Proposing Guidelines and Approaches to Make Anomaly Detection More Effective for Industrial Control Systems

Clement Fung

CMU-S3D-25-118

September 15, 2025

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:

Prof. Lujo Bauer, Carnegie Mellon University (Chair)
Prof. Eunsuk Kang, Carnegie Mellon University
Prof. Vyas Sekar, Carnegie Mellon University
Prof. Michael Reiter, Duke University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Societal Computing.

Copyright © 2025 Clement Fung

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 and Contract No. W911NF-20-F-0015; the U.S. Department of Defense under Contract No. FA8702-15-D-0002; DARPA GARD under Cooperative Agreement No. HR00112020006; the National Science Foundation under Award No. 1801391 and Award No. 2338301; the Defense Science and Technology Agency; the Bill and Melinda Gates Foundation; the Secure and Private IoT initiative at Carnegie Mellon Cylab (IoT@CyLab); and Mitsubishi Heavy Industries through the Carnegie Mellon CyLab partnership program.

The views and conclusions contained in this document are those of the author and do not necessarily reflect the official policies or positions, either expressed or implied, of any sponsoring institution, and no official endorsement should be inferred.

Keywords: Security, Machine Learning, Anomaly Detection, Industrial Control Systems

To my parents, Felix Fung and Ivy Fung, whose constant example of strong work ethic and self-belief has guided my life.

Abstract

Industrial control systems (ICS) govern critical infrastructure and processes, such as power generation, chemical processing, and water treatment. Given their widespread impact and their critical nature, there is a strong incentive for adversaries to attack ICS. An adversary that gains access to an ICS network can manipulate its process values to cause physical damage and harm. Anomaly-detection methods based on machine learning (ML) can detect these manipulations from real-time data and is commonly proposed for defending ICS. To make anomaly detection more effective for ICS, this thesis investigates and proposes solutions to several challenges when applying anomaly detection to an ICS. First, it is unclear what ML models and methods are best for detecting ICS anomalies; we comprehensively evaluate prior approaches and compare their performance, identifying what strategies were most effective. Second, it is unclear if and how the outputs of ML-based anomaly-detection approaches can be used to diagnose ICS anomalies; we evaluate a variety of approaches for attributing ICS anomalies to the underlying components that were manipulated. Third, many anomaly-detection approaches proposed in prior work are based on general-purpose ML models that learn spurious relationships; we propose a method that embeds ICS-specific domain knowledge into structurally sparse ML models to improve detection, attribution, and robustness. Finally, to better understand how ML-based anomaly-detection approaches could be used more effectively for ICS in practice, we conduct an interview-based study to understand the workflows and perspectives of practitioners that work with ICS, and we recommend ways for researchers to design ML-based approaches for better adoption in ICS.

Thesis Statement: To make anomaly detection more effective for industrial control systems (ICS), we design approaches for detecting and attributing ICS anomalies and propose guidelines for choosing and configuring anomaly-detection models. In particular, we demonstrate that adopting ICS-specific models and objectives improves detection and attribution, and we propose new approaches for use cases beyond real-time detection, such as post-hoc diagnosis. Our approaches and guidelines improve effectiveness across the ICS anomaly-detection workflow: (i) when detecting anomalies; (ii) when identifying the root cause of anomalies; and (iii) when deploying, using, and maintaining anomaly-detection systems.

Contents

Abstract v

1 Introduction 1

2 Background and Related Work 5
2.1 Attacks on industrial control systems (ICS) . . . 5
2.2 Datasets for ICS anomaly detection . . . 6
2.3 Traditional anomaly detection metrics . . . 7
2.4 Models for process-level ICS anomaly detection . . . 7
2.5 Attribution methods for machine-learning models . . . 8
2.6 Challenges faced by practitioners in contexts similar to ICS anomaly detection . . . 9

3 Comparing models and techniques used for detecting ICS anomalies 11
3.1 Introduction . . . 11
3.2 Describing the reconstruction-based ICS anomaly detection process . . . 12
3.3 Comparing methodologies from prior work . . . 13
3.3.1 Comparing models, hyperparameters, and metrics . . . 13
3.3.2 Comparing training and data-processing techniques . . . 15
3.4 Comparing ML model architectures and datasets for ICS anomaly detection . . . 17
3.4.1 Experiment setup . . . 17
3.4.2 Optimization results . . . 18
3.5 Tuning and evaluating with range-based metrics . . . 19
3.5.1 Issues with the point-F1 score . . . 19
3.5.2 Range-based performance metrics . . . 20
3.5.3 Using range-based metrics to tune detection hyperparameters . . . 22
3.5.4 Using range-based metrics to select model hyperparameters . . . 23
3.6 Summary . . . 24

4 Evaluating attributions for ICS anomaly detection 25
4.1 Introduction . . . 25
4.2 Methodology . . . 26
4.2.1 Datasets used for training and evaluation . . . 26
4.2.2 Implementing ICS anomaly detection . . . 28
4.2.3 Attribution methods for ICS anomaly detection . . . 29

4.2.4 Evaluation metric for attributions: AvgRank . . . 31
4.3 Results: Evaluating attributions of ICS anomalies . . . 32
4.3.1 Assessing prior attribution strategies . . . 32
4.3.2 Selecting attribution methods with a counterfactual benchmark . . . 33
4.3.3 Evaluating ML-based attribution methods . . . 35
4.4 Results: Factors that affect attribution accuracy . . . 36
4.4.1 Effect of detection timing on attributions . . . 36
4.4.2 Effect of attack properties on attributions . . . 40
4.4.3 Evaluating against stealthier manipulations . . . 42
4.4.4 Evaluating ensembles of attribution methods . . . 44
4.5 Survey: ICS operator perceptions of attributions . . . 45
4.6 Discussion and recommendations . . . 48
4.6.1 Recommendations for researchers . . . 48
4.6.2 Recommendations for practitioners . . . 49
4.7 Summary . . . 50

5 CyPRESS: a structurally sparse model for ICS anomaly detection 51
5.1 Introduction . . . 51
5.2 Model architectures for anomaly detection . . . 52
5.2.1 Data description models . . . 52
5.2.2 Models used for ICS anomaly detection . . . 53
5.3 CyPRESS: Cyber-Physical Representations with Sparse Structures . . . 55
5.3.1 Specifying inter-feature relationships . . . 55
5.3.2 Learning weights in CyPRESS . . . 56
5.4 Analyzing spurious relationships learned by ICS anomaly-detection models . . . 56
5.5 Evaluation setup . . . 58
5.5.1 Baseline models . . . 58
5.5.2 CyPRESS . . . 59
5.6 Evaluation results . . . 61
5.6.1 Anomaly detection . . . 62
5.6.2 Anomaly attribution . . . 63
5.6.3 Robustness to stealthy attack strategies . . . 66
5.7 Future work and limitations . . . 70
5.8 Summary . . . 71

6 Examining practitioner's perspectives of ML-based tools for ICS alarms 73
6.1 Introduction . . . 73
6.2 Participants and methodology . . . 74
6.2.1 Participant recruitment and demographics . . . 75
6.2.2 Interview methodology . . . 76
6.2.3 Analysis methodology . . . 76
6.2.4 Ethics . . . 77
6.2.5 Limitations . . . 78
6.3 Results: Current practices for alarms in ICS . . . 78

6.3.1 Systems for raising alarms . . . 79
6.3.2 Human tasks in alarm workflows . . . 80
6.3.3 Challenges with alarms . . . 82
6.3.4 Factors that affect alarm workflows . . . 83
6.3.5 Adopting vendor tools in ICS . . . 84
6.4 Results: Perceptions of AI . . . 85
6.4.1 Conceptual models of AI . . . 86
6.4.2 Perceived benefits of adopting AI in ICS . . . 86
6.4.3 Perceived barriers to adopting AI in ICS . . . 87
6.5 Analysis and recommendations . . . 88
6.5.1 Deploying AI in systems for alarms . . . 88
6.5.2 Using AI to support alarm workflow tasks . . . 89
6.5.3 Navigating barriers to AI adoption . . . 90
6.6 Summary . . . 91

7 Conclusion **93**

A Survey text used in Chapter 4 **95**

B Interview scripts used in Chapter 6 **99**

C Qualitative codes used in Chapter 6 **103**

Bibliography **107**

List of Figures

2.1 Overview of a layered ICS architecture. 5

3.1 An overview of the anomaly-detection process (left) and model optimization pipeline (right). 12
3.2 A comparison of training runs that demonstrates the impact of random seeds and early stopping. 16
3.3 The final point-F1 scores of each model when trained and tuned on three experimental ICS datasets. 18
3.4 Two detection examples that demonstrate the difference between point-F1 and range-F1. 20
3.5 The final range-F1 scores of each model when trained and tuned on three experimental ICS datasets. 23

4.1 An overview of the ICS anomaly attribution process. 26
4.2 The results of our counterfactual benchmark test of attribution methods. 34
4.3 Attribution results for ICS anomalies at the time of detection. 35
4.4 An example that shows different timing strategies for attributing an ICS anomaly. 36
4.5 Attribution results for ICS anomalies based on relative detection time. 37
4.6 Attribution results for ICS anomalies with different timing strategies. 39
4.7 Attribution results for ICS anomalies using a weighted average of attribution methods. 44

5.1 A comparison of the internal structures of CNNs, FCDD, FSNs, and CyPRESS. 53
5.2 An illustration of the graph specification and training processes for CyPRESS. . 54
5.3 Methodology and results of our counterfactual test for spurious relationships. . . 57
5.4 Feature-based and PLC-based attribution results for best-performing anomaly detection models. 64
5.5 Feature-based and PLC-based attribution results for anomaly detection models. . 65
5.6 An overview of an ICS and the threat models we consider. 66
5.7 Results of stealthy replay attacks and variable manipulation attacks on best-performing anomaly detection models. 67
5.8 Attack success rates for evasion attacks on best-performing anomaly detection models. 68
5.9 CDF of attack success rates for evasion attacks on best-performing anomaly detection models. 69

6.1 A mapping of participant percentages to qualitative terms used in this work. . . . 77
6.2 An overview of the different tasks performed in ICS alarm workflows. 78

List of Tables

3.1 A categorization of prior work in ICS anomaly detection, based on which model architectures, datasets, and metrics are used. . . . 14
3.2 A categorization of prior work in ICS anomaly detection, based on which pre-processing and model training techniques are used. . . . 15
3.3 The effect of tuning metric on final detection outcomes for each optimal model proposed in prior work. . . . 22

4.1 A summary of the manipulations used for evaluation in prior work. . . . 28
4.2 The number of detected attacks on each dataset for each anomaly-detection model. 29
4.3 The attribution accuracy of the baseline attribution strategy from prior work. . . . 33
4.4 The number of attacks captured in different detection-timing cases. . . . 37
4.5 The results of various statistical tests for the impact of attack factors on attribution accuracy. . . . 41
4.6 Attribution results for ICS anomalies with stealthy manipulation strategies. . . . 43
4.7 List of survey participants for work presented in Chapter 4. . . . 47
4.8 Participants' ratings for the usefulness of hypothetical attribution outputs. . . . 47

5.1 A comparison of ML model architectures used in prior work for ICS anomaly detection. . . . 52
5.2 Parameter count and inference times for different anomaly detection models. . . . 60
5.3 Detection results for CYPRESS and baseline models, based on the point-F1. . . . 61
5.4 Detection results for CYPRESS and baseline models, based on the range-F1. . . . 62

6.1 List of survey participants for work presented in Chapter 6. . . . 76
6.2 A list of the different tasks performed for ICS alarms, with opportunities for AI adoption. . . . 88

A.1 Sample output from the detector used in the survey of operators. . . . 96

C.1 Codes for responses in Part II of our interview script. . . . 104
C.2 Codes for responses in Part III and Part IV of our interview scripts. . . . 105

Chapter 1

Introduction

Industrial control systems (ICS) govern critical infrastructure, such as water treatment, power generation, and chemical processing. Because of their criticality and wide-ranging impact, ICS are common targets for attacks [117]. For example, the 2016 BlackEnergy attack on the Ukrainian power grid caused over 200,000 people to lose electric power for several hours [102], and the Colonial Pipeline attack disrupted oil production in the United States for five days [19].

One strategy for attacking ICS is to manipulated process data in real-time: an attacker that gains access to an ICS network and manipulates a subset of its process values can destabilize the ICS to cause physical damage and harm [23, 45, 49, 74, 81]. To prevent such attacks from causing attacker-intended damage, researchers have proposed anomaly-detection approaches that can be used to detect manipulated ICS process values in real time [46, 75, 132]. Researchers have proposed different types of process-level ICS anomaly detection, including approaches based on invariants [1, 6, 39, 130], program-level models [60, 63], clustering methods [12, 73], physics-based models [25, 99, 104, 110], light-weight machine-learning (ML) methods [23, 52, 80, 133, 142], and deep learning models [29, 41, 54, 69, 70, 98, 148].

In this thesis, I investigate the effectiveness of ML-based anomaly-detection approaches, both based on light-weight models and based on deep learning, which are becoming increasingly popular [75, 85, 131]. These approaches first train a model to recognize and reconstruct the process values expected during normal ICS operation, and then use these models to detect when an ICS deviates from its safe, expected operation. Sufficiently high deviations are used to indicate an anomaly. Although researchers report strong detection performance with such approaches, they have yet to achieve widespread effectiveness and adoption [8, 26, 53, 121]. To improve the effectiveness and adoption of anomaly detection for ICS, I design approaches for detecting and explaining anomalies and propose guidelines for choosing, configuring, and deploying anomaly-detection models. These approaches and guidelines are used across the anomaly-detection workflow:

- **When detecting anomalies**: evaluating which model architectures work best for detection; identifying the best techniques for training and evaluating models; and tuning, designing, and developing models based on domain-specific needs.
- **When identifying the sources of anomalies**: designing and developing approaches to attribute detected anomalies to manipulated ICS components; and analyzing when and

why certain approaches are more effective than others based on ICS attack and defense configurations.

- **When end-users and organizations interact with anomaly-detection systems**: examining how ICS are monitored in practice; and recommending guidelines to use, maintain, and deploy anomaly detection at organizations that work with ICS.

