Perspectives from a Comprehensive Evaluation of Reconstruction-based Anomaly Detection in Industrial Control Systems

Clement Fung*, Shreya Srinarasi, Keane Lucas,
Hay Bryan Phee, and Lujo Bauer

Carnegie Mellon University, Pittsburgh, USA
{clementf,ssrinara,kjlucas,hphee,lbauer}@andrew.cmu.edu

Abstract. Industrial control systems (ICS) provide critical functions to society and are enticing attack targets. Machine learning (ML) models—in particular, reconstruction-based ML models—are commonly used to identify attacks during ICS operation. However, the variety of ML model architectures, datasets, metrics, and techniques used in prior work makes broad comparisons and identifying optimal solutions difficult. To assist ICS security practitioners in choosing and configuring the most effective reconstruction-based anomaly detector for their ICS environment, this paper: (1) comprehensively evaluates previously proposed reconstruction-based ICS anomaly-detection approaches, and (2) shows that commonly used metrics for evaluating ML algorithms, like the point-F1 score, are inadequate for evaluating anomaly detection systems for practical use. Among our findings is that the performance of anomaly-detection systems is not closely tied to the choice of ML model architecture or hyperparameters, and that the models proposed in prior work are often larger than necessary. We also show that evaluating ICS anomaly detection over temporal ranges, e.g., with the range-F1 metric, better describes ICS anomaly-detection performance than the commonly used point-F1 metric. These so-called range-based metrics measure objectives more specific to ICS environments, such as reducing false alarms or reducing detection latency. We further show that using range-based metrics to evaluate candidate anomaly detectors leads to different conclusions about what anomaly-detection strategies are optimal.

1 Introduction

Industrial control systems (ICS) govern vital infrastructures such as power grids, water treatment plants, and transportation networks. These systems collect and monitor real-time information from an industrial process and use a programmed model to govern its operation [11]. As ICS become further interconnected, particularly with the public Internet, the attack risk increases. An adversary could either directly or over the network interfere with an ICS (e.g., by injecting false

* Corresponding author.

data or commands [36]) and cause the ICS to modify the physical process, potentially causing damage or risking human life [31]. Given the potential harms of attacks on ICS, detecting and preventing them in a timely manner is critical.

In response, researchers have proposed using machine learning (ML) to detect attacks; among ML-based techniques, reconstruction-based ML models have been shown to be particularly promising [18,32,37]. Reconstruction-based ML involves training an ML model to represent the expected, benign behavior of an ICS. After a model is trained, it is used to assess the observed real-time behavior of an ICS; any behaviors that are not consistent with the trained model are called out to operators as potentially dangerous anomalies.

To use these anomaly-detection techniques, ICS security practitioners must: (1) select the ML model architecture (e.g., convolutional neural networks), (2) select hyperparameters for the model (e.g., the size and number of hidden layers in the model), (3) collect a sufficient volume of benign ICS operational data, (4) train an ML model to reconstruct system states, and (5) tune detection hyperparameters (the threshold for an anomaly to be declared) to turn system-state reconstructions into attack predictions in a live setting. Despite the variety of work in reconstruction-based ICS anomaly detection, there is no consensus on what solutions are best. Proposed approaches use different ML model architectures (e.g., autoencoders [7,32], CNNs [18,19], LSTMs [8,37]), use different datasets [4,10,33], and employ different data pre-processing and training techniques. Thus, when one approach is reported to outperform another, it is not clear what characteristics are responsible for the improved performance.

In this work, we perform a comprehensive, empirical evaluation of techniques across the most common datasets used in reconstruction-based ICS anomaly detection. We find that most ML model architectures, regardless of the choice of model hyperparameters, perform about equally well. Additionally and to our surprise, we find that many proposed models are larger (i.e., contain more parameters) than necessary and that far smaller models provide similar detection performance. Furthermore, we identify training and data pre-processing techniques that strongly affect the results of reconstruction-based ICS anomaly detection, but are not used consistently across prior work.

Another important consideration when designing ICS anomaly-detection systems is the metric used to tune and evaluate detection strategies. Typically, prior work equally penalizes false alarms and missed attacks on a per-timestep basis by evaluating with the point-F1 score [30]. However, as we describe in Sec. 5.1, since ICS attacks take place over a sequence of timesteps [4,10], and because timely detection of attacks is important [14], ICS anomaly detection is better evaluated over temporal ranges, rather than on each timestep independently. Unlike the point-F1, which scores on individual timesteps, *range-based* metrics score detection performance on temporal ranges and can express tradeoffs between increased detection rates, reduced false-alarm rates, and lowered detection latency [34].

In this paper, we demonstrate the impact of using range-based metrics for ICS anomaly detection, building on research from other anomaly-detection do-

mains [14,21,34]. We show empirically that using these metrics to tune and evaluate ICS anomaly-detection models gives a better understanding of what models are optimal. Furthermore, we propose the use of specific ICS objectives that describe anomaly-detection performance in terms relevant to ICS operations. Given the wide variety of potential ICS anomaly-detection environments, we opt for general objectives: examples include a low false-alarm rate, a high attack-detection rate, and low-latency attack detection.

In summary, our work answers two research questions. **RQ1**: across proposed techniques for reconstruction-based ICS anomaly detection, what model architectures, model hyperparameters, and pre-processing techniques are optimal? **RQ2**: can using range-based metrics lead to a different understanding of what models are most effective for reconstruction-based ICS anomaly detection? In answering these questions, we make the following contributions:

- We perform a comprehensive comparison across the ML model architectures and datasets used in reconstruction-based ICS anomaly detection and find that the choice of model hyperparameters has little effect on detection performance; prior work often proposes model hyperparameters that are far larger than necessary.
- We implement and make publicly available[1] a comprehensive test framework that allows tuning of models and comparing the impact of factors such as datasets, metrics, and hyperparameters. We instantiate the framework with recently proposed reconstruction-based ML model architectures.
- We use range-based metrics to tune and evaluate reconstruction-based ICS anomaly detectors. We provide examples of range-based metrics that support various ICS objectives and demonstrate that models tuned with these range-based metrics outperform their point-F1-tuned counterparts on the desired ICS objectives.
- We find that using range-based metrics for optimizing anomaly-detection systems provides a different understanding of what models are best compared to using the point-F1 metric.

2 Background and Related Work

In this section, we provide background on ICS and attacks and defenses for them (Sec. 2.1). We also introduce the various models (Sec. 2.2), metrics (Sec. 2.3), and datasets (Sec. 2.4) used in prior work. We lastly categorize prior work in reconstruction-based ICS anomaly detection along these dimensions (Sec. 2.5).

