, 2304–2313.

[26] Neil Eklund Justinian Rosca justinian, Nicholas Williard and Zhen Song. 2021. *PHM Data Challenge*. Retrieved November 12, 2021 from https://phmsociety.org/conference/annual-conference-of-the-phm-society/annual-conference-of-the-prognostics-and-health-management-society-2015/phm-data-challenge-3/

[27] Tom Kocmi and Ondrej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. *arXiv preprint arXiv:1707.09533* (2017).

[28] Allison Koenecke and Amita Gajewar. 2019. Curriculum learning in deep neural networks for financial forecasting. In *Workshop on Mining Data for Financial Applications*. Springer, 16–31.

[29] Ron Kohavi et al. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, Vol. 14. Montreal, Canada, 1137–1145.

[30] Moshe Kravchik and Asaf Shabtai. 2018. Detecting cyber attacks in industrial control systems using convolutional neural networks. *Proceedings of the ACM Conference on Computer and Communications Security* (2018), 72–83. https://doi.org/10.1145/3264888.3264896 arXiv:arXiv:1806.08110v2

[31] Kai A Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition* 110, 3 (2009), 380–394. https://doi.org/10.1016/j.cognition.2008.11.014

[32] M Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. *Advances in neural information processing systems* 23 (2010), 1189–1197.

[33] Sunil Kumar and Ilyoung Chong. 2018. Correlation analysis to identify the effective data in machine learning: Prediction of depressive disorder and emotion states. *International journal of environmental research and public health* 15, 12 (2018), 2907.

[34] Zhe Li, Jingyue Li, Yi Wang, and Kesheng Wang. 2019. A deep learning approach for anomaly detection based on SAE and LSTM in mechanical equipment. *The International Journal of Advanced Manufacturing Technology* 103, 1-4 (2019), 499–510.

[35] Ying Liu, Lin Zhang, Yuan Yang, Longfei Zhou, Lei Ren, Fei Wang, Rong Liu, Zhibo Pang, and M Jamal Deen. 2019. A novel cloud-based framework for the elderly healthcare services using digital twin. *IEEE Access* 7 (2019), 49088–49101.

[36] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng Yao. 2021. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.

[37] Qingming Ma, Bryan Burns, Krishna Narayanaswamy, Vipin Rawat, and Michael Chuong Shieh. 2011. Network attack detection using partial deterministic finite automaton pattern matching. US Patent 7,904,961.

[38] Tao Ma, Shaukat Ali, Tao Yue, and Maged Elaasar. 2019. Testing self-healing cyber-physical systems under uncertainty: a fragility-oriented approach. *Software Quality Journal* 27, 2 (2019), 615–649.

[39] Alexander Maier. 2014. Online Passive Learning of Timed Automata for Cyber-Physical Production Systems. (2014).

[40] Aditya P Mathur and Nils Ole Tippenhauer. 2016. 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, 31–36.

[41] Thomas H. Morris, Zach Thornton, and Ian P. Turnipseed. 2015. Industrial Control System Simulation and Data Logging for Intrusion Detection System Research.

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. 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., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[43] Gustavo Penha and Claudia Hauff. 2020. Curriculum Learning Strategies for IR: An Empirical Study on Conversation Response Ranking. *Advances in Information Retrieval* 12035 (2020), 699.

[44] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M. Mitchell. 2019. Competence-based curriculum learning for neural machine translation. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1 (2019), 1162–1172. https://doi.org/10.18653/v1/n19-1119 arXiv:1903.09848

[45] Shivesh Ranjan and John HL Hansen. 2017. Curriculum learning based approaches for noise robust speaker recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26, 1 (2017), 197–210.

[46] Douglas L.T. Rohde and David C. Plaut. 1999. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition* 72, 1 (1999), 67–109. https://doi.org/10.1016/S0010-0277(99)00031-1

[47] T D Sanger. 1994. Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation* 10, 3 (jun 1994), 323–333. https://doi.org/10.1109/70.294207

[48] Roberto M. Scheffel and Antônio A. Fröhlich. 2019. Increasing sensor reliability through confidence attribution. *Journal of the Brazilian Computer Society* 25, 1 (2019). https://doi.org/10.1186/s13173-019-0094-6

[49] Joel L Schiff. 2011. *Cellular automata: a discrete view of the world*. Vol. 45. John Wiley & Sons.

[50] Oliver G Selfridge, Richard S Sutton, and Andrew G Barto. 1985. Training and Tracking in Robotics.. In *Ijcai*. Citeseer, 670–672.

[51] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2021. Curriculum Learning: A Survey. 14, 8 (2021), 1–29. arXiv:2101.10382 http://arxiv.org/abs/2101.10382

[52] Valentin I Spitkovsky and Daniel Jurafsky. 2010. From Baby Steps to Leapfrog : How “ Less is More ” in Unsupervised Dependency Parsing. June (2010), 751–759.