This thesis is comprised of the following chapters:

- In Chapter 2, I cover the background and related work for this thesis, composing multiple areas of research: ICS security, anomaly-detection models, attribution methods for deep-learning-based models, and studies on challenges faced by security-relevant practitioners in contexts that partially overlap with the ICS-anomaly-detection context.
- In Chapter 3, I describe work that investigates what models and techniques are most effective for detecting ICS anomalies. In this work, my collaborators and I evaluate previously proposed ICS anomaly-detection approaches with a common methodology: comparing deep-learning-based models, training and data processing techniques, and metrics used for ICS anomaly detection. In contrast to what is suggested in prior work, we find that the choice of model hyperparameters for deep-learning-based models plays a small role in determining which approaches work best. We instead find that data processing, training methods, and the choice of metrics have a much larger impact on anomaly-detection effectiveness. This work is published in the *27th European Symposium on Research in Computer Security* (ESORICS 2022) [41].
- In Chapter 4, I describe work that investigates if and how attributions can be used to help operators with anomaly diagnosis. In this work, my collaborators and I evaluate how accurate attribution methods are at identifying which features were manipulated in an attack on ICS. We find that prior, off-the-shelf attribution methods are ineffective for ICS anomaly detection; furthermore, factors such as the timing of attribution input and the type of feature that was manipulated affect how well attribution methods perform and which methods were best. We ultimately propose an approach that uses an ensemble of attribution methods, and we showed that it performs best. This work is published in the *31st Network and Distributed System Security Symposium* (NDSS 2024) [42].
- In Chapter 5, I propose and describe the evaluation of a technique that makes anomaly detection and attribution more effective for ICS by augmenting anomaly-detection models with domain-specific knowledge. In this work, my collaborators and I use spatial and logical information from an ICS to improve prior anomaly-detection approaches; we represent an ICS as a connected graph, and we use these graphs to build CYPRESS, a novel, structurally sparse anomaly detection model. We ultimately show that, by enforcing sparse representations of ICS in CYPRESS, we can train models that are competitive with the state-of-the-art in detection, outperform baselines in attribution and robustness, and requiring far fewer model parameters than deep-learning-based models. This work is in submission to the *35th USENIX Security Symposium* (USENIX Security 2026).
- Finally, in Chapter 6, I describe work that investigates opportunities to make ICS anomaly detection more effective in practice. In this work, my collaborators and I conduct semi-structured interviews with practitioners who monitor ICS and protect ICS from potentially

harmful anomalies, asking them about (i) the tools and technology used to monitor ICS and raise alarms, (ii) the human tasks that are commonly performed with alarm data, and (iii) their perspectives on machine-learning-based approaches and its potential to help with ICS anomalies. We analyze these interview responses and identified opportunities for machine-learning-based tools to help with ICS anomalies, by focusing on assisting with anomaly diagnosis and alarm management. We also make recommendations for researchers to more effectively adopt machine-learning-based approaches to protect ICS. This work is published in the *21st Symposium on Usable Privacy and Security* (SOUPS 2025) [43].

Chapter 2

Background and Related Work

In this chapter, I cover the relevant background and related work for the research areas encompassed by this thesis proposal. I first describe ICS (Section 2.1), attacks on ICS (Section 2.1), and public ICS datasets used for evaluation in research (Section 2.2). I then describe models that are used to detect ICS anomalies (Section 2.4) and methods that attribute machine-learning model predictions to their input features (Section 2.5). Finally, I describe studies of practitioners that work in contexts that partially overlap with the ICS anomaly-detection context (Section 2.6).

2.1 Attacks on industrial control systems (ICS)

ICS monitor and control physical, safety-critical processes. ICS are interconnected systems that are separated into layers of access and function in the hierarchical Purdue model of ICS [62].

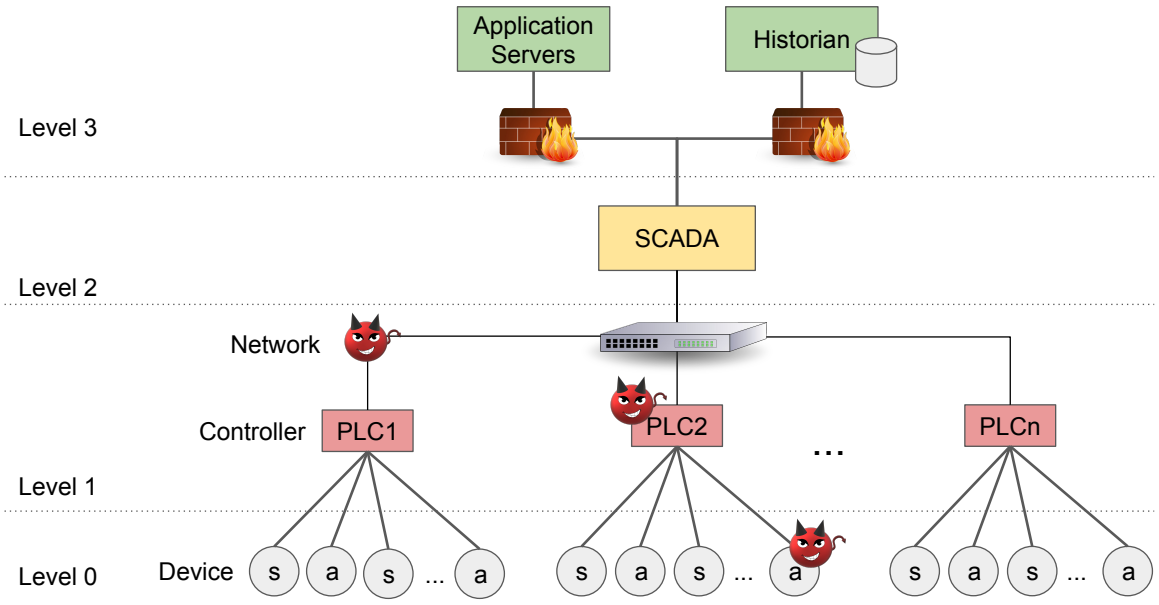


Figure 2.1: An overview of a typical, layered ICS architecture with examples of compromised endpoints and communication channels.

Figure 2.1 shows an example of the layers in an ICS. The Purdue model divides an ICS into levels: from the physical process (Level 0, the strictest level of access) to higher-level applications (Level 3, less strict). Sensors and actuators (Level 0) provide feedback from and input to the physical process. Programmable logic controllers (PLCs, Level 1) directly interface with sensors and actuators to control the ICS process. Supervisory control and data acquisition (SCADA, Level 2) governs multiple PLCs by collecting process data and providing an interface for operators to control and analyze the physical process [122]. With the exposure of ICS environments to the Internet and third parties [117], the potential of compromise has increased significantly. If an attacker compromises parts of an ICS in Levels 0–2, they can manipulate the data being sent over the network to cause process degradation or failure [81]. To prevent these potentially harmful outcomes, it is critical to monitor ICS networks for signs of potential compromise and manipulation.

In this thesis, I focus on multiple aspects of ICS anomaly detection, spanning from the effectiveness of anomaly-detection models (Chapter 3, Chapter 5), techniques that help identify which feature was manipulated (Chapter 4, Chapter 5), and how users would interact with and use anomaly-detection systems (Chapter 6).

2.2 Datasets for ICS anomaly detection

The works presented in this thesis use a variety of data sources, ranging from public datasets to simulation environments.

Public datasets We use three public datasets for evaluation: BATADAL[1] [126], SWaT[2] [48], and WADI[3] [7]. All three datasets are provided by the Singapore University of Technology and Design iTrust Lab[4]. Each dataset contains one or more multi-day executions of an ICS, including benign executions (i.e., data from normal operation) and attacked executions (i.e., data from operations where pre-defined manipulations are performed). In the benign executions, all samples are labeled as benign (i.e., $y = 0$). In the attacked executions, samples are labeled for whether they were collected during an attack or not (i.e., $y = 1$ or $y = 0$ respectively). The attacked executions for each dataset include documentation that reports the start and end time of each attack, the component(s) that were manipulated, and the values used in each manipulation. For each system, we use the corresponding benign executions for unsupervised model training, and we use the corresponding attacked executions for evaluating our trained models.

Simulation We use two simulators to generate data for evaluation: the Tennessee Eastman Process (TEP) [17, 32] and the Digital Hydraulic Simulator (DHALSIM) [92]. TEP is a simulation of a chemical process written in C and MATLAB. DHALSIM is a simulation of a water distribution system written in Python, which can be configured to use different topologies. In

[1] "Battle of Attack Detection Algorithms" dataset
[2] "Secure Water Treatment" dataset
[3] "Water Distribution" dataset
[4] `https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/`

prior work and in this thesis, we use C-Town, a default topology that is included with DHALSIM [37, 92]. For both TEP and DHALSIM, we execute the simulator in normal operation and record its time-series process values, using this data to train anomaly-detection models. To generate attack data for TEP, we develop a module that executes pre-defined process-value manipulations during TEP's execution. We configure and perform several simulations to collect a variety of attack data. Our modified simulator is publicly available[5]. To generate attack data for C-Town, we use an existing set of DHALSIM configurations from prior work [37].

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. Although the point-F1 is commonly used to evaluate ICS anomaly detection, in Chapter 3, 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 Models for process-level ICS anomaly detection

A variety of prior work has designed and applied models for ICS anomaly detection. These models are *unsupervised*; they are trained with only benign data that does not contain any anomalies. In contrast, *supervised* learning requires explicit data labels (attack or benign). As ICS attack data is rare and difficult to generalize, unsupervised learning is commonly used.

Instead of anomaly-detection approaches that use network traffic [90] or information from computer hosts [55], this thesis focuses on process-level anomaly detection: models are trained with and predict ICS process values, such as sensor readings and actuator commands. We define the process values at a given time t as X_t. Given a d-by-h sequence of the previous h process values $(X_{t-h}, ..., X_{t-1})$ as input, the model can either predict an anomaly score a_t [12], or the model can predict the next expected set of process values X'_t, which are then compared with the next true ICS values X_t to compute a mean squared error (MSE) [23, 133]. An anomaly is then declared when the anomaly score or MSE exceeds a predefined threshold. Such anomaly-detection models include statistical models [12, 141], linear models [52, 133, 142], models based on deep learning [69, 77, 125, 148], and graph-based models [29, 54].

[5] `https://github.com/pwwl/tep-attack-simulator`

Statistical and linear anomaly-detection models Statistical and linear anomaly-detection models for ICS include PASAD [12], AR [52, 133], and GeCo [142]. AR (auto-regressive modelling) is an approach that trains a reconstruction-based linear model for each feature; AR predicts each feature value from a linear combination of its historical values [52]. Prior work has extended AR by using a stateful, cumulative sum (CUSUM) of prediction errors as a detection threshold [133]. PASAD trains an embedding into a lower-dimensional subspace for each feature; to detect anomalies, inputs are projected onto this subspace and the distance from the subspace centroid is used as an anomaly score [12]. A threshold is then set on the anomaly score using validation data, and inputs that exceed this threshold are declared as anomalous. GeCo trains a linear model for each ICS process value, by searching over process values and using only a few features as input [142].

Deep-learning-based anomaly-detection models A variety of deep-learning-based architectures are also used for ICS anomaly detection: these include autoencoders (AE) [125], convolutional neural networks (CNN) [69, 70], recurrent neural networks (RNN) [40, 77], and long-short-term memory networks (LSTM) [98, 148]. AEs are trained to reconstruct inputs through a low-dimensional representation, supporting high quality reconstruction of states similar to those observed during training, and low quality reconstruction of inputs dissimilar to those observed during training. CNNs are commonly used for image-based tasks [71, 78] but they can also be applied over time-series data with one-dimensional convolutional kernels [69]. RNNs and LSTMs are similar to CNNs but do not required fixed-time-length convolutional kernels; RNN units are trained over sequences with the ability to update or reset parameters based on time sequences [27], whereas LSTM units further include the ability to maintain parameter values over time [56].

Graph-based anomaly-detection models Graphs can be used for ICS anomaly detection by training and making predictions with graph neural networks, two prominent examples that have been applied to ICS anomaly detection are graph decision networks (GDNs [29]) and FuSAG-Net (FSNs [54]). Both GDN and FSN perform graph-based convolutions, which require a graph representation of the relationships between its input features. In the case of ICS anomaly detection which uses sensors and actuator values as features, nodes represent sensors or actuators, and edges between nodes represent causal relationships between a pair of sensors or actuators. Prior work that trains GDN and FSN dynamically learns this graph structure from ICS data while training [29, 54].

In this thesis, I evaluate how effective these models are for detecting ICS anomalies (in Chapter 3) and propose improvements to these models for the ICS anomaly detection use case (in Chapter 5).

2.5 Attribution methods for machine-learning models

Attribution methods compute the influence of a machine-learning model's input features on their predictions [4]. In this thesis, we focus on *instance-based* attributions, which compute a distinct

attribution for a single input example. Instance-based attributions depend on the input example's feature values, the weights of the model, and the chosen output of interest. Such attribution methods are further divided into two categories: *white-box* attribution methods and *black-box* attribution methods.

White-box methods White-box attribution methods use gradients computed over model parameters. Saliency maps (SM) are computed by directly computing the gradient of the output of interest (e.g., the probability of a specific class) with respect to the input features [113]. Other white-box attribution methods build on SM by introducing approaches that improve the robustness of the attribution: SmoothGrad (SG) perturbs inputs with random noise [118], integrated gradients (IG) compute gradients with respect to a reference baseline [124], and expected gradients (EG) compute gradients over an expectation of randomly sampled reference baselines [38].

Black-box methods Black-box attribution methods do not use model parameters. Instead, they approximate the behavior of the model around the provided input. This approximation is built from several model queries drawn from a neighborhood surrounding the input of interest [50, 84, 100]. Several black-box attribution methods exist and the primary differences amongst them come from how the model approximation is constructed. For example, LIME uses the coefficients of a linear regression [100], SHAP uses Shapley values from game theory [84], and LEMNA uses the coefficients of a fused Lasso model and a Guassian mixture model [50].