2.1 Industrial Control Systems: Threats and Defenses

An ICS governs the operation of a physical, safety-critical process. Fig. 1 shows the structure and components of an ICS, and how they are separated in the hierarchical Purdue model of ICS [16]. The model divides an ICS into levels from

[1] `https://github.com/pwwl/ics-anomaly-detection`

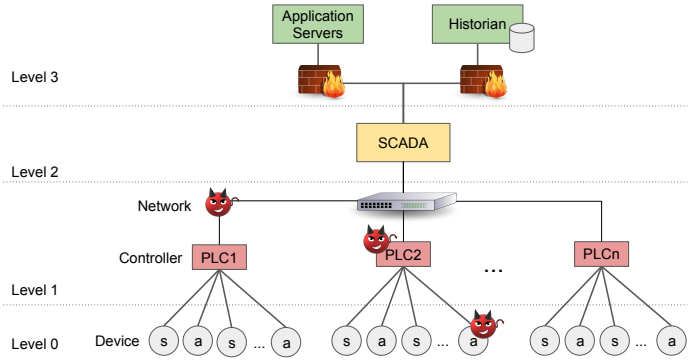


Fig. 1. An overview of a typical ICS/SCADA hierarchical layered architecture with examples of compromised endpoints and communication channels.

the physical process (Level 0, the strictest level of access) to higher-level applications (Level 3, less strict). Sensors and actuators (Level 0) allow feedback and input with the physical process. Programmable logic controllers (PLCs, Level 1) directly interface with sensors and actuators to automate the ICS process. Supervisory control and data acquisition (SCADA, Level 2) governs multiple PLCs by collecting data and providing an interface to operators for control and analysis of the physical process [31].

ICS networks were previously not monitored for security purposes. Instead, ICS were typically isolated from external threats by a firewall between Levels 2 and 3, preventing compromised, higher-level devices from manipulating the physical process [31]. With the exposure of ICS environments to the Internet and third parties [11], the potential of compromise has increased significantly: when an attacker compromises parts of an ICS in Levels 0 through 2, they can manipulate the data being sent over the network to cause process degradation or even failure. This strategy was used in the BlackEnergy (2015) and Industroyer (2016) attacks on the Ukrainian power grid [20], which caused over 200,000 people to lose electric power for several hours; and in the Triton malware attack (2017) [6], which caused a chemical processing plant to shut down. For this reason, it is critical to monitor ICS networks for signs of potential compromise and misuse.

In this work, we focus on techniques that train deep-learning ML models to perform system reconstructions and identify as potentially anomalous any system states for which the reconstruction error is high [18,32,37]. We focus on deep-learning models, as they have been shown to outperform other classical methods in anomaly detection [25]. For these techniques, three ML model architectures have risen to prominence: autoencoders (AEs), convolutional neural networks (CNNs), and long-short-term-memory (LSTM) networks. We overview these model architectures in Sec. 2.2.

Table 1. The ICS datasets most commonly used for training and evaluating reconstruction-based anomaly-detection models.

Name	# of Points in Benign Dataset	# of Points in Attack Dataset	# of Features	# of Attacks
BATADAL	48,106	10,081 (16% attack)	43	7
WADI	1,048,571	172,801 (6% attack)	103	15
SWaT	496,800	449,919 (12% attack)	51	36

2.2 ML Model Architectures for ICS Anomaly Detection

In this section, we provide an overview of ML model architectures commonly used in ICS anomaly-detection systems. Autoencoders (AEs) are composed of a sequence of stacked, fully connected layers that compress inputs into a smaller latent representation [12]; an AE is trained to reconstruct an input system state. Convolutional neural networks (CNNs) [5] and long-short-term-memory units (LSTMs) [13] instead use time-based information to predict system states: based on a fixed-time-length input, CNNs and LSTMs predict the next expected system state. CNNs use 1-D convolutional kernels to process time, whereas LSTMs use a custom unit that maintains separate weighted connections that pass information far along the time axis (long-term memory) and to immediate recent states (short-term memory).

2.3 Traditional Anomaly Detection Metrics

Anomalies are rare and accuracy scores may misrepresent the anomaly detection performance. Much of prior work uses the point-F1 score—the harmonic mean of the precision and recall—to characterize anomaly-detection performance:

$$\textit{point-F1} = \frac{2 * \textit{prec} * \textit{rec}}{\textit{prec} + \textit{rec}} \qquad \textit{prec} = \frac{TP}{TP + FP} \qquad \textit{rec} = \frac{TP}{TP + FN}$$

TP (true positives) is the number of timesteps during which an attack was correctly detected, FP (false positives) is the number of timesteps where an attack was falsely reported, and FN (false negatives) is the number of timesteps where an attack is undetected. In Sec. 5, we show the shortcomings of using the point-F1 score to tune and evaluate anomaly detectors and instead propose the use of range-based metrics.

2.4 Publicly Available ICS Datasets

In lieu of direct access to a real ICS, a variety of ICS datasets have been made publicly available for research. Each dataset is typically partitioned into two parts: a *benign dataset* and *attack dataset*. The benign dataset contains a sequence of system states during a benign execution of ICS operations. The attack dataset contains a sequence of system states during an execution that models

an attacker who gains access to the ICS and manipulates a subset of sensor and actuator values in a false-data-injection attack [10]. These datasets cover a variety of domains, including water distribution [4,33], water treatment [10], gas pipelines [24], and power generation [3,28].

In our analysis, we focus on the most commonly used datasets: BATADAL [33] (water distribution), SWaT [10] (water treatment), and WADI [4] (water distribution). Table 1 shows the details of each dataset. Since the originally released SWaT (2015) and WADI (2017) datasets, additional data from the same system has been released. However, we opt to use the original versions of both datasets to match what is used in the majority of prior work.

2.5 Prior Work in ICS Anomaly Detection

In this section, we overview the prior work in ICS anomaly detection, across the most commonly used ICS datasets identified in Sec. 2.4. Table 2 shows, for each prior work, the details of the ML model architecture, suggested optimal model hyperparameters, and metrics used for tuning and evaluation.

We identify two gaps across the state of the art. First, although some prior work compares ML model architectures [1,19,37], none covers the full selection of model architectures, datasets, and pre-processing techniques, making it is unclear what approaches are optimal across all settings.