[53] Riccardo Taormina, Stefano Galelli, Nils Ole Tippenhauer, Elad Salomons, Avi Ostfeld, Demetrios G Eliades, Mohsen Aghashahi, Raanju Sundararajan, Mohsen Pourahmadi, M Katherine Banks, B M Brentan, M Herrera, Amin Rasekh, Enrique Campbell, I Montalvo, G Lima, J Izquierdo, Kelsey Haddad, Nikolaos Gatsis, Ahmad Taha, Saravanakumar Lakshmanan Somasundaram, D Ayala-Cabrera, Sarin E Chandy, Bruce Campbell, Pratim Biswas, Cynthia S Lo, D Manzi, E Luvizotto Jr, Zachary A Barker, Marcio Giacomoni, M Fayzul K Pasha, M Ehsan Shafiee, Ahmed A Abokifa, Mashor Housh, Bijay Kc, and Ziv Ohar. 2018. The Battle Of The Attack Detection Algorithms: Disclosing Cyber Attacks On Water Distribution Networks. *Journal of Water Resources Planning and Management* 144, 8 (aug 2018), 4018048. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000969

[54] Yi Tay, Shuohang Wang, Luu Anh Tuan, Jie Fu, Minh C Phan, Xingdi Yuan, Jinfeng Rao, Siu Cheung Hui, and Aston Zhang. 2019. Simple and effective curriculum pointer-generator networks for reading comprehension over long narratives. *arXiv preprint arXiv:1905.10847* (2019).

[55] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney, and Chris Dyer. 2016. Learning the curriculum with bayesian optimization for task-specific word representation learning. *arXiv preprint arXiv:1605.03852* (2016).

[56] Jonathan G Tullis and Aaron S Benjamin. 2011. On the effectiveness of self-paced learning. *Journal of memory and language* 64, 2 (feb 2011), 109–118. https://doi.org/10.1016/j.jml.2010.11.002

[57] Heng Wang and Zubin Abraham. 2015. Concept drift detection for streaming data. *Proceedings of the International Joint Conference on Neural Networks* 2015-Septe (2015). https://doi.org/10.1109/IJCNN.2015.7280398 arXiv:1504.01044

[58] Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[59] Yunchao Wei, Xiaodan Liang, Yunpeng Chen, Xiaohui Shen, Ming Ming Cheng, Jiashi Feng, Yao Zhao, and Shuicheng Yan. 2017. STC: A Simple to Complex Framework for Weakly-Supervised Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 11 (2017), 2314–2320. https://doi.org/10.1109/TPAMI.2016.2636150 arXiv:1509.03150

[60] Daphna Weinshall, Gad Cohen, and Dan Amir. 2018. Curriculum learning by transfer learning: Theory and experiments with deep networks. *35th International Conference on Machine Learning, ICML 2018* 12 (2018), 8331–8339. arXiv:1802.03796

[61] Zhenyu Wu, Yang Guo, Wenfang Lin, Shuyang Yu, and Yang Ji. 2018. A weighted deep representation learning model for imbalanced fault diagnosis in cyber-physical systems. *Sensors* 18, 4 (2018), 1096.

[62] Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum Learning for Natural Language Understanding. (2020), 6095–6104. https://doi.org/10.18653/v1/2020.acl-main.542

[63] Qinghua Xu, Shaukat Ali, and Tao Yue. 2021. Digital Twin-based Anomaly Detection in Cyber-physical Systems. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 205–216.

[64] Qinghua Xu, Shaukat Ali, Tao Yue, and Maite Arratibel. 2022. Uncertainty-Aware Transfer Learning to Evolve Digital Twins for Industrial Elevators. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Singapore, Singapore) *(ESEC/FSE 2022)*. Association for Computing Machinery, New York, NY, USA, 1257–1268.

https://doi.org/10.1145/3540250.3558957

[65] Danfeng Yao, Xiaokui Shu, Long Cheng, and Salvatore J Stolfo. 2017. Anomaly detection as a service: challenges, advances, and opportunities. *Synthesis Lectures on Information Security, Privacy, and Trust* 9, 3 (2017), 1–173.

[66] Tao Yue, Shaukat Ali, Paolo Arcaini, and Fuyuki Ishikawa. 2022. Towards requirements engineering for digital twins of cyber-physical systems. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 9–21.

[67] Tao Yue, Paolo Arcaini, and Shaukat Ali. 2021. Understanding digital twins for cyber-physical systems: a conceptual model. In *International Symposium on Leveraging Applications of Formal Methods*. Springer, 54–71.

[68] Man Zhang, Shaukat Ali, and Tao Yue. 2019. Uncertainty-wise test case generation and minimization for cyber-physical systems. *Journal of Systems and Software* 153 (2019), 1–21.

[69] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. 2019. Uncertainty-wise cyber-physical system test modeling. *Software & Systems Modeling* 18, 2 (2019), 1379–1418.

[70] Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. *arXiv preprint arXiv:1905.05816* (2019).

[71] Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *International Conference on Machine Learning*. PMLR, 4006–4015.