In this thesis, I investigate how attribution methods can be applied to and designed for ICS anomaly detection. I propose that attribution methods can be used to help identify which component in an ICS was manipulated by an adversary, potentially aiding practitioners when responding to anomalies. I compare existing models (in Chapter 4) and propose improvements to them (in Chapter 5) when attributing ICS anomalies.

2.6 Challenges faced by practitioners in contexts similar to ICS anomaly detection

A variety of prior work has studied the challenges that practitioners face when (i) working with ICS, (ii) using and deploying machine-learning-based solutions for computer security, and (iii) responding to security alarms in real-time.

Working with ICS Prior work has studied the security-related perspectives of professionals that work with the electric power grid [44, 114]. Although the electric power grid is a prevalent example of an ICS, these works do not focus on real-time anomaly detection and remediation, instead focusing on how practitioners perceive the severity of attacks and vulnerabilities. Other prior works have studied usability and maintenance challenges in industrial control [18, 116]. These works identify relevant practical and organizational challenges within ICS but are not focused on their implications on security. In this thesis, I explore challenges similar to those

reported in prior work [18, 44, 114, 116] but focus specifically on how these challenges affect the adoption of AI for protecting ICS.

Security practitioners perceptions' of machine-learning-based tools Prior work has studied the perceptions of IT security practitioners, either focusing on their perceptions of a specific machine-learning-based tool [94, 96] or by studying their perceptions of machine learning in general [89]. These works identify promising uses for machine-learning-based explanations to help practitioners understand security events, but also identify concerns with accuracy, trust, and usability.

Mink et al. compare rule-based approaches and machine-learning-based approaches, suggesting that they differ in terms of false positives, false negatives, interpretability, and required domain expertise [89]. Ultimately, they suggest that a hybrid solution that leverages rules and machine learning is likely required for most tasks to balance tradeoffs. In this thesis, I similarly explore perceptions of machine learning for security tasks, but focus specifically on machine learning for securing ICS.

Real-time security alarms in security operations centers Security operations centers (SOCs) are organizational units that monitor organizations for malicious and potentially harmful activity in real-time [147]. A variety of prior work studies the challenges that SOC operators face in their day-to-day roles: alarm response [9], alarm ruleset management [134], and organizational challenges [67]. Common themes include operator burnout, a high volume of false alarms, difficulty interpreting alarms, and difficulties when maintaining rule sets.

Although these prior works cover various parts of the ICS-anomaly-detection workflow, none of them are focused on our specific context, in which anomaly-detection systems raise alarms when ICS are attacked in real-time. SOCs and IT systems operate at high levels of the Purdue model (i.e., level 4) and do not directly interact with operational technology (OT), such as PLCs or SCADA. Furthermore, IT and OT professionals exhibit different cultural beliefs about ICS security [44]. In this thesis, I explore the alarm-diagnosis workflow and perspectives specifically for the ICS anomaly-detection context (in Chapter 6).

Chapter 3

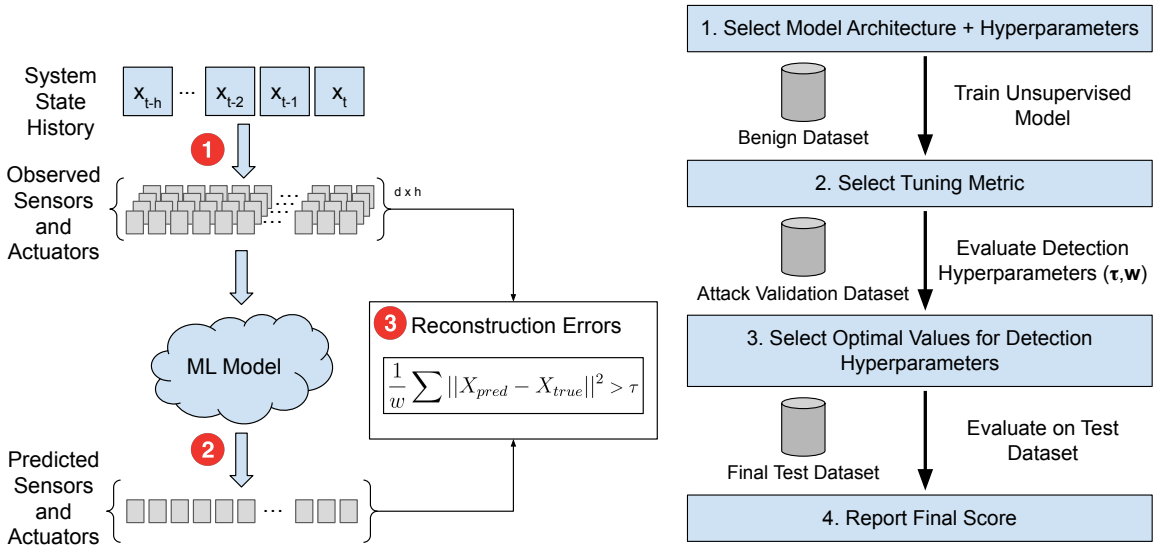
Comparing models and techniques used for detecting ICS anomalies

Several approaches based on machine learning (ML) have been proposed for ICS anomaly detection, including those based on autoencoders [36, 125], convolutional neural networks [68, 69, 70], and LSTMs [39, 148]. These approaches share many common datasets, yet make differing conclusions about which architectures are best. Thus, when practitioners are determining which anomaly-detection approaches to adopt, it is unclear which models and techniques should be used. In this chapter, we systematize prior work in ICS anomaly detection to examine why these discrepancies in findings exist, and we determine the models and techniques that are most effective for ICS anomaly detection. This work described in this chapter is published in the *Proceedings of the 27th European Symposium on Research in Computer Security* (ESORICS 2022) [41].

3.1 Introduction

When using a reconstruction-based anomaly-detection approach, practitioners must: (1) select an ML model architecture (e.g., convolutional neural networks), (2) select hyperparameters for the model (e.g., the number of hidden layers in the model), (3) collect a sufficient volume of benign ICS process data, (4) train an ML model to predict expected process values, and (5) tune detection hyperparameters (e.g., the threshold for an anomaly) to turn process-value predictions into alarms. Design decisions made in each of these steps play a role in the final performance of the anomaly-detection model.

Despite the variety of work in ICS anomaly detection, there is no consensus on what solutions are best. Proposed approaches use different ML model architectures (e.g., autoencoders [36, 125], CNNs [68, 70], LSTMs [39, 148]), use different datasets [7, 48, 126], and employ different data pre-processing and training techniques. As a result, when one approach is reported to outperform another, it is not clear what is responsible for the improved performance. In this work, we perform a comprehensive, empirical evaluation of techniques across datasets commonly used in reconstruction-based ICS anomaly detection. Perhaps surprisingly, we find that the best performance can be achieved by most models, including models that are far smaller (i.e., contain



(a) Anomaly detection process

(b) Model optimization pipeline

Figure 3.1: 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.

fewer parameters) than the previously reported best models. We also 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 design consideration is the choice of metric used to tune and evaluate anomaly-detection models [75, 143]. Typically, prior work equally penalizes false positives and false negatives with the point-F1 score computed on a per-timestep basis [119]. However, since ICS attacks take place over a sequence of timesteps [7, 48] and timely detection of attacks is important [59], ICS anomaly-detection models should be evaluated over temporal ranges. Unlike point-F1, *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 [59, 76, 127]. When used to tune ICS anomaly-detection models, range-based metrics produce models that perform better on metric-specific tradeoffs. We also show that using range-based metrics to evaluate anomaly-detection models gives a better understanding of what models are optimal.

3.2 Describing the reconstruction-based ICS anomaly detection process

An anomaly detector reconstructs ICS system states to determine if an anomaly is occurring. Figure 3.1a outlines this process.[1] First, system states $\vec{X}$ over the previous h timesteps are col-

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

lected 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 $(\vec{e_0}, \vec{e_1}, \ldots, \vec{e_t})$: $y'_t = 1$ 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. Figure 3.1b shows the 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.

In this work, we independently evaluate both steps. In Section 3.4, 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 Section 3.5, we keep the underlying trained model constant and compare how the choice of tuning metric affects detection hyperparameter tuning. Lastly, in Section 3.5.4, we show how the choice of tuning metric affects both the optimal model hyperparameters and detection hyperparameters in an end-to-end optimization.

3.3 Comparing methodologies from prior work

In this section, we overview the prior work in ICS anomaly detection across BATADAL, SWaT, and WADI—three commonly used ICS datasets.

3.3.1 Comparing models, hyperparameters, and metrics

We first compare the models and model hyperparameters used in prior work. Table 3.1 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 [2, 36, 70], 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,

Table 3.1: 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**			**Tuning**	**Evaluation**	**Source**
	B	**S**	**W**	**Metric**	**Metric(s)**	
AE: 3-layers	●	●	●	FPR	Precision, Recall Point-F1	[70]
AE: 4-layers		●		None	Precision, Recall Point-F1, **Numenta**	[107]
AE: 5-layers	●			Point-F1	Precision, Recall Point-F1	[125]
AE: 5-layers	●		●	None	Precision, Recall Accuracy, Point-F1	[36]
CNN: 8-layers, 32 filters		●		**Range-F1**	**Range-F1**	[68]
CNN: 8-layers, 32 filters	●	●	●	FPR	Precision, Recall Point-F1	[70]
LSTM: 2-layers, 256 units		●	●	None	TPR, **Norm-TPR** FPR, **Atk TP**	[39]
LSTM: 3-layers, 100 units		●		Point-F1	Precision, Recall Point-F1	[61]
LSTM: 3-layers, 100 units		●		None	**Atk TP, Atk FP**	[47]
LSTM: 4-layers, 64 units		●		None	**Atk TP, Atk FP**	[64]
LSTM: 4-layers, 512 units		●		None	CM, Point-F1, **Atk TP**	[98]
LSTM: 4-layers, 512 units		●		Point-F1	Point-F1	[148]
1-class SVM		●		Point-F1	Point-F1	[61]
DNN: 3-layer	●			None	CM, TPR, TNR	[3]
Custom wide and deep CNN		●	●	None	Precision, Recall Point-F1, **Atk TP**	[2]
GAN		●	●	Point-F1	Precision, Recall Point-F1	[79]
Bayesian Network		●		None	**Atk FP, Atk TP FP length, Coverage**	[80]

recall, and latency in anomaly detection. Across this prior work, only one tunes with a range-based metric [68]; 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 [39, 68, 80, 107]. In Section 3.5, 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.

Table 3.2: 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.

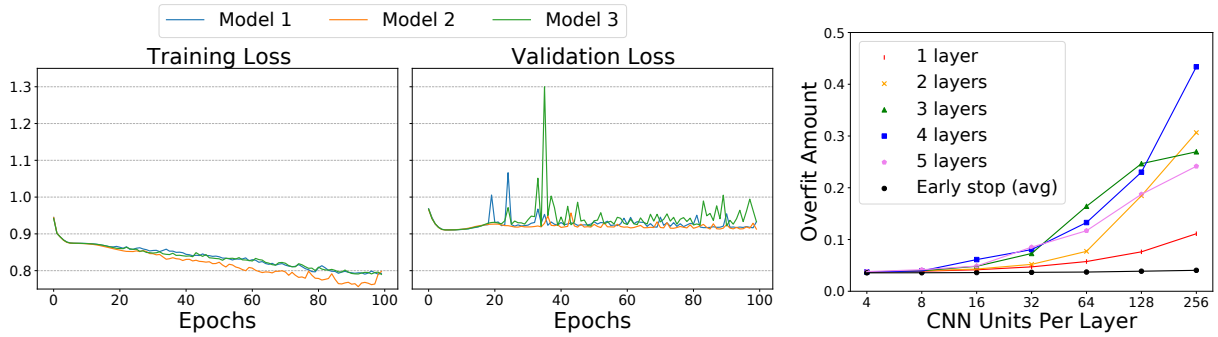
Feature Selection	Attack Cleaning	Benign Data Shuffling	Early Stopping	Source
○	○	**?**	○	[2]
○	○	●	○	[3]
○	○	**?**	●	[36]
○	○	○	○	[39]
◐	◐	**?**	●	[47]
○	●	○	●	[61]
○	●	○	○	[64]
○	◐	**?**	●	[69]
●	○	**?**	○	[68]
◐	○	○	○	[79]
○	●	○	○	[80]
●	●	○	○	[98]
○	○	○	○	[107]
○	○	**?**	●	[125]
●	◐	**?**	○	[148]

3.3.2 Comparing training and data-processing techniques

We also compare prior work by their training and data-processing techniques. In this work, we identified four techniques that enhance the quality and reproducibility of anomaly detection performance: (i) selecting features, (ii) shuffling benign data, (iii) cleaning attack start and end times, and (iv) early stopping while training. These techniques are necessary to fairly compare anomaly-detection approaches, as they improve the quality and consistency of anomaly-detection results. Table 3.2 shows, for each prior work, which key techniques are used.

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

Key Technique #1: Feature selection. In WADI and SWaT, some benign-labeled test data appears significantly different from benign-labeled training data [70, 129]. 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*) [70] 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 [70]. We found that feature selection is only effective on the SWaT dataset, so we only use feature selection for SWaT.



(a) Comparing losses for a 4-layer, 64-unit CNN.

(b) Overfit by model size.

Figure 3.2: 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.

Key Technique #2: Attack cleaning. Some attacks in the SWaT dataset do not execute as described [61, 64]: although labelled as attacks, the SWaT description [48] 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 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 [148]. Hence, we recommend that the times of the labelled attacks be corrected.[2]

Key Technique #3: Benign data shuffling. When most prior work divides the benign dataset into training and validation portions, it divides by a fixed time [39] 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.

Key Technique #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. Figure 3.2a 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, Figure 3.2b 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.

Although some prior works evaluate multiple ML model architectures [2, 69, 148], no work covers the full selection of model architectures, datasets, and pre-processing techniques, making

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

it is unclear what approaches are optimal across all settings. We therefore establish a standardized evaluation of three model architectures (AE, CNN, LSTMs) across three datasets (BATADAL, SWaT, and WADI) using all four key techniques. We perform a comprehensive comparison of models proposed in prior work to determine which models are most effective for ICS anomaly detection.