Second, models are commonly tuned with the point-F1 (or not tuned at all), which ignores the temporal aspect of time-series detection, and does not balance the trade-offs between precision, recall, and latency in anomaly detection. Across this prior work, only one tunes with a range-based metric [18]; although some prior work considers ranges in evaluation, most only remark on the number of attacks detected or missed and only four evaluate with a range-based metric [8,18,23,27]. In Sec. 5, we show that tuning with range-based metrics results in different selections of optimal hyperparameters and different conclusions about which models perform better than others.

3 Reconstruction-based ICS Anomaly Detection Process

An anomaly detector reconstructs ICS system states to determine if an anomaly is occurring. Fig. 2a outlines this process.[2] First, system states $\vec{X}$ over the previous h timesteps are collected from observed network traffic, up to the current timestep t. Second, the trained ML model is provided the system state sequence $(\vec{X}_{t-h}, \vec{X}_{t-h+1}, \ldots, \vec{X}_t)$ and predicts the next system state $\vec{X'}_{t+1}$. Third, the predicted and observed states are compared, and the reconstruction error $\vec{e_t}$ is computed through the mean-squared-error (MSE): $\vec{e_t} = ||\vec{X'_t} - \vec{X_t}||^2$. Lastly, the prediction y'_t is calculated over a sequence of reconstruction errors $(\vec{e_0}, \vec{e_1}, \ldots, \vec{e_t})$:

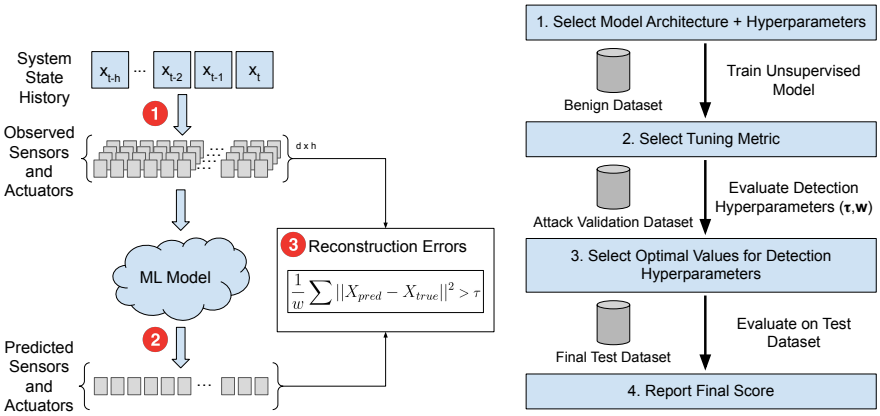
[2] Autoencoders are a special case since they do not consider a sequence of states ($h = 0$), and instead reconstruct the current state $\vec{X'_t}$.

Table 2. ML model architectures, datasets, and metrics from prior ICS anomaly-detection work. Range-based metrics are shown in **bold**. (CM = confusion matrix; TPR/FPR = true/false positive rate; TNR = true negative rate; Coverage % = percentage of detection overlap; Norm-TPR = normalized true positive rate.)

Model Details	Datasets B	S	W	Tuning Metric	Evaluation Metric(s)	Source
AE: 3-layers	●	●	●	FPR	Precision, Recall Point-F1	[19]
AE: 4-layers		●		None	Precision, Recall Point-F1, **Numenta**	[27]
AE: 5-layers	●			Point-F1	Precision, Recall Point-F1	[32]
AE: 5-layers	●		●	None	Precision, Recall Accuracy, Point-F1	[7]
CNN: 8-layers, 32 filters		●		**Range-F1**	**Range-F1**	[18]
CNN: 8-layers, 32 filters	●	●	●	FPR	Precision, Recall Point-F1	[19]
LSTM: 2-layers, 256 units		●	●	None	TPR, **Norm-TPR** FPR, **Atk TP**	[8]
LSTM: 3-layers, 100 units		●		Point-F1	Precision, Recall Point-F1	[15]
LSTM: 3-layers, 100 units		●		None	**Atk TP, Atk FP**	[9]
LSTM: 4-layers, 64 units		●		None	**Atk TP, Atk FP**	[17]
LSTM: 4-layers, 512 units		●		None	CM, Point-F1, **Atk TP**	[26]
LSTM: 4-layers, 512 units		●		Point-F1	Point-F1	[37]
1-class SVM		●		Point-F1	Point-F1	[15]
DNN: 3-layer	●			None	CM, TPR, TNR	[2]
Custom wide and deep CNN		●	●	None	Precision, Recall Point-F1, **Atk TP**	[1]
GAN		●	●	Point-F1	Precision, Recall Point-F1	[22]
Bayesian Network		●		None	**Atk FP, Atk TP FP length, Coverage %**	[23]

$y'_t = 1$ when the reconstruction error exceeds a threshold τ for w consecutive timesteps: $y'_t = \prod_{i=t}^{t+w} \mathbb{I}(\vec{e_i} > \tau)$. The threshold τ is determined using the distribution of benign-validation errors. For example, τ can be set to the distribution's 99.5-th percentile value. Both τ and the window length w are *detection hyperparameters*: they are independent of the underlying trained ML model and convert the system state reconstruction to attack predictions. We show that detection hyperparameter tuning is closely affected by the choice of metric, and optimal models often change when different metrics are used.

End-to-end, to optimize reconstruction-based anomaly detection, (1) we train a ML model to minimize MSE and (2) we tune its detection hyperparameters to maximize its performance according to a chosen metric. Fig. 2b shows the



(a) Anomaly detection process (b) Model optimization pipeline

Fig. 2. The anomaly-detection process is shown on the left: a sequence of system states is reconstructed by an ML model, and high reconstruction errors are used to identify anomalies. The optimization pipeline for the anomaly-detection process is shown on the right, with each optimization step and its relevant datasets.

steps and datasets used in optimization. Most prior work focuses on selecting the best model architecture and best model hyperparameters (step 1), but in this work we show that optimization across *both* steps plays a substantial role in the effectiveness of reconstruction-based ICS anomaly detection.

We independently evaluate both steps. In Sec. 4, we keep the choice of tuning metric (point-F1) constant and compare the performance across various ML model architectures and hyperparameters from prior work, In Sec. 5, we keep the underlying trained model constant and compare how the choice of tuning metric affects detection hyperparameter tuning. Lastly, in Sec. 5.4, we show how the choice of tuning metric affects both the optimal model hyperparameters and detection hyperparameters in an end-to-end optimization.

4 Comparing ML Model Architectures and Datasets for ICS Anomaly Detection

In this section, we report on a comprehensive comparison of model architectures and model hyperparameter values, evaluating across techniques proposed in prior work. For each model hyperparameter setting, we optimize the anomaly-detection system through the steps shown in Fig. 2b. We explain our experimental setup in Sec. 4.1 and present our findings in Sec. 4.2.