3.4 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 Figure 3.1b. We explain our experimental setup in Section 3.4.1 and present our findings in Section 3.4.2.

3.4.1 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 Section 3.3.2, we described techniques that impact the quality and reproducibility of results but were used inconsistently in prior work. Thus we use the described techniques in our evaluation: data pre-processing through feature selection, benign data shuffling, attack cleaning, and early stopping, as they improve the quality and consistency of anomaly-detection results.

Model Hyperparameter Tuning. We perform a hyperparameter search for three ML model architectures: autoencoders, CNNs, and LSTMs. For autoencoders, we vary the number of hidden layers in the encoder and 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: $\{lr = 0.001, \beta_1 = 0.9, \beta_2 = 0.999\}$. We use a batch size of 512 samples during training and train each model 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 [61].

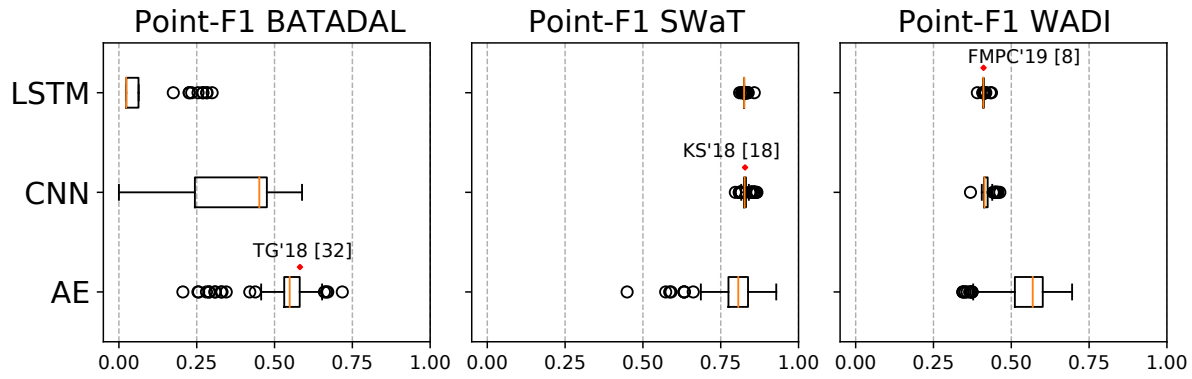


Figure 3.3: The final point-F1 scores of each model when trained and tuned on three experimental ICS datasets. For each dataset, a model hyperparameter setting from prior work is included for comparison. When using the point-F1, the performance of AEs vary greatly, and most LSTM and CNN configurations perform similarly.

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 3.1 shows is commonly used in prior work.

3.4.2 Optimization results

Figure 3.3 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 [125], an 8-layer, 32-unit CNN with a history of 200 [68], and a 2-layer, 256-unit LSTM with a history of 50 [39]. Figure 3.3 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 $\sim$48,000 datapoints, compared to $\sim$500,000 in SWaT and $\sim$1,000,000 in WADI); in our surveyed prior work, only one work trains a CNN or LSTM on BATADAL [70]. In Section 3.5.4, 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

[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.

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 [39, 68, 125].

Finding 2: 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 [68], 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). 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 Section 3.5.3, we show that tuning models with range-based metrics can produce outcomes that more meaningfully address ICS anomaly-detection objectives.

Finding 3: 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.

3.5 Tuning and evaluating with range-based metrics

In this section, we first describe, in Section 3.5.1, the shortcomings of point-F1, which is commonly used by prior work in ICS anomaly detection. We introduce range-based metrics in Section 3.5.2. In Section 3.5.3, we show how range-based metrics affect detection hyperparameter tuning and in Section 3.5.4 we show how they affect what ML model architectures and hyperparameters are optimal.

3.5.1 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 [59]. 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

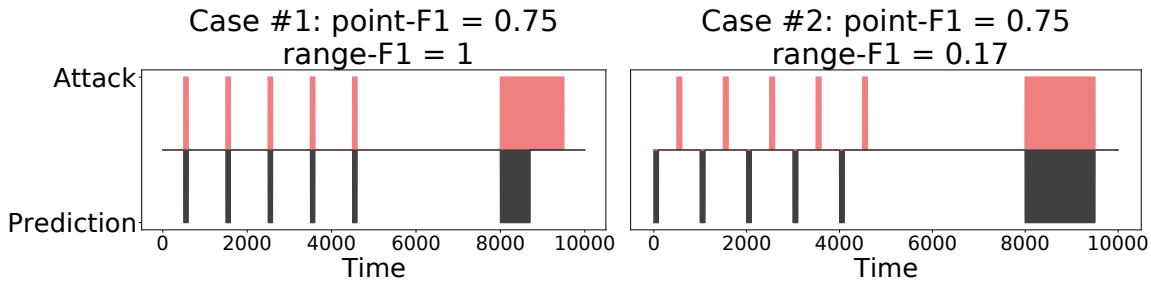


Figure 3.4: Two detection examples: in each case, the x-axis represents time and the y-axis shows attacks (top, red) and attack predictions (bottom, grey). In the example on the left (case 1), all attacks are detected with no false positives, while in the example on the right (case 2) only one attack is detected, with five false positives; yet, the point-F1 scores are the same.

prediction within the attack may be sufficient. (4) The point-F1 score does not consider *when* in the attack the detection occurs [76]. 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 Figure 3.4. 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 [59, 76, 127]. We define these metrics in Section 3.5.2 and evaluate their implications in Sections 3.5.3–3.5.4.

Finding 4: The point-F1 score gives a misleading sense of performance for many time-series-based detection tasks.

3.5.2 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 Sections 3.5.3–3.5.4, 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 [127] and (2) the Numenta anomaly score [76], 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 acores 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 [127]. 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 [76] 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 [115] 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 Section 3.5.2 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.

Table 3.3: For each optimal model proposed in 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 Architecture	**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 default configuration of the Numenta anomaly score sets $\kappa = 5$ and positions the sigmoid at the 50% point of each labeled anomaly [76]. 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 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.

3.5.3 Using range-based metrics to tune detection hyperparameters

In contrast to Section 3.4, 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 [68], a 5-layer, 2-compression AE trained on BATADAL [125], and a 2-layer, 256-unit LSTM trained on WADI [39].

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 3.3. Prior work hypothesized that a TP:FP ratio of 1 or greater was acceptable and used the TP:FP ratio as a success metric [39]. 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.

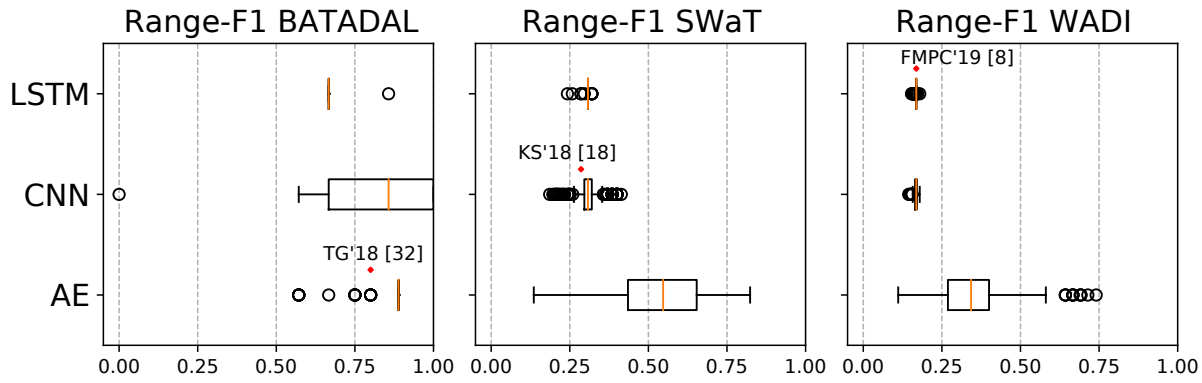


Figure 3.5: The final range-F1 scores of each model when trained and tuned on three experimental ICS datasets. For each dataset, a selected model hyperparameter setting from prior work is included for comparison.

Table 3.3 shows the final detection results for our additional metrics (defined in Section 3.5.2) after tuning the SWaT CNN from prior work [68]. 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 5: 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.

3.5.4 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 Section 3.4. Compared to the point-F1, using a range-based metric for tuning and evaluation consistently leads to different conclusions about which models are optimal. Figure 3.5 shows the final range-F1 scores after repeating the experiments described in Section 3.4.2: 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 (Section 3.4.2): 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 6: 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.

3.6 Summary

In this chapter, we determine what factors (e.g., the choice of model, training techniques, metrics, etc.) are most important when training ICS deep-learning-based anomaly-detection models. Contrary to what is suggested by prior work, we find that the choice of deep-learning model does not have a strong effect on anomaly-detection performance; the best performance can be achieved over a range of model architectures and hyperparameters. Instead, 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 for the point-F1, and better success measures are needed to practically tune and evaluate ICS anomaly-detection models.

Chapter 4

Evaluating attributions for ICS anomaly detection

A detected ICS anomaly requires follow-up diagnosis and remediation. In this chapter, we investigate if and how outputs from the anomaly-detection model can be used to assist with alarm remediation. We evaluate if attributions of anomalies detected by anomaly-detection models can be used to identify the manipulated component in an ICS attack. We first evaluate if prior approaches are sufficient for attribution, before performing a broader evaluation across anomaly-detection models, attribution methods, and ICS attack properties. We identify factors that affect attribution performance and make recommendations that make attributions more effective for ICS. The work described in this chapter is published in the *Proceedings of the 31st Network and Distributed System Security Symposium* (NDSS 2024) [42].

4.1 Introduction

In Chapter 3, we described our evaluation of a variety of anomaly-detection models for ICS. Each of these models is reconstruction-based: they predict a set of ICS values and detect anomalies by comparing these predictions to their observed values. In this chapter, we investigate if outputs from anomaly-detection models can be used to attribute the causes of anomalies by identifying the component (i.e., the sensor or actuator) that was manipulated in an ICS attack.

We first investigate if, as suggested in prior work in small-scale evaluations [58, 69], per-feature error ranking can be used for attribution. We expand on prior evaluations by evaluating with over 150 diverse attacks; this work is the first to systematically evaluate attributions for ICS anomalies. After finding that raw-error ranking fails to generalize across a more diverse set of attacks, we investigate if machine-learning-based *attribution methods* proposed in prior work [38, 50, 84, 100, 113, 118, 124] can be used to effectively attribute ICS anomalies. Attribution methods are predominately designed for and evaluated on image classification, so we must first adapt these methods for the ICS anomaly-detection domain.

Attacks in ICS datasets vary in how they are performed, such as the manipulation magnitude and the type of component that is attacked. We perform a statistical analysis of attribution accuracy across these factors to identify which attribution methods perform best for specific types

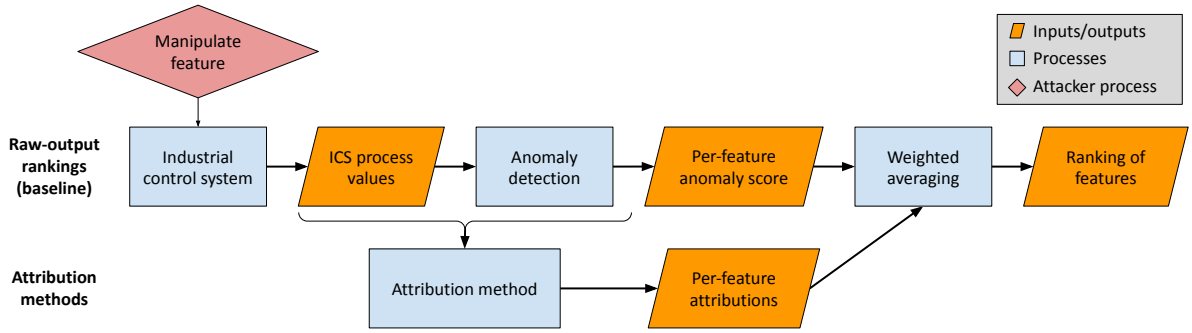


Figure 4.1: In this work, we describe and evaluate each step when attributing an ICS attack. Raw-output rankings (top) are the baseline method from prior work. We introduce ML-based attribution methods (bottom) as an alternative. We average and sort attribution scores to produce an overall ranking of suggested features for investigation.

of attacks. Since we find that different methods perform best for different types of attacks, we ultimately design an ensemble of attribution methods and show that it outperforms all other individual attribution methods.

4.2 Methodology

In this section, we describe our methodology, which is composed of several steps; Figure 4.1 shows our overall methodology in detecting and attributing ICS anomalies.

First, we prepare anomalous data for evaluating anomaly detection and attribution. Section 4.2.1 describes the datasets used in our work, spanning both publicly collected and newly generated datasets. Second, we train ICS anomaly-detection models, closely following techniques from prior work, as described in Section 4.2.2. Third, we compute attributions of anomalies, using baseline methods from prior work (raw-error ranking) and ML-based attribution methods; Section 4.2.3 describes how we adapt attribution methods to account for the time-series and unsupervised aspects of ICS anomaly detection. Finally, we sort, score, and average attributions to produce an overall ranking of features for investigation; Section 4.2.4 describes *AvgRank*, our evaluation metric that captures attribution accuracy over a broad set of anomalies.

4.2.1 Datasets used for training and evaluation

We evaluate against two groups of anomalies: (1) real attacks found in public ICS datasets [7, 48] and (2) synthetic anomalies created with an open-source ICS simulator [17].

Real attacks

In this work, we use SWaT [48] and WADI [7], as they are commonly used for training deep-learning-based anomaly-detection models [29, 41, 69, 98, 148]. In the respective attack datasets of SWaT and WADI, each attack is performed by the system operator: the value of one or more sensors or actuators is manipulated for a fixed duration while the response from the physical ICS

is captured. Each attack's start time, end time, and location (which sensors/actuators are manipulated) are documented. In total, the SWaT and WADI datasets contain 47 attacks for testing: 32 in SWaT and 15 in WADI. These datasets also contain anomalies where multiple features are manipulated simultaneously; when evaluating attributions, we consider each manipulation independently. Across the 47 attacks in our datasets, 67 manipulations (43 in SWaT, 24 in WADI) are performed, forming the set of *real attacks*.

Synthetic anomalies

To further increase anomaly diversity for our attribution evaluation, we created an additional dataset of anomalies by modifying a simulated ICS. We use a public MATLAB 7.0 simulator of the Tennessee Eastman process (TEP) [17], an anonymized chemical process [32].

We implement a MATLAB module that interfaces with the TEP simulation and manipulates process values, based on an attacker model used in prior work [23, 72]: for a feature j, a benign sequence $x_j(t)$ is replaced with a manipulated sequence $x'_j(t)$ for a time period T_a.

$$\tilde{x}_j(t) = \begin{cases} x_j(t) & \text{for } t \notin T_a \\ x'_j(t) & \text{for } t \in T_a \end{cases}$$

Using the modified simulator, we systematically perform ICS feature manipulations to generate a set of *synthetic anomalies*. For each anomaly, we execute a 40-hour TEP simulation, manipulate a chosen sensor or actuator, and record the resulting system states. For every sensor and actuator in TEP, we simulate four anomalies with different magnitudes, creating 89 anomalies[1].

Although these anomalies are generated synthetically, they simulate physically realizable anomalies. For each manipulation, the data collected from non-manipulated features were produced as the output of the physical process (i.e., the chemical process) responding to the manipulation. In cases where manipulations produced runtime errors in the MATLAB simulation, we excluded them from our dataset. Maintaining physical realizability is important to ensuring that the executed attacks are possible, and is essential when evaluating ML-based approaches in other security contexts, such as face recognition [109] and malware detection [128].

Though the anomalies in this dataset do not necessarily correspond to an intentional ICS attack outcome, they are genuine statistical anomalies (i.e., 95th percentile event or higher) that share common properties (e.g., manipulation patterns and magnitudes) with the manipulations observed in the real attack dataset. We use the synthetic anomalies to support a systematic analysis of the relationship between manipulation properties and attributions.

Defining manipulation properties

We define each anomaly by its manipulation magnitude and the type of feature that is attacked. The anomalies contained in all datasets are summarized in Table 4.1. In Section 4.4.2, we identify properties that significantly affect attribution accuracy and which attribution methods are optimal.

[1] 11 out of 100 manipulations triggered a shutdown sequence, causing the MATLAB simulator to exit and preventing data collection.

Table 4.1: A summary of the manipulations used for evaluation, across a set of real attacks from prior work (SWaT, WADI) and a set of synthetic anomalies generated with a public simulator (TEP).

Dataset	Magnitude	Location	Total
SWaT	0.06–36.31 std devs	24 sensors 19 actuators	43
WADI	0.5–91.00 std devs	16 sensors 8 actuators	24
TEP	2–5 std devs	56 sensors 33 actuators	89

We define the *manipulation magnitude* using the difference in standard deviations between feature j's benign distribution and its replaced value:

$$\text{magnitude} = \frac{|\max(x'_j(t)) - \text{ mean}(x_j(t))|}{\text{stddev}(x_j(t))}$$

The attacks in the SWaT and WADI datasets are performed with manipulations that span a wide magnitude range, from small manipulations within the benign distribution (magnitude of 0.06) to large manipulations outside the benign distribution (magnitudes over 35). When generating synthetic anomalies, we perform manipulations at four different magnitudes: +2, -2, +3, and +5. Our synthetic anomalies are performed within the distribution of magnitudes observed in the real attack dataset, which has an average magnitude of 3.25 standard deviations.

We also define attacks by the *type of feature* that is manipulated. Features in TEP are grouped into three categories: *actuators*, which directly control the chemical process; *sensors*, which are read by controllers to compute future actuator values; and *out-of-loop features*, which do not impact the ICS process. For TEP, we only perform manipulations on sensors that result in changes to the physical process. SWaT and WADI provide documentation for each feature and each attack: we manually verified that each manipulation was executed as described and that each manipulation affects the physical ICS process.

4.2.2 Implementing ICS anomaly detection

In this section, we describe our implementation of statistical and deep-learning-based anomaly-detection models for ICS, closely following the methodology from prior work.

Statistical and linear anomaly detection

We implement two statistical and linear anomaly-detection methods from prior work: PASAD [12] and AR [52].

We use the open-source PASAD implementation[2] by Aoudi et al. and tune it for each dataset in our use case. PASAD is parameterized by the training length N, the input window length

[2] `https://github.com/mikeliturbe/pasad`

Table 4.2: The number of detected attacks (at least one example exceeds the MSE threshold), when using a validation-error-based tuning for the error threshold (99.95%).

	Total	Total Detected		
		CNNs	GRUs	LSTMs
SWaT	43	33	20	34
WADI	24	15	8	15
TEP	89	55	58	64
All datasets	156	103	86	113

(called lag) L and the statistical dimension r. We refer to the anomaly-detection methodology and configurations by Aoudi et al. [12], using the same default parameters for the SWaT and TEP datasets. Since WADI is based on the SWaT system, we opt for the same parameter values for WADI and SWaT. For SWaT and WADI, our parameters are: $N = 30000$, $L = 5000$ and $r = 10$; for TEP, our parameters are: $N = 10000$, $L = 5000$ and $r = 16$.

The original AR implementation[3] is in C++ , so we opt to implement a linear model on our own in Python. AR is parameterized by p, the number of prior states used in the linear model. Based on the default settings, we use $p = 10$, and train our linear models with the Adam optimizer.

Deep-learning-based anomaly detection

A variety of prior work performs hyperparameter tuning across model architectures to find optimal deep-learning-based anomaly-detection models for ICS [69, 148], including the work described in Chapter 3; *this is not the focus of this chapter*. Nevertheless, we perform a best-effort training of anomaly-detection models for attributions.

We evaluate across three model architectures: convolutional neural networks (CNNs) [69], gated-recurrent-unit networks (GRUs) [40], and long-short-term-memory units (LSTMs) [98, 148]. To best compare across model architectures, we use a similar model size for each architecture: we train a 2-layer, 64-unit, 50-length-history model for each combination of dataset (SWaT, WADI, TEP) and architecture (CNN, GRU, LSTM), with the default Adam optimizer. For each training dataset, we use 80% of the benign dataset for training and 20% of the benign dataset for validation. As suggested by our findings in Chapter 3, we implement early stopping, which halts training when the validation loss stops decreasing. Finally, we test the anomaly-detection models against the manipulations from SWaT, WADI, and TEP. The results are shown in Table 4.2. We find that the size of the underlying reconstruction model has a small effect on detection accuracy; each model's benign validation error is below 0.25.

4.2.3 Attribution methods for ICS anomaly detection

After implementing ICS anomaly-detection models (as described in Section 4.2), in this section we describe how we adapt attribution methods for anomaly-detection outputs.

[3] `https://github.com/RhysU/ar`

We provide the technical definitions of our attribution methods, adapted for ICS anomaly detection. Each attribution method A requires an anomaly-detection model $F(x_{t-h}, ..., x_{t-1}) \rightarrow \hat{x}_t$ and time-series input $X_e \in \mathbb{R}^{dxh}$: computing an attribution $A(X_e)_j$ for feature j.

Raw-error ranking Our baseline attribution method uses the raw, per-feature anomaly scores produced by each model as the attribution, as suggested in prior evaluations of ICS anomaly-detection attribution [58, 69].

For AR and deep-learning-based anomaly detection models, we use the per-feature prediction error (i.e., before taking the average for MSE) between input x_j and its prediction $\hat{x}_j$ as the anomaly score. Each error s_j corresponds to the anomaly score for an ICS feature j:

$$s_j = (x_j - \hat{x}_j)^2.$$

With PASAD, each feature's input x_j produces a departure distance that represents its deviation from normal; we use the departure distance as the anomaly score. Details on how to compute projection matrix P and subspace centroid $\tilde{c}$ can be found in the original publication [12].

$$s_j = (\tilde{c} - Px_j)^2.$$

Counterfactuals Given input X_e and baseline X_b, a counterfactual attribution is computed by changing the value of each feature and measuring the change in the MSE. We use the feature-wise average benign value as X_b and define a masking function $M_j(X)$, which removes all but the j-th feature from X. We define the additive counterfactual A_A:

$$A_A(X_e)_j = MSE(X_b - M_j(X_b) + M_j(X_e)) - MSE(X_b)$$

We define the subtractive counterfactual A_S:

$$A_S(X_e)_j = MSE(X_e) - MSE(X_e - M_j(X_e) + M_j(X_b))$$

Saliency map [113] The saliency map $A_{SM}(X)$ is the product of X_e and the gradient of the quantity of interest with respect to the input window, computed at X_e. We compute the gradient with respect to the MSE:

$$A_{SM}(X_e) = X_e \times \frac{\partial MSE(X_e)}{\partial X}$$

SmoothGrad [118] Prior work found that saliency maps are sensitive to small input changes; in response, SmoothGrad averages saliency maps over multiple perturbations of X_e.

$$A_{SG}(X_e) = X_e \times \frac{1}{n} \sum \frac{\partial MSE(X_e + \varepsilon)}{\partial X}, \varepsilon \sim N(0, \sigma)$$

SmoothGrad is defined by n, the number of samples used, and σ, the sampled noise variance. We use the suggested values $\sigma = 0.1 * (X^{max} - X^{min})$ and $n = 50$.

Integrated gradients [124] Integrated gradients calculate the change in a quantity of interest between X_e and a baseline X_b, producing more meaningful results. Attributions are computed through an approximate path integral that interpolates between X_b and X_e.

$$A_{IG}(X_e) \approx (X_e - X_b) \times \frac{1}{n} \sum_{k=1}^{n} \frac{\partial MSE(X_b + \frac{k}{n}(X_e - X_b))}{\partial X}$$

Using larger n increases the accuracy of the path-integral estimate. In our work, we use $n = 200$ and the feature-wise average benign value as the baseline X_b.

Expected gradients [38] Instead of assuming a single baseline, expected gradients use samples from the training distribution D. Attributions are computed as the expectation over baseline examples and interpolation points. The expectation is approximated by averaging over estimates: for each estimate i, a sample baseline X_{bi} is drawn from the benign training dataset and an interpolation point α_i is drawn from the uniform distribution.

$$A_{EG}(X_e) \approx \frac{1}{n} \sum_{i=0}^{n} [(X_e - X_{bi}) \frac{\partial MSE(X_{bi} + \alpha_i(X_e - X_{bi}))}{\partial X}]$$
$$\alpha_i \sim U(0,1), X_{bi} \sim D$$

To sample baselines from our training dataset, we sample a timestep t from the benign dataset and use its process values and corresponding history. As the number of samples n increases, the stability of the expected gradients increases. We use the suggested $n = 200$ for convergence.

LIME [100], SHAP [84], and LEMNA [50] LIME, SHAP, and LEMNA are prominent, black-box attribution methods. These techniques use perturbed samples to train a local, linear approximation around an input, and use the approximation model's coefficients as attributions. We use public LIME[4] and SHAP[5] libraries maintained by their original authors. We implement LEMNA based on its published description [50] and public code examples[6]. Each of the described implementations assumes a single-output classification or regression task. To adapt these implementations for ICS anomaly detection, we use a technique from prior work [11, 58]: for a given input, we identify the feature with the highest prediction error and compute its LIME, SHAP, or LEMNA attributions. Thus, we adapt the anomaly-detection task as a single-output regression task to comply with the design and API of LIME, SHAP, and LEMNA.

4.2.4 Evaluation metric for attributions: AvgRank

Prior work that evaluated explanations of ICS anomaly detection uses a mix of qualitative evaluations of visualizations and analyses of individual attacks [29, 58]. To quantitatively compare attribution methods over full datasets, we propose the metric *AvgRank*. Across a set of ICS

[4] `https://github.com/marcotcr/lime`
[5] `https://github.com/slundberg/shap`
[6] `https://github.com/Henrygwb/Explaining-DL`

attacks, AvgRank represents the average ranking of the manipulated feature when features are ranked by their attributions.

For a given anomaly's attribution s, we sort each feature's attribution s_j in descending order and identify the placement of the manipulated feature j'. Since the analysis in this work is across three ICS of varying dimension, we normalize rankings by dividing the placement by the number of features in the evaluated ICS dataset. We then report the average placement across all anomalies for a given attribution method.

This produces the AvgRank: a score $\in [0, 1]$ (lower is better) that represents the average proportional ranking of attacked features when identified by an attribution method. In other words, an attribution method with an AvgRank of 0.2 indicates that this attribution method will, on average, place the attacked feature in the top 20% of features.

We design AvgRank with inspiration from prior work that proposed criteria for explanations of ML models when applied to security-relevant tasks [139]. Although these proposed criteria assume a classification task, some of them are relevant for attributions of ICS anomaly detection. First, *descriptive sparsity* requires that attributions identify a small set of features; our proposed use of attributions filters out sensors and actuators in an ICS. AvgRank could be interpreted as the average number of sensors and actuators that would need to be displayed in an anomaly alert to ensure that the manipulated feature is shown. Second, *completeness* requires that attribution methods perform well over a variety of inputs; AvgRank measures performance over a set of anomalies. In Section 4.4.2, we demonstrate the importance of evaluating attributions over diverse datasets by using AvgRank to reveal discrepancies in attribution accuracy across attack properties.

4.3 Results: Evaluating attributions of ICS anomalies

In this section, we report on our evaluation of two types of attribution method for ICS anomaly detection:

- Raw-error ranking: a baseline method that ranks features in descending order by their reconstruction error
- ML-based attribution methods: attribution methods from other ML domains, adapted for ICS anomaly detection

Section 4.3.1 describes our evaluation of attribution methods using strategies from prior work: we find that raw-error rankings perform much less well that previously reported, and ML-based attribution methods also perform worse than anticipated.

4.3.1 Assessing prior attribution strategies

We first describe our results for raw-error ranking, which ranks features in descending order by their error (i.e., by the amount they deviate from the predicted value). We apply raw-error ranking to all detected attacks in all three datasets: 128 attacks for the AR model, 93 attacks for PASAD, 103 attacks for the CNN, 83 attacks for the GRU, and 113 attacks for the LSTM. Table 4.2 shows the breakdown of detected attacks. We find the first detection point for each labeled attack and use the per-feature reconstruction errors at that timestep as attributions. We rank all features by

Table 4.3: Top-k feature attribution accuracy and AvgRank for the baseline attribution strategy from prior work (ranking features by raw error at detection time). We find that this strategy performs worse than previously reported; for all methods, less than 40% of all attacks are identified by the highest score.