4.1 Experiment Setup

Data Pre-processing. Before training and evaluating each model, each feature is normalized; the scaling transformation is saved and applied to the attack dataset before evaluating the model. 70% of the training dataset is randomly chosen for training the ML model. The other 30%, referred to as the *benign validation dataset*, is used to give an unbiased score during training; we use the benign-validation loss as an indicator for early stopping to prevent overfitting.

In our experiments, we identify techniques that impact the quality and reproducibility of results but were used inconsistently in prior work. Techniques such as: data pre-processing through feature selection, benign data shuffling, attack cleaning, and early stopping are necessary when comparing across solutions, as they improve the quality and consistency of anomaly-detection results. We present descriptions of these techniques and their impacts in Appendix A.

Model Hyperparameter Tuning. We perform a hyperparameter search for 3 ML model architectures: autoencoders, CNNs, and LSTMs. For autoencoders, we vary the number of hidden layers in the encoder/decoder from 1 to 5 (by 1) and the compression factor from 1.5 to 4.0 (by 0.5). For CNNs, we vary the number of layers from 1 to 5 (by 1), and vary the number of units per layer from 4 to 256 (by a factor of 2). The kernel size is fixed at 3 and we use history lengths of 50, 100, or 200 timesteps. For LSTMs, we vary the number of layers from 1 to 4 (by 1), the number of units per layer from 4 to 128 (by a factor of 2), and use history lengths of 50 or 100 timesteps. Each model was implemented in Tensorflow 1.14.0 using the `tf.keras` API and trained with the Adam optimizer using its default parameters: $\{lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999\}$. A batch size of 512 samples was used during training; each model was trained for up to 100 epochs. We apply early stopping while training through the `tf.keras.callbacks.EarlyStopping` callback class, with `patience=3` (which terminates training if validation loss does not improve over 3 consecutive epochs). Across our trained models, we found that early stopping was always applied within the first 20 epochs: a finding that is consistent with prior work [15].

Detection Hyperparameter Tuning. After the model is trained, we determine the optimal detection hyperparameters using 30% of the attack dataset, referred to as the *attack validation* dataset. To simulate a setting with unseen attacks, when dividing the attack dataset into validation and testing portions, we divide the dataset into two continuous sequences.[3] To find optimal detection hyperparameter values, we perform a parameter search, based on a chosen *tuning metric*, over the following ranges: τ-percentile $\in [0.95, 0.99995]$, $w \in [1, 100]$. We report the final performance on the remaining 70% of the attack dataset for a chosen *evaluation metric*. We use the point-F1 score as both the tuning metric and evaluation metric, which Table 2 shows is commonly used in prior work.

[3] We use the first 30% of the SWaT and WADI test datasets as their corresponding attack validation datasets. We use the final 30% of the BATADAL test dataset as its corresponding attack validation dataset, since the first 30% of the BATADAL test dataset does not contain any attacks.

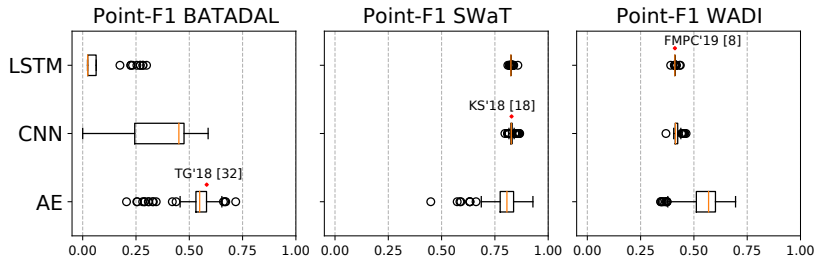


Fig. 3. The final point-F1 scores of each model when trained and tuned on three experimental ICS datasets. For each dataset, a model hyperparameter setting from prior work is included for comparison. When using the point-F1, the performance of AEs vary greatly, and most LSTM and CNN configurations perform similary.

4.2 Optimization Results

Fig. 3 shows the final point-F1 scores for each model hyperparameter setting, for each ML model architecture and dataset. We perform a full optimization three times over different random seeds for CNNs and LSTMs. For autoencoders, we observed a higher variance in the resulting point-F1 scores and thus repeat this process five times. Furthermore, we train three selected models from prior work with the same methodology. We include a 5-layer autoencoder [32], an 8-layer, 32-unit CNN with a history of 200 [18], and a 2-layer, 256-unit LSTM with a history of 50 [8]. Fig. 3 includes the point-F1 scores for these three models.

We find that larger models (CNNs and LSTMs) performed poorly on the BATADAL dataset. We attribute the poor performance to the relatively small size of the BATADAL dataset (only ∼48,000 datapoints, compared to ∼500,000 in SWaT and ∼1,000,000 in WADI); in prior work, only one study trains a CNN or LSTM on BATADAL [19]. In Sec. 5.4, we find that using a range-based evaluation metric shows CNNs and LSTMs for BATADAL in a different light, providing another example where the point-F1 may be misleading. For the SWaT and WADI datasets, we find that almost all model hyperparameter settings provide similarly strong performance: a 1-layer, 4-unit CNN or LSTM produces a similar point-F1 score to CNNs and LSTMs with more layers and units, including the optimal models from prior work [8,18,32].

Finding 1a: Substantially smaller models can achieve similar point-F1 scores as the suggested model sizes from prior work.

Prior work noted that the performance of trained models differed between runs [18], even under the same model hyperparameter settings. We found that when early stopping and benign data shuffling are used, the results for CNNs and LSTMs are more consistent: across random seeds, the final point-F1 scores always differ by less than 0.05 (and less than 0.01 for a vast majority of cases). More experimental results on the benefits of early stopping and dataset shuffling

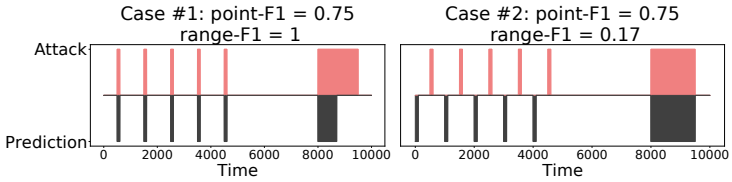


Fig. 4. Two detection examples: in each case, the x-axis represents time and the y-axis shows attacks (top, red) and attack predictions (bottom, grey). In the example on the left (case 1), all attacks are detected with no false positives, while in the example on the right (case 2) only one attack is detected, with five false positives; yet, the point-F1 scores are the same.

are provided in Appendix A. There is a higher variance across autoencoder hyperparameters, with some models achieving far higher scores than others. This is likely because the autoencoder is trained to reconstruct independent timesteps and does not consider temporal effects, rendering the performance of autoencoders unstable.

In conclusion, although prior work performs model hyperparameter searches and claims to find the optimal models for ICS anomaly detection, our experiments show that equivalent results can be achieved over a range of ML model architectures and hyperparameters when using the point-F1 score. In Sec. 5.3, we show that tuning models with range-based metrics can produce outcomes that more meaningfully address ICS anomaly-detection objectives.

Finding 1b: Although prior work focuses on optimizing the choice of ML model architecture and hyperparameters, equivalent performance can be achieved by several ML model architectures and over a wide range of model hyperparameters.

5 Tuning and Evaluating with Range-based Metrics

In this section, we first describe, in Sec. 5.1, the shortcomings of point-F1, which is commonly used by prior work in ICS anomaly detection. We introduce range-based metrics in Sec. 5.2. In Sec. 5.3, we show how range-based metrics affect detection hyperparameter tuning and in Sec. 5.4 we show how they affect what ML model architectures and hyperparameters are optimal.

5.1 Issues with the Point-F1 Score

ICS detection performance is poorly captured by the point-F1 for several reasons. (1) The point-F1 score weighs false positives and false negatives equally, whereas the cost of each may not be equal for a given ICS. (2) The point-F1 score places more importance on longer attacks [14]. A high point-F1 score can be achieved even if several short attacks are undetected; these attacks may be equally or even more harmful than attacks with a longer duration. (3) When an attack occurs

over a long period of time, it may not be important to detect *every* timestep as anomalous; once a prediction is made, corrective actions will be taken, and the existence of *any* correct prediction within the attack may be sufficient. (4) The point-F1 score does not consider *when* in the attack the detection occurs [21]. In reality, if an attack is only detected as it ends, harm may already have been caused to the ICS, rendering the detection unhelpful.

We illustrate some of these deficiencies of point-F1 using two examples of detection performance in Fig. 4. The true attack sequence is shown in red: six attacks of varying length are executed in sequence. In case 1 (left), the first five attacks are all detected perfectly, and approximately half of the last attack is detected. In case 2 (right), the first five attacks are completely missed, 5 false alarms occur, and the last attack is detected perfectly. When using the point-F1 score, the two examples misleadingly result in equal detection success: the point-F1 for both is 0.75. For many practical applications, however, case 1 shows a detection system that works well, and case 2 a detection system that works poorly. To address the shortcomings of point-F1, prior work proposes metrics better suited to time-series detection tasks [14,21,34]. We define these metrics in Sec. 5.2 and evaluate their implications in Sec. 5.3–5.4.

Observation 2a: The point-F1 score gives a misleading sense of performance for many time-series-based detection tasks.

5.2 Range-based Performance Metrics

In this section, we provide examples of range-based metrics that could be used for tuning and evaluating anomaly-detection performance. In Sec. 5.3–5.4, we show the effect of these metrics on our understanding of what models are best. We describe two types of range-based metrics: (1) range-Fβ metrics, which we define based on a prior framework for range-based metrics [34] and (2) the Numenta anomaly score [21], a metric from prior work.

Defining the Range-based Setting. Given sequences of binary labels ($y_t \in \{0, 1\}$) and predicted labels ($y'_t \in \{0, 1\}$), we convert these sequences to ranges. Let $(y_0, y_1, \ldots, y_t)$ be represented as $R = \{R_0, R_1, \ldots, R_k\}$, where each range R_i represents a continuous sequence of positive ($y_t = 1$) labels. We express the predictions $(y'_0, y'_1, \ldots, y'_t)$ in the same way to produce $R' = \{R'_0, R'_1, \ldots, R'_m\}$. If no predictions or anomalies exist ($\forall t : y_t = 0$), then $R = \emptyset$.

Range-F1 and Range-Fβ Scores. Prior work has defined a general range-based metric framework that combines existence rewards (whether any intersection exists) and overlap rewards (the size of the intersection) when scoring a time-series prediction [34]. When demonstrating the impact of range-based metrics on the understanding of ICS anomaly detection, we assume that any alarm raised by the anomaly-detection system leads to investigation, so we only consider existence rewards and leave exploring overlap rewards to future work. For our existence reward, we count any overlap between a true attack R_i and the

entire predicted range R' as a true detection. Using this notion, the range-based recall and precision are calculated as follows:

$$IsTP(R_i) = \mathbb{I}[|R_i \cap R'| \geq 1] \qquad R\text{-}rec = \frac{\sum_i IsTP(R_i)}{|R|}$$

$$IsFP(R'_i) = \mathbb{I}[|R \cap R'_i| == 0] \qquad R\text{-}prec = \frac{\sum_i IsTP(R_i)}{\sum_i^k IsTP(R_i) + \sum_i^m IsFP(R'_i)}$$

The Fβ score is a generalized version of the F1 score that scores precision with a relative weight of β. $\beta > 1$ indicates that precision is more important, whereas $\beta < 1$ indicates that recall is more important. We define the range-F1 and range-Fβ score in the same fashion as the point-F1:

$$R\text{-}F1 = \frac{2 * R\text{-}prec * R\text{-}rec}{R\text{-}prec + R\text{-}rec} \qquad R\text{-}F\beta = \frac{(1+\beta^2) * R\text{-}prec * R\text{-}rec}{(\beta^2 * R\text{-}prec) + R\text{-}rec}$$

Numenta Anomaly Score [21]. When using the Numenta anomaly score, each attack is represented by an inverted sigmoid function, plotted with its origin at the earliest true prediction. This (1) benefits earlier predictions within an anomaly and (2) assigns a small positive score to when detection is made shortly after the anomaly ends. In the original proposed Numenta score, both the position and width of the sigmoid were fixed; we use recommendations from follow-up work [29] for tuning. The Numenta score is adjusted by the position of the sigmoid function: an earlier placement in the anomaly assigns a lower score to late detection and penalizes false positives that occur shortly after the anomaly ends. κ controls the width of the sigmoid function: lower values of κ cause the function to be flatter, making the scoring more lenient towards late detection and false positives.

Parameterizing Range-based Metrics for ICS Objectives. Each range-based metric requires parameterization to contextualize their scoring. We describe the default setting for each range-based metric and provide three additional example settings for them, each prioritizing a different ICS objective.