	Total	Top-1	Top-5	Top-10	AvgRank
CNNs	103	40 (39%)	59 (57%)	70 (68%)	0.187
GRUs	86	26 (30%)	54 (63%)	62 (72%)	0.141
LSTMs	113	27 (24%)	62 (55%)	75 (66%)	0.171

descending attribution and use the rank of the manipulated feature to compute AvgRank (e.g., an AvgRank of 0.25 implies that the manipulated feature is on average, ranked within the top 25%), repeating the process for each anomaly-detection model.

To find the detection points for our deep-learning-based models, we use each model's 99.95-th percentile validation MSE as a threshold, employing a common strategy from prior work [69, 148]. For the AR model and PASAD, we use the 99.5-th percentile validation error, as very few anomalies are detected at the 99.95-th percentile threshold. At test time, an anomaly is detected if any input within the labeled anomaly region produces an MSE above the threshold.

Table 4.3 shows the accuracy of the prior attribution strategy for each anomaly-detection method, spanning both statistical methods and deep-learning-based models (CNNs, GRUs, and LSTMs). Although prior work has reported high attribution accuracies (e.g., 80% accuracy within the top few features[7] [69]), we find that raw-error rankings are not as effective as reported; for all models, less than half of all attacked features are correctly identified by the highest-error feature, and at least one quarter of attacked features could not be identified within the top 10 features. Across all three deep-learning-based models, the AvgRank ranges from 0.14 to 0.19; in other words, on average, the manipulated feature is ranked within the top 14–19%, which is far more than the top few, as suggested in prior work.

Although the AR model and PASAD are effective at detecting attacks, their attributions perform far worse than for the deep-learning-based models; their AvgRank is much higher (over 0.3) and over 50% of all attacks are not correctly attributed by the top 10 features. This is likely because AR and PASAD model each feature independently, meaning that they cannot consider inter-feature relationships when performing anomaly detection. Thus, when an ICS feature is manipulated and subsequent components respond to the change, AR and PASAD identify are particularly likely to identify those additional features as anomalous.

4.3.2 Selecting attribution methods with a counterfactual benchmark

Before comparing raw-error ranking attributions with ML-based attribution methods, we select a group of best-performing ML attribution methods as candidates for comparison. To select these attribution methods, we design and use a counterfactual benchmark to systematically compare attribution methods. Figure 4.2 shows the result of this benchmark: We find that the saliency map (SM), SHAP, and LEMNA are the best-performing attribution methods.

[7]The number of features selected for investigation varied by attack: from one single feature to as many as four.

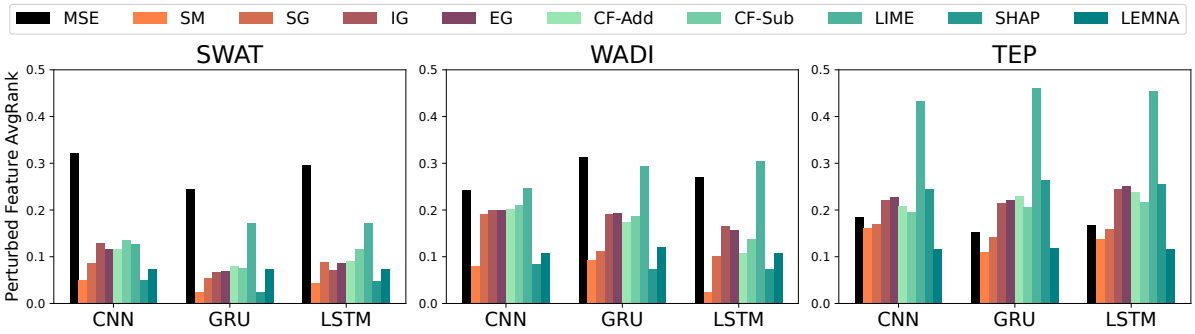


Figure 4.2: Results for all datasets (SWaT, WADI, TEP) and model architectures (CNN, GRU, LSTM) over our counterfactual benchmark; the AvgRank is shown (lower is better). Within white-box attribution methods (SM, SG, IG, EG), the saliency map (SM) always performs the best, and within black-box attribution methods (CF-Add, CF-Sub, LIME, SHAP, LEMNA), SHAP and LEMNA generally perform best. Notably, SM and LEMNA always outperform the raw-ranking of MSEs (MSE).

ICS anomaly detection is inherently noisy: benign features produce (small) errors and interactions between sensors and actuators can complicate the analysis of attribution methods. To remove these effects from our evaluation, we craft synthetic inputs to systematically evaluate an attribution method for a given anomaly-detection model. We evaluate attribution methods by computing counterfactual inputs; this ensures that a controlled manipulation introduced to a single input feature is the *only* source of error in an unsupervised anomaly-detection model.

First, we craft a zero-MSE, input-output pair by selecting an input window X^{base} from the benign training data, feeding it to an anomaly-detection model F, and storing the corresponding process-value prediction $Y^{base} = F(X^{base})$. We then perturb a feature j in X^{base} by two standard deviations to generate X^{pert}: when computing errors, we compare the prediction $F(X^{pert})$ with the synthetic ground-truth Y^{base}. To compute attributions, an attribution method uses X^{pert} and Y^{base} as the input window and ground-truth respectively. The perturbation is the only change introduced in the zero-MSE input-output pair, so a correct attribution would assign the perturbed feature j the highest score. We rank feature j in the attribution and repeat this measurement for all features, ultimately computing the AvgRank for each method.

We evaluate each attribution method for all nine combinations of model architecture (CNNs, GRUs, and LSTMs) and dataset (SWaT, WADI, TEP). Figure 4.2 shows the resulting AvgRank across all counterfactual inputs. Three attribution methods perform well: the saliency map (SM), SHAP, and LEMNA. These three methods outperform the MSE on all models and datasets, with the exception of SHAP on TEP (e.g., for SWAT CNNs, the MSE AvgRank is 16.4, whereas the SM, SHAP, and LEMNA AvgRanks are 2.6, 2.6 and 3.8 respectively). This suggests that attribution methods (black-box or white-box) can provide stronger insight than raw MSEs when attributing ICS anomalies.

White-box variants (SG, IG, and EG) outperform saliency maps on images [38, 118, 124], and our results suggest that the performance of these methods may not translate to ICS anomaly detection. We suggest two reasons why: First, the dynamics of ICS are more precise than in images; although adding random noise to images (as done in SmoothGrad) helps generalize attributions for images, randomness does not provide this benefit for ICS anomaly detection.

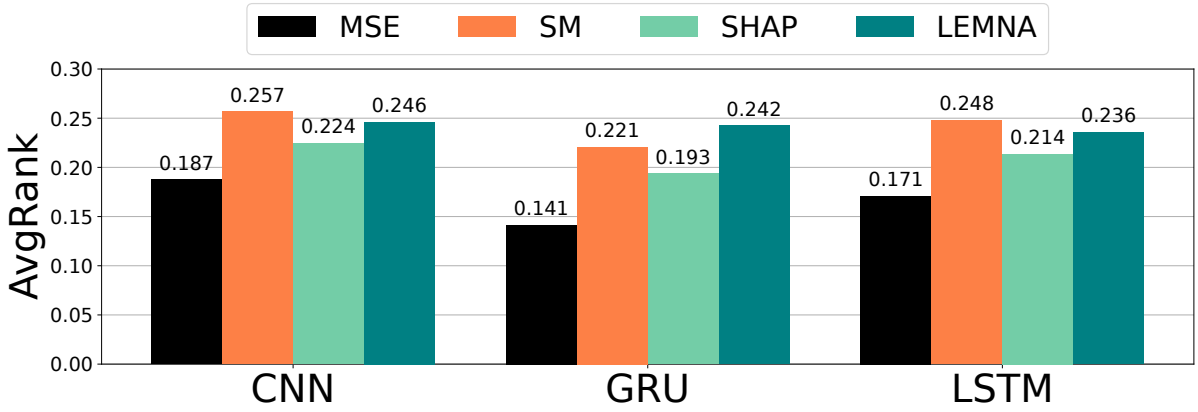


Figure 4.3: When attributing ICS anomalies at the time of detection, the raw-error feature (MSE) produces a lower AvgRank (lower is better) than all best-performing ML-based attribution methods: the saliency map (SM), SHAP, and LEMNA.

Second, benign ICS behavior cannot be well-represented with a single (or sample of) reference input(s), and thus choosing an effective baseline (required for IG and EG) is difficult.

When comparing black-box attribution methods, we find that SHAP and LEMNA outperform LIME. LIME uses a linear approximation: in contrast, SHAP (which uses Shapley values) and LEMNA (which uses a fused-lasso, Gaussian mixture model) can better capture the inter-feature dynamics of an ICS.

4.3.3 Evaluating ML-based attribution methods

Next, we compare raw-error ranking to our selection of ML-based attribution methods. For each detected attack, we find the first detection point using the same methodology as with raw-error rankings. We then find the corresponding model input for the detection point—using the range of data from the 50 timesteps prior to the detection point. The model input is used as input to an attribution method, producing a score for each ICS feature. We use these scores to rank features and compute AvgRank, repeating the process for each combination of deep-learning-based anomaly-detection model and attribution method.

Figure 4.3 shows the resulting AvgRank for all attribution methods and deep-learning anomaly-detection models. Although the results of our benchmark suggest that, in theory, ML-based attribution methods outperform raw-error ranking, ML-based attribution methods perform far worse when applied in practice to attack scenarios: for each model architecture, raw-error ranking produces a lower AvgRank than all ML-based attribution methods (0.14–0.19 for raw-error ranking, compared to over 0.2 for most ML-based attribution methods).

Finding 7: When computed at the timestep when the anomaly is detected, attributions based on ranking raw reconstruction errors from anomaly-detection models are less accurate than previously reported, and ML-based attribution methods have similar or worse accuracy.

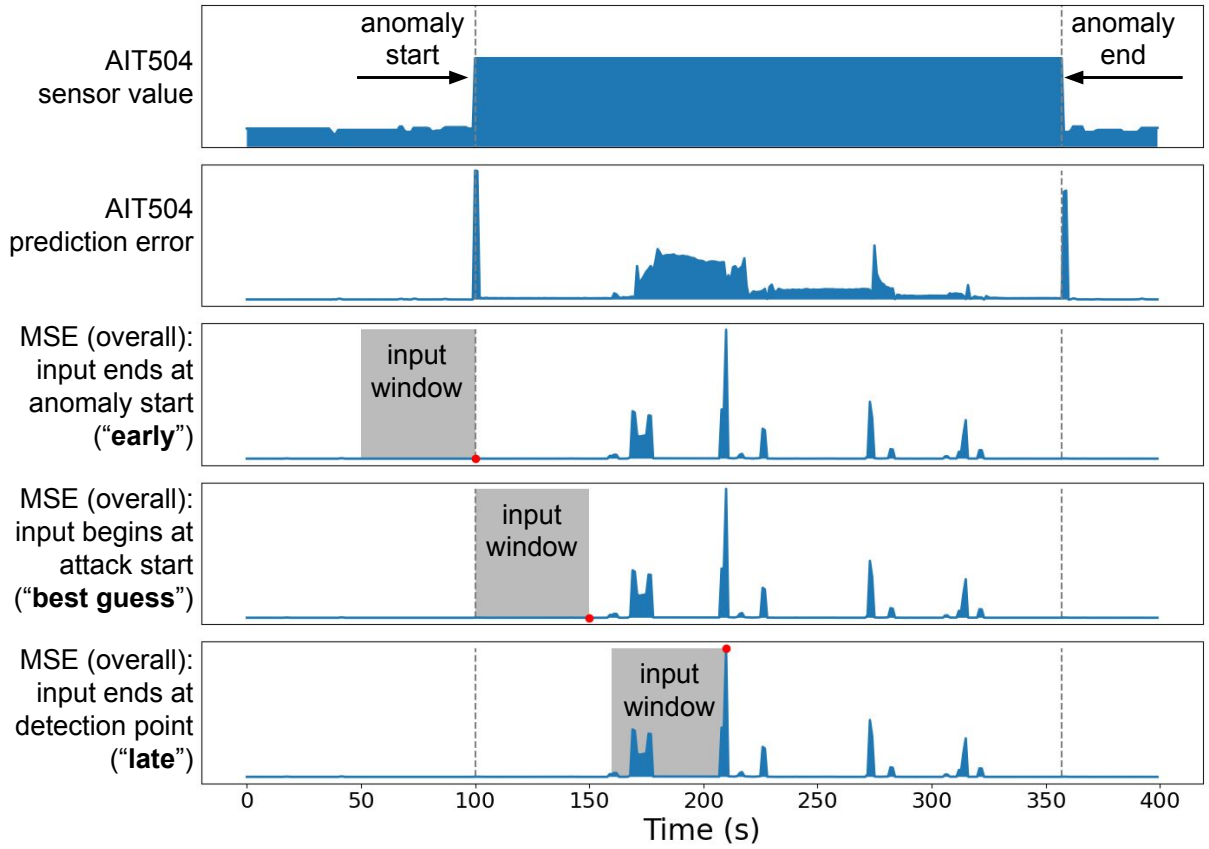


Figure 4.4: We show outputs of a GRU-based anomaly-detection model on SWaT attack #10: when sensor AIT504 is manipulated (top), its prediction error (2nd) is insufficient to trigger an anomaly. As the ICS responds and the total error increases, the model detects the anomaly over 100 seconds later. From this example, attributions can be computed at three points (shown in red) with corresponding input windows (shown in grey): at the anomaly start (3rd), when the input window coincides with the anomaly (4th), or when the anomaly is detected (5th, bottom).

4.4 Results: Factors that affect attribution accuracy

To better understand why attributions fail, we perform a more detailed analysis of ICS domain-specific characteristics: the timing of the attribution relative to the time the manipulation occurred (Section 4.4.1), properties of the manipulations (Section 4.4.2), and the stealthiness of the manipulations (Section 4.4.3). We find that different attribution methods perform best in different situations, and thus we propose an ensemble attribution method and find that it outperforms all individual methods (Section 4.4.4).

4.4.1 Effect of detection timing on attributions

ICS anomaly detection is performed over a time-series input, so selecting the best timing for attributions is an important consideration for optimal performance. The data in the selected input window may contain too much noise or too little signal to make an accurate attribution. We observe that there exists a “best-guess” timing: when ML-based attribution methods are

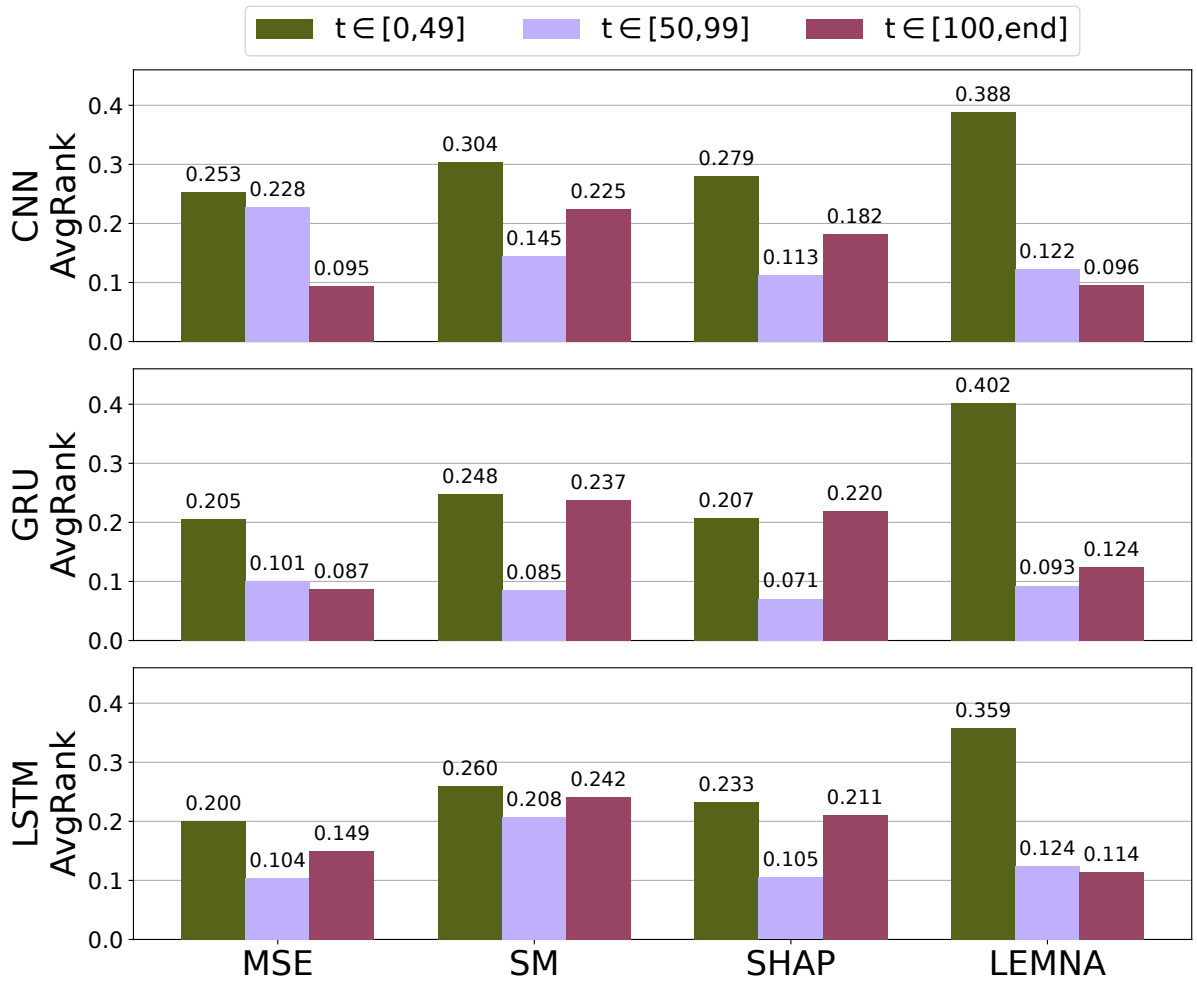


Figure 4.5: Across all detected attacks, we compare AvgRank (lower is better) across three timing cases, based on the detection time t relative to the start of the anomaly ($t = 0$). Considering that 50 timesteps are used for the model input, we divide attacks based on if $t \in [0, 49]$, $t \in [50, 99]$, or $t \in [100, \text{end}]$. For most cases, the AvgRank is lowest when $t \in [50, 99]$.

Table 4.4: We categorize each attack by its detection time t relative to the start of the anomaly (considered $t = 0$), dividing into cases where the detection is early ($t \in [0, 49]$), slightly late $t \in [50, 99]$, or very late ($t \in [100, \text{end}]$). For each model architecture, the number of attacks that fall into each case is shown.

	Total	$t \in [0, 49]$	$t \in [50, 99]$	$t \in [100, \text{end}]$
CNNs	103	52	10	41
GRUs	86	38	12	36
LSTMs	113	56	9	48

computed at this timing, their accuracy improves.

Example: How timing can affect attribution

To illustrate how timing can affect attribution, Figure 4.4 shows the sensor value, sensor prediction error, and the total MSE for a GRU-based anomaly-detection model for the duration of SWaT attack #10. In this attack, a chemical sensor's value (AIT504) is increased to 16, causing a reverse-osmosis sequence to shut down; the GRU model detects the anomaly within two minutes.

If the input window is selected immediately before the detection point ("late", shown in bottom row of Figure 4.4), the overall MSE is high: many features have drifted from their expected values in reaction to the attack and will appear anomalous, making it difficult to attribute the attack to the correct feature.

If the input window is selected immediately at the start of the attack ("early", shown in third row of Figure 4.4), observing such feature drift can be avoided; however, since our anomaly-detection models rely on historical input, the input window will contain benign signal, complicating attribution.

We observe that, in our dataset, the detection point can vary relative to the start of the anomaly. Given the model's input-window length (50 timesteps), the detection can occur before the window length has passed, far after multiple window lengths have passed, or at a time between these two cases—within one and two window lengths. Table 4.4 shows the number of occurrences for each of the described three cases: all three cases are prominent. We analyze AvgRank across these three cases and show the results in Figure 4.5. In most settings, attributions computed within 50 seconds of the anomaly start perform the worst, and attributions computed within 50 to 100 seconds of the anomaly start perform the best.

Finding 8: ICS anomalies vary in when they are detected relative to their start time. Differences in detection timing affect attribution accuracy.

Based on these observations, the ideal timing (for performing attribution) should be sufficiently near the start of the anomaly to avoid observing the original manipulation's subsequent effects; and should be sufficiently after the start of the anomaly, such that the input window contains sufficient manipulated information. Towards achieving these goals, we select an input window that starts at the same time as the anomaly, as shown in the fourth row of Figure 4.4 (e.g., given a 50-timestep input window length, we would use the 51st timestep after the anomaly start), and we call the strategy using this input window the "best-guess" timing[8].

Comparing timing strategies

Based on the observed differences in attribution accuracy across timing, we evaluate attribution methods across two timing strategies:

- "Practical" timing: when the input window immediately precedes the detection time (used in prior evaluation)
- "Best-guess" timing: when the input window starts at the same time as the anomaly

[8] "Best-guess" since in practice the start time is not known.

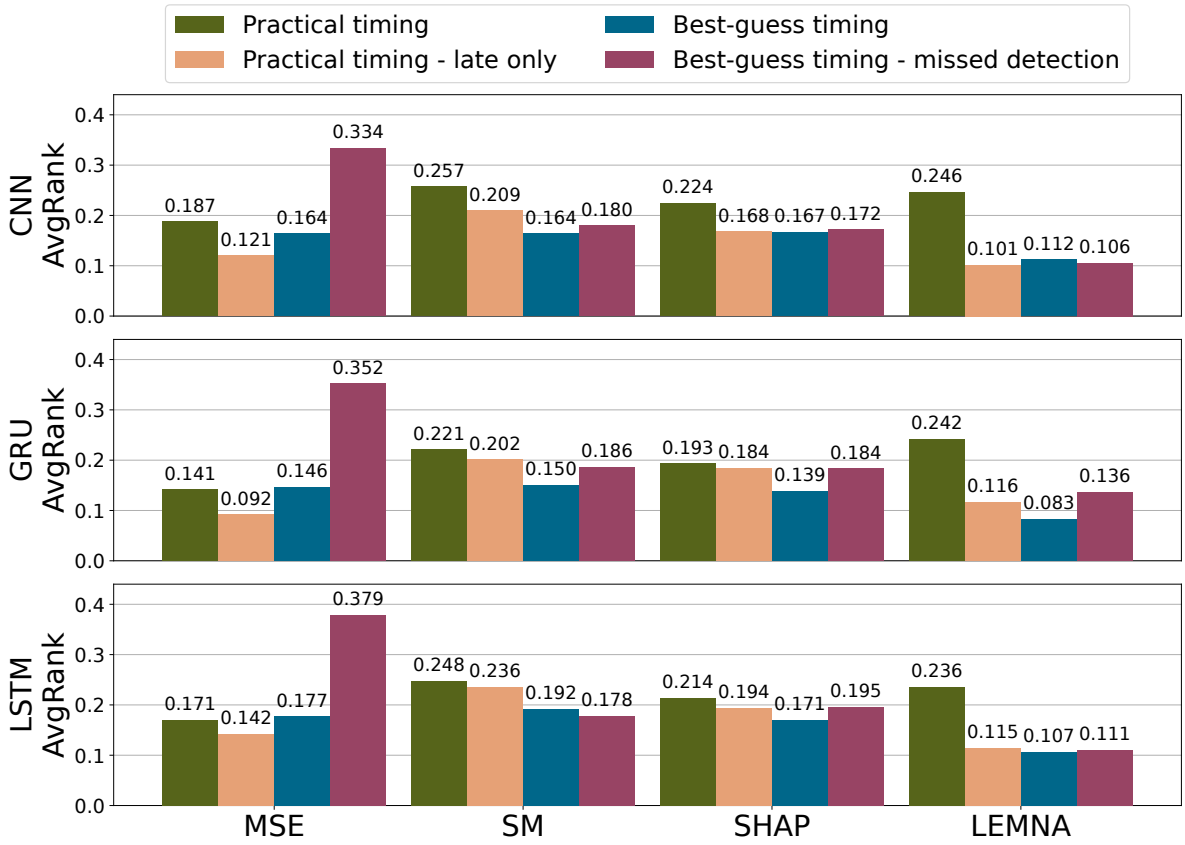


Figure 4.6: For all datasets, the AvgRank (lower is better) is reported after attributions are computed with different timing strategies: "practical timing", the prior attribution strategy that computes attributions immediately when anomalies are detected, and "best-guess timing", which computes attributions such that the input starts with the anomaly. Two additional variants are reported: practical timing with early detections removed, and best-guess timing for attacks that are not detected by the underlying anomaly-detection model. Results for the CNN (top), GRU (middle), and LSTM (bottom) are shown. In all cases, choosing an alternate timing strategy from the "practical" strategy improves attribution accuracy.

We compare the AvgRank at the best-guess timing to the practical timing, across all datasets (SWaT, WADI, and TEP) and all deep-learning-based model architectures (CNNs, GRUs, LSTMs). When evaluating the best-guess timing for each anomaly, the input is *exactly the same* across attribution methods and models, even if models detect the anomaly at different times.

Figure 4.6 shows the AvgRank for different timing strategies. We first compare the best-guess and practical timings ("best-guess timing" vs. "practical timing"): when an anomaly is detected, if instead the best-guess timing is used, does AvgRank improve? We find that in 10 out of 12 cases (including all cases with ML-based attribution methods), the best-guess timing outperforms the practical timing. For example, LEMNA improves for all models: the AvgRank drops from 0.246 to 0.112 for CNNs, from 0.242 to 0.083 for GRUs, and from 0.236 vs 0.107 for LSTMS. Furthermore, when the best-guess timing is used, LEMNA is the best-performing attribution method for all models.

Finding 9: ML-based attribution methods outperform raw-MSE rankings when attributions are computed with inputs beginning at the start of the anomaly.

To analyze the impact of early detections on practical timings, we compare the AvgRank after removing attacks that are detected before the 50th timestep, shown in Figure 4.6 ("practical timing" vs "practical timing—late only"). We discuss the results of the CNN; the results for the GRU and LSTM show similar observations. Although early detection of anomalies is clearly a beneficial outcome in practice, it results in worse attributions: removing the early detections improves the MSE AvgRank from 0.187 to 0.121. In all 12 cases, AvgRank improves after removing early detections.

Separating timing from detection outcome

Finally, we investigate if attribution methods could be useful even in cases where anomaly detection fails. Using the best-guess timing, we compare the AvgRank between anomalies that are detected and anomalies that are missed by the anomaly-detection model, shown in Figure 4.6 ("best-guess timing" vs "best-guess timing—missed detection").

For raw-MSE rankings, the performance is drastically worse for anomalies that are missed: the AvgRank increases from 0.164 to 0.334 for CNNs, from 0.146 to 0.352 for GRUs, and from 0.177 to 0.379 for LSTMs. This is expected: when the MSEs are insufficient to detect the anomaly, they are also insufficient to identify the manipulated feature. However, when these same inputs are used with ML-based attribution methods, the AvgRank performs far better for missed attacks. ML-based attribution methods perform approximately as well, regardless of whether the anomaly is detected or not. For example, the CNN LEMNA attribution AvgRank changes from 0.112 to 0.106. One potential implication of this observation is that attribution methods should be computed separately from detection times and detection outcomes; this could potentially be accomplished with a data historian or other post-hoc incident analytics for anomalies that are not detected in real time.

In summary, we found that the timing of attributions has a large impact on attribution accuracy. Although we do not advocate for a specific timing strategy, we would like to highlight that computing attributions at "practical" timings, the strategy most commonly used in prior work [58, 69], does not lead to best attribution outcomes, although it reflects how attribution methods might be used in practice.