By default, the range-F1 score as defined in Sec. 5.2 places equal importance on reducing false positives and reducing false negatives. If an example use case requires a high detection rate, we optimize for a higher recall by using the Fβ score with $\beta = 1/3$ (range-F$\beta_{1:3}$), such that recall is three times more important than precision. An alternate use case for a highly critical ICS may require that no false alarms occur. For this use case, we use the Fβ score with $\beta = 3$ (range-F$\beta_{3:1}$), which weighs precision three times more heavily than recall.

The default configuration of the Numenta anomaly score sets $\kappa = 5$ and positions the sigmoid at the 50% point of each labeled anomaly [21]. We propose an additional ICS objective that requires early attack detection, as harm may be caused to the ICS even before the attack is completed. We optimize for early detection by re-positioning the Numenta sigmoid to the 25% point of an anomaly, reducing the false positive cost by 50%, and setting $\kappa = 10$, producing a stricter decision boundary. We call this metric *NA-early*. With *NA-early*, a detection in

Table 3. For each optimal model proposed by prior work, we use a different tuning metric to select the optimal detection hyperparameters and show the resulting number of false alarms, detected attacks, and TP:FP ratio. Using range-F1 always outperforms its point-F1 counterpart in TP:FP ratio.

Dataset and Arch.	**Tuning Metric**	**False Alarms**	**Detected Attacks**	TP:FP **Ratio**
BATADAL AE	Point-F1	11	4/4	0.36
	Range-F1	1	4/4	4.00
WADI LSTM	Point-F1	143	10/13	0.07
	Range-F1	63	7/13	0.11
SWaT CNN	Point-F1	32	6/18	0.19
	Range-F1	4	4/18	1.00
	range-F$\beta_{3:1}$	0	3/18	∞
	range-F$\beta_{1:3}$	47	7/18	0.15
	NA-early	89	11/18 (7 early)	0.12

the last 75% of an attack is considered to be late and is penalized as a missed attack, as we assume that the ICS has already been damaged.

5.3 Using Range-based Metrics to Tune Detection Hyperparameters

In contrast to Sec. 4, where we selected optimal *model hyperparameters*, in this section we select optimal *detection hyperparameters* for a fixed ML model. In doing so, we reveal whether using tuning metrics other than the point-F1 leads to a different selection of detection hyperparameters and to markedly different anomaly-detection performance, which may lead to a changed understanding of which models are best or whether any are adequate for a particular deployment. For each ML model architecture, we again use the optimal model hyperparameters declared in prior work: a 8-layer, 32-unit CNN trained on SWaT [18], a 5-layer, 2-compression AE trained on BATADAL [32], and a 2-layer, 256-unit LSTM trained on WADI [8].

We compare the detection outputs when using the point-F1 and the range-F1 and show the number of detected attacks, false alarms, and ratio of true positives to false positives (TP:FP ratio) in Table 3. Prior work hypothesized that a TP:FP ratio of 1 or greater was acceptable and used the TP:FP ratio as a success metric [8]. For all three optimal models from prior work, using the range-F1 selects different detection hyperparameter values than using the point-F1. For BATADAL and SWaT, using the point-F1 for detection hyperparameter tuning results in an unacceptable model (TP:FP ratio < 1), whereas using the range-F1 for detection hyperparameter tuning results in an acceptable model (TP:FP ratio ≥ 1).

Using range-based metrics in tuning can achieve outcomes beyond an improved TP:FP ratio. Table 3 shows the final detection results for our additional metrics (defined in Sec. 5.2) after tuning the SWaT CNN from prior work [18].

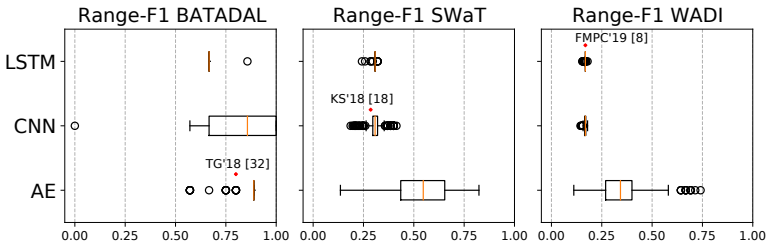


Fig. 5. The final range-F1 scores of each model when trained and tuned on 3 experimental ICS datasets. For each dataset, a selected model hyperparameter setting from prior work is included for comparison.

To detect more attacks, we tune with range-F$\beta_{1:3}$. The resulting tuning detects more attacks (7/18) than prior tunings, at the cost of more false alarms (47). Conversely, to detect attacks with absolutely no false alarms, the range-F$\beta_{3:1}$ tuning can be used; fewer attacks (3/18) are detected but no false alarms occur. Both tunings outperform their point-F1 or range-F1 counterparts on the chosen objectives.

Lastly, we use NA-early to optimize for an ICS where only early detections (within the first 25% of the attack) are useful. The original point-F1 tuning produces 32 false alarms and detects six attacks, five of which are detected early. With NA-early, the total number of false alarms (89) and attacks detected (11/18) increase, but seven attacks are detected early, which outperforms the general tuning selected by the point-F1.

Given the various ICS trade-offs and use cases, a universally optimal strategy for hyperparameter tuning cannot exist, and we do not advocate for specific metrics or hyperparameter values. Rather, we show that when tuning with range-based metrics, it is possible to produce anomaly-detection systems that better match defined ICS objectives.

Finding 2b: By using objective-driven range-based metrics to tune detection hyperparameters, the resulting anomaly detection systems can better address the defined objectives than their point-F1-tuned counterparts.

5.4 Using Range-based Metrics to Select Model Hyperparameters

In this section, we revisit model hyperparameter selection and show how range-based metrics alter the findings from Sec. 4. Compared to the point-F1, using a range-based metric for tuning and evaluation consistently leads to different conclusions about which models are optimal. Fig. 5 shows the final range-F1 scores after repeating the experiments described in Sec. 4.2: we train each ML model architecture under each model hyperparameter setting and tune the detection hyperparameters with the range-F1.

Across model hyperparameters, CNNs/LSTMs on SWaT/WADI perform similarly regardless of whether range-F1 or point-F1 is used in tuning: the difference in range-F1 (or point-F1) between model hyperparameter choices is small, and the best performance can be achieved over a wide range of model hyperparameters. The results on BATADAL are different from those computed by tuning with point-F1 (Sec. 4.2): Despite far lower point-F1 scores, over 25% of CNNs produce a range-F1 of 1, detecting all attacks without a single false alarm! Range-F1-optimal LSTMs for BATADAL yield similar results: the best models detect two out of four attacks with no false positives (perfect segment precision, 50% segment recall) and exhibit a high range-F1, but point-F1 scores below 0.2. In summary, previous experiments indicated that autoencoders were best for BATADAL but no model performed particularly well; using the range-F1 still reveals that autoencoders are on average, the best, but that all models perform quite well. When the combination of ML model architecture and dataset is held constant, the selected model hyperparameters *always* differ between the range-based metric tuning (range-F1, range-Fβ or NA-early) and the point-F1 tuning, changing our understanding of what models are optimal.

Finding 2c: When using range-based metrics to optimize reconstruction-based ICS anomaly detection, the selected ML model architectures and hyperparameters are typically different from what would be selected when using point-F1; this often changes the understanding of what model performs best by a substantial margin.

In summary, we show that using range-based metrics to tune and evaluate ICS anomaly-detection models (i) selects different outcomes compared to when using the point-F1 and (ii) better addresses objectives relevant to ICS anomaly detection. We evaluated these claims across three ICS datasets and note that these datasets may not encompass the wide range of ICS behavior. Extending our analysis to other datasets remains future work.

6 Conclusion

In this work, we explored the optimization of reconstruction-based ICS anomaly detection. We performed a comprehensive comparison across anomaly-detection solutions proposed in prior work, spanning three ICS datasets and three ML model architectures. In doing so, we found that there is no globally optimal technique and the best performance can be achieved over a range of ML model architectures and hyperparameters. We used range-based metrics to optimize ICS anomaly detection and found that they lead to different and potentially more useful outcomes than the common approach of relying on the point-F1 score. Ultimately, we found that effective anomaly detection extends beyond optimizing ML models for the point-F1, and better success measures are needed to practically tune and evaluate ICS anomaly-detection models. We hope that future work in reconstruction-based ICS anomaly detection considers its various use cases when designing new ICS anomaly-detection techniques.

Acknowledgements. We thank our shepherd and our anonymous reviewers for their insightful feedback. We also thank Camille Cobb, Trevor Kann, and Brian Singer for helpful comments on prior drafts of this paper. This material is based upon work supported by: the U.S. Army Research Office and the U.S. Army Futures Command under Contract No. W911NF-20-D-0002; DARPA GARD under Cooperative Agreement No. HR00112020006; a DoD National Defense Science and Engineering Graduate fellowship; the Secure and Private IoT initiative at Carnegie Mellon Cylab (IoT@CyLab); and Mitsubishi Heavy Industries through the Carnegie Mellon CyLab partnership program.

A Key Findings in the Optimization Process

We identified four techniques that enhance the quality and reproducibility of anomaly detection performance. Table 4 shows which previous works use these techniques; no prior work incorporates all four.

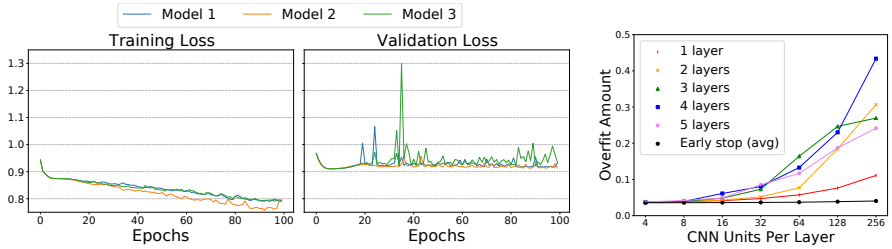
Finding 1c: Techniques such as benign data shuffling, attack cleaning, feature selection, and early stopping increase the quality and reproducibility of results, but are applied inconsistently in prior work.

Finding #1: Feature Selection. In WADI and SWaT, some benign-labeled test data appears significantly different from benign-labeled training data [19,35]. To address this problem, statistical tests are used to select features for the ML model. Prior work used a modified version of the Kolmogorov-Smirnov test (called K-S*) [19] to identify features with a significant difference between their training and test distributions. 11 features are removed from SWaT, and 10 features are removed from WADI, which matches the proportion of features removed from these datasets in prior work [19]. We found that feature selection is only effective on the SWaT dataset, so we only use feature selection for SWaT.

Finding #2: Attack Cleaning. Some attacks in the SWaT dataset do not execute as described [15,17]: although labelled as attacks, the SWaT description [10] notes that they did not actually perform as intended. These cases should not be evaluated as attacks, yet the majority of prior work does. We recommend

Table 4. Identifying key pre-processing and model training techniques from prior ICS anomaly-detection work. '●', '◐', and '○' indicate if the technique was used, partially used, or not used respectively. '**?**' indicates that we could not determine if the technique was used.

Source	[1]	[2]	[7]	[8]	[9]	[15]	[17]	[18]	[19]	[22]	[23]	[26]	[27]	[32]	[37]
Feature Selection	○	○	○	○	◐	○	○	○	●	◐	○	●	○	○	●
Attack Cleaning	○	○	○	○	◐	●	●	◐	○	○	●	●	○	○	◐
Benign Data Shuffling	**?**	●	**?**	○	**?**	○	○	**?**	**?**	○	○	○	○	**?**	**?**
Early Stopping	○	○	●	○	●	●	○	●	○	○	○	○	○	●	○



(a) Comparing losses for a 4-layer, 64-unit CNN. (b) Overfit by model size.

Fig. 6. On left (a): the training and validation loss for a 4-layer, 64-unit CNN, across random seeds. On right (b): the average overfit amount without early stopping, shown for all CNN sizes, compared to the average overfit amount for all layers with early stopping.

removing the benign "attacks" from the dataset. Furthermore, other prior work has noted that the start and end times of attacks in SWaT are incorrect [37]. Hence, we recommend that the times of the labelled attacks be corrected.[4]

Finding #3: Benign Data Shuffling. When most prior work divides the benign dataset into training and validation portions, it divides by a fixed time [8] or does not describe how the division is performed. Since system behavior can differ between days (e.g., if the final 30% of timesteps in SWaT are used for validation, the distributions of the training and validation datasets are significantly different), splitting should be *random* across the benign dataset. For CNNs and LSTMs, each timestep's history should be collected before splitting.

Finding #4: Early Stopping. When early stopping is not used, models overfit quickly and tend to diverge. We train a 4-layer, 64-unit CNN with a history length of 50, repeated three times across random seeds; the model hyperparameters, data ordering, and training parameters are all unchanged. Fig. 6a shows the training and validation losses for 100 epochs. When early stopping is not used, the models overfit (validation loss plateaus after the 6th epoch and begins to increase afterward) and diverge after 10-20 epochs; this happens across all model architectures, model hyperparameters, and datasets. Across CNN sizes, Fig. 6b compares the final training and validation loss difference (overfit amount) with and without early stopping, averaged across three random seeds. With early stopping, the overfit amount is small for all model sizes. Without early stopping, larger models overfit more.