4.4.2 Effect of attack properties on attributions

Although broad evaluations of attribution methods can reveal general trends, a deeper analysis across attack properties reveals imbalances in attribution accuracy between different attacks. In this section, we investigate how (i) the magnitude of the manipulation used in the attack and (ii) the type of feature attacked affect the accuracy of attribution methods. To enable such analysis, we define all anomalies from our three datasets (SWaT, WADI, and TEP) along common properties (as described in Section 4.2.1). We compute attributions at their best-guess anomaly timing (as described in Section 4.4.1) and perform statistical tests to quantify each dimension's

Table 4.5: We perform three statistical tests across our attribution results under "best-guess" timing: we compare the effect on AvgRank from manipulation magnitude (left), the type of feature attacked (middle), and whether the attack is multi-point (right). Results with p-value below 0.016 (applying Bonferroni correction) are bolded. Raw-error rankings (MSE) perform better on high-magnitude, sensor-based attacks, whereas attribution methods perform better on high-magnitude, actuator-based attacks. This analysis suggests that no attribution method is always optimal across a variety of attacks.

Attribution Method	Model	Magnitude (Pearson)		Sensor vs. actuator (continuous) vs. actuator (categorical) (ANOVA)			Single-point vs. multi-point (ANOVA)		
		Corr.	p-value	AvgRank	F(2, 153)	p-value	AvgRank	F(1, 154)	p-value
MSE	AR	-0.214	**p = 0.007**	0.227, 0.486, 0.244	9.02	**p < 0.001**	0.300, 0.232	1.18	p = 0.279
	CNN	-0.336	**p < 0.001**	0.162, 0.306, 0.330	14.87	**p < 0.001**	0.199, 0.304	4.60	p = 0.034
	GRU	-0.445	**p < 0.001**	0.154, 0.385, 0.361	17.95	**p < 0.001**	0.224, 0.288	1.75	p = 0.189
	LSTM	-0.399	**p < 0.001**	0.152, 0.362, 0.360	13.21	**p < 0.001**	0.224, 0.265	0.65	p = 0.421
SM	CNN	-0.430	**p < 0.001**	0.176, 0.059, 0.281	13.28	**p < 0.001**	0.160, 0.205	1.72	p = 0.192
	GRU	-0.605	**p < 0.001**	0.160, 0.050, 0.330	21.67	**p < 0.001**	0.156, 0.204	1.79	p = 0.183
	LSTM	-0.558	**p < 0.001**	0.197, 0.047, 0.329	19.55	**p < 0.001**	0.172, 0.245	3.71	p = 0.056
SHAP	CNN	-0.497	**p < 0.001**	0.176, 0.059, 0.275	13.89	**p < 0.001**	0.151, 0.231	5.86	p = 0.017
	GRU	-0.595	**p < 0.001**	0.149, 0.054, 0.325	24.31	**p < 0.001**	0.141, 0.225	6.47	p = 0.012
	LSTM	-0.513	**p < 0.001**	0.181, 0.039, 0.333	23.39	**p < 0.001**	0.159, 0.243	5.41	p = 0.021
LEMNA	CNN	-0.544	**p < 0.001**	0.085, 0.034, 0.291	26.82	**p < 0.001**	0.070, 0.252	39.06	**p < 0.001**
	GRU	-0.537	**p < 0.001**	0.084, 0.034, 0.278	26.76	**p < 0.001**	0.067, 0.252	46.80	**p < 0.001**
	LSTM	-0.523	**p < 0.001**	0.084, 0.034, 0.248	27.23	**p < 0.001**	0.070, 0.248	40.55	**p < 0.001**

effect on AvgRank. The results of this analysis are shown in Table 4.5.

Effect of magnitude

We compare the effect of manipulation magnitude on AvgRank. Since the range of observed magnitudes in our dataset is large (0.06–91 standard deviations), we take the natural log of the magnitude for our analysis. The first column of Table 4.5 shows, across all attacks, (i) the Pearson correlation coefficient between the log-scaled manipulation magnitude and AvgRank and (ii) the resulting p-value of the non-correlation test with Student's t-distribution.

For all methods, we observe a statistically significant relationship between manipulation magnitude and AvgRank. Since the anomaly-detection methods studied in this work rely on statistical modelling, lower-magnitude manipulations are more difficult to attribute. High-magnitude manipulations produce more immediate and obvious dispersions [49], so attribution methods that perform well on these manipulations may not be needed. An important area of future work would be to design effective attribution methods that are robust to low-magnitude manipulations.

Effect of feature type

Sensors and actuators are fundamentally different: actuators induce changes in the industrial process, while sensors provide feedback from the industrial process. Some actuators are also encoded as categorical variables (e.g., a valve in SWaT is ON (1) or OFF (0)), while all sensors in our datasets are continuous-valued. Although sensors and actuators differ in context and representation, current statistical and deep-learning-based anomaly-detection models treat these features equally as raw-valued features.

We analyze if whether a sensor or actuator was manipulated affects AvgRank. Raw-error rankings perform significantly better for sensor-based attacks while ML-based attribution methods perform better on actuator-based attacks. We empirically explore this difference by using a one-way ANOVA test to compare the AvgRank distributions when separating manipulations on sensors, manipulations on categorical actuators, and manipulations on continuous actuators. The results of this test are provided in the second column of Table 4.5.

For raw-error ranking (MSE), the AvgRank for sensors ranges from 0.152 to 0.162, whereas the AvgRank for actuators is always above 0.306. For ML-based attribution methods, the findings are different: attribution methods perform best on continuous-valued actuators (AvgRank below 0.060), while still performing well on sensors (AvgRank below 0.198). This suggests that attribution methods may be able to capture relationships that connect sensors with their corresponding actuators, beyond what can be found by the raw-error ranking. We also find that attribution methods perform worst on categorical-valued actuators (AvgRank above 0.248), likely because attribution methods compute attributions for categorical-valued features as if they were continuous-valued features.

In general, anomaly-detection models would likely benefit from modelling sensors and actuators differently when computing attributions, in ways that consider (i) interdependencies between sensors and actuators and (ii) categorical variables as states, rather than continuous values. Attacker models for sensors and actuators are also different; prior work has argued that actuator attacks require more ICS knowledge and are more difficult to execute in practice [133].

4.4.3 Evaluating against stealthier manipulations

Although attackers can adjust modify manipulation properties to reduce attribution accuracy, stealthier manipulations strategies can be even more effective. In this section, we explore how attribution methods are affected by stealthier manipulation strategies, extending beyond strategies used for current datasets. We find that: (i) multi-point attacks are more difficult to attribute and (ii) summing and linear manipulations are particularly effective at reducing attribution accuracy.

Multi-point attacks

We first consider multi-point attacks: when multiple features are manipulated simultaneously. Multi-point attacks are included in the SWaT and WADI datasets. In general, correctly attributing multi-point attacks is more difficult, since the effects of multiple manipulations are observed in the ICS. We use a one-way ANOVA test to compare the AvgRank distributions for single-point attacks and multi-point attacks; results are in the third column of Table 4.5.

We find that LEMNA is significantly more accurate for single-point attacks; the AvgRank is over three times higher when a multi-point attack is performed. In general, all attribution methods (except raw AR-score ranking) are less accurate on multi-point attacks.

Table 4.6: By introducing summing or linear manipulations, attackers can reduce attribution method accuracy. When comparing summing or linear manipulations to their constant-valued counterparts: (i) the manipulation is detected later and (ii) the AvgRank for all attribution methods increases.

		Constant	Summing	Linear
CNNs	Detection latency	200s	694s	1232s
	MSE AvgRank	0.075	0.140	0.147
	SM AvgRank	0.287	0.536	0.619
	SHAP AvgRank	0.223	0.415	0.362
	LEMNA AvgRank	0.117	0.423	0.551
GRUs	Detection latency	242s	1132s	1316s
	MSE AvgRank	0.064	0.102	0.072
	SM AvgRank	0.279	0.513	0.525
	SHAP AvgRank	0.204	0.472	0.366
	LEMNA AvgRank	0.087	0.389	0.468
LSTMs	Detection latency	174s	571s	1090s
	MSE AvgRank	0.087	0.151	0.113
	SM AvgRank	0.355	0.551	0.574
	SHAP AvgRank	0.174	0.377	0.464
	LEMNA AvgRank	0.072	0.343	0.634

Summing and linear manipulations

We implement two alternate manipulation types with the TEP simulator: linear and summing manipulations[9]. These manipulations achieve the same sensor value as the constant-valued manipulations prevalent in SWaT, WADI, TEP, and prior work [23, 72], yet are more stealthy (i.e., harder to detect and attribute correctly).

A linear manipulation incrementally increases in magnitude with each timestep, and a constant-sum manipulation adds a constant value to the original sensor value at each timestep, which maintains the natural amount of noise. We define the stealthier manipulation types based on the original attack model used in Section 4.2.1. For a summing manipulation:

$$x'_j(t) = x_j(t) + c$$

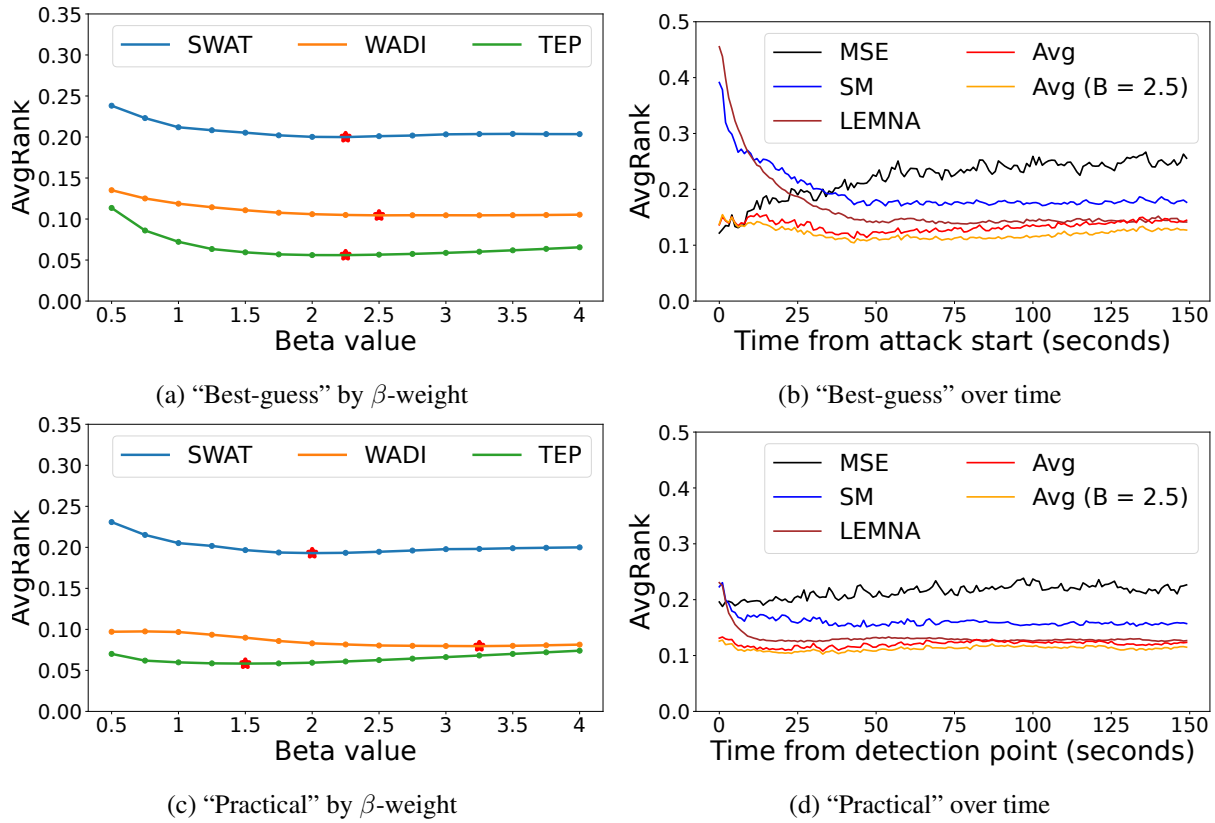
For a linear manipulation, where t_a is the initial point of the attack, and m is the slope:

$$x'_j(t) = m(t - t_a) + x_j(t_a)$$

We use the modified TEP simulator to perform the stealthier manipulations on every sensor in the system. For each sensor, we perform a two-standard-deviation-magnitude manipulation with the stealthier manipulation types. We compare the AvgRank and detection latency for the five cases where, regardless of manipulation type, all attacks are detected by all three models.

Table 4.6 shows the resulting AvgRank and detection latency for constant, constant-sum, and linear manipulations. On average, performing an attack with a stealthier manipulation causes the

[9]Included in our public set of 286 manipulations.



(a) "Best-guess" by β-weight

(b) "Best-guess" over time

(c) "Practical" by β-weight

(d) "Practical" over time

Figure 4.7: Attribution averaging is evaluated on CNNs at two timings ("best-guess" on top, "practical" on bottom) and in two ways. On left, the AvgRank (lower is better) of attribution methods is reported over 150 timesteps. Regardless of timing strategy and the selected timestep, an average of attribution methods outperforms any individual method. On right, across datasets, different β values are used when performing a weighted average, across all six settings, $\beta \in (1.5, 3.25)$ are optimal (red star).

attack to be detected later: compared to constant manipulations, constant-sum manipulations are detected at least three times later and linear manipulations are detected at least five times later. In addition, the attributions computed at these detection points are less accurate: the AvgRank increases in all cases. When using alternate manipulation types, attribution accuracy decreases while the same target sensor value is achieved.

4.4.4 Evaluating ensembles of attribution methods

Different attribution methods are best in different scenarios: ML-based attribution methods work best for continuous-valued actuators and at best-guess timings, whereas raw-error ranking works best for sensors and at practical timings (Sections 4.4.1–4.4.2). Thus, in this section, we design an ensemble of attribution methods to combine the strengths of both types of attribution methods, by using a weighted average over attributions from multiple methods. We find that this ensemble outperforms all individual attribution methods.