References

1. Abdelaty, M., Doriguzzi-Corin, R., Siracusa, D.: DAICS: A deep learning solution for anomaly detection in industrial control systems. arXiv:2009.06299 (2020)

[4] The recommended SWaT corrections can be found at `https://github.com/pwwl/ics-anomaly-detection`.

2. Abokifa, A.A., Haddad, K., Lo, C.S., Biswas, P.: Detection of cyber physical attacks on water distribution systems via principal component analysis and artificial neural networks. In: World Environmental and Water Resources Congress (2017)
3. Adepu, S., Kandasamy, N.K., Mathur, A.: EPIC: An electric power testbed for research and training in cyber physical systems security. In: Workshop on the Security of Industrial Control Systems and Cyber-Physical Systems (2018)
4. Ahmed, C.M., Palleti, V.R., Mathur, A.P.: WADI: a water distribution testbed for research in the design of secure cyber physical systems. In: 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks (2017)
5. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of machine learning research **12** (2011)
6. Di Pinto, A., Dragoni, Y., Carcano, A.: TRITON: The first ICS cyber attack on safety instrument systems. In: Black Hat USA (2018)
7. Erba, A., Taormina, R., Galelli, S., Pogliani, M., Carminati, M., Zanero, S., Tippenhauer, N.O.: Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In: Annual Computer Security Applications Conference (2020)
8. Feng, C., Palleti, V.R., Mathur, A., Chana, D.: A systematic framework to generate invariants for anomaly detection in industrial control systems. In: Network and Distributed System Security Symposium (2019)
9. Goh, J., Adepu, S., Tan, M., Lee, Z.S.: Anomaly detection in cyber physical systems using recurrent neural networks. In: 18th International Symposium on High Assurance Systems Engineering (2017)
10. Goh, J., Adepu, S., Junejo, K.N., Mathur, A.: A dataset to support research in the design of secure water treatment systems. In: International Conference on Critical Information Infrastructures Security (2016)
11. Hasselquist, D., Rawat, A., Gurtov, A.: Trends and detection avoidance of internet-connected industrial control systems. IEEE Access **7** (2019)
12. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786) (2006)
13. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8) (1997)
14. Hwang, W.S., Yun, J.H., Kim, J., Kim, H.C.: Time-series aware precision and recall for anomaly detection: Considering variety of detection result and addressing ambiguous labeling. In: 28th ACM International Conference on Information and Knowledge Management (2019)
15. Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C.M., Sun, J.: Anomaly detection for a water treatment system using unsupervised machine learning. In: IEEE International Conference on Data Mining Workshops (2017)
16. Jones, A.T., McLean, C.R.: A proposed hierarchical control model for automated manufacturing systems. Journal of Manufacturing Systems **5**(1) (1986)
17. Kim, J., Yun, J.H., Kim, H.C.: Anomaly detection for industrial control systems using sequence-to-sequence neural networks. In: 5th Workshop on the Security of Industrial Control Systems and of Cyber-Physical Systems (2019)
18. Kravchik, M., Shabtai, A.: Detecting cyber attacks in industrial control systems using convolutional neural networks. In: Workshop on Cyber-Physical Systems Security and Privacy (2018)
19. Kravchik, M., Shabtai, A.: Efficient cyber attack detection in industrial control systems using lightweight neural networks and PCA. IEEE Transactions on Dependable and Secure Computing (2021)

20. Kshetri, N., Voas, J.: Hacking power grids: A current problem. Computer **50**(12) (2017)
21. Lavin, A., Ahmad, S.: Evaluating real-time anomaly detection algorithms–the Numenta anomaly benchmark. In: 14th International Conference on Machine Learning and Applications (2015)
22. Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.K.: MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In: International Conference on Artificial Neural Networks (2019)
23. Lin, Q., Adepu, S., Verwer, S., Mathur, A.: TABOR: A graphical model-based approach for anomaly detection in industrial control systems. In: Asia Conference on Computer and Communications Security (2018)
24. Morris, T.H., Thornton, Z., Turnipseed, I.: Industrial control system simulation and data logging for intrusion detection system research. 7th Annual Southeastern Cyber Security Summit (2015)
25. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM Computing Surveys (CSUR) **54**(2) (2021)
26. Perales Gómez, Á.L., Fernández Maimó, L., Huertas Celdrán, A., García Clemente, F.J.: MADICS: A methodology for anomaly detection in industrial control systems. Symmetry **12**(10) (2020)
27. Shalyga, D., Filonov, P., Lavrentyev, A.: Anomaly detection for water treatment system based on neural network with automatic architecture optimization. arXiv:1807.07282 (2018)
28. Shin, H.K., Lee, W., Yun, J.H., Kim, H.: HAI 1.0: HIL-based augmented ICS security dataset. In: 13th USENIX Workshop on Cyber Security Experimentation and Test (2020)
29. Singh, N., Olinsky, C.: Demystifying Numenta anomaly benchmark. In: International Joint Conference on Neural Networks (2017)
30. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Information Processing and Management **45**(4) (2009)
31. Stouffer, K.: Guide to industrial control systems (ICS) security. NIST special publication **800**(82) (2011)
32. Taormina, R., Galelli, S.: Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems. Journal of Water Resources Planning and Management **144**(10) (2018)
33. Taormina, R., Galelli, S., Tippenhauer, N.O., Salomons, E., Ostfeld, A., Eliades, D.G., Aghashahi, M., Sundararajan, R., Pourahmadi, M., Banks, M.K., et al.: Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. Journal of Water Resources Planning and Management **144**(8) (2018)
34. Tatbul, N., Lee, T.J., Zdonik, S., Alam, M., Gottschlich, J.: Precision and recall for time series. In: Advances in Neural Information Processing Systems (2018)
35. Turrin, F., Erba, A., Tippenhauer, N.O., Conti, M.: A statistical analysis framework for ICS process datasets. In: Joint Workshop on CPS&IoT Security and Privacy (2020)
36. Ye, D., Zhang, T.Y.: Summation detector for false data-injection attack in cyber-physical systems. IEEE Transactions on Cybernetics **50**(6) (2020)
37. Zizzo, G., Hankin, C., Maffeis, S., Jones, K.: Intrusion detection for industrial control systems: Evaluation analysis and adversarial attacks. arXiv:1911.04278 (2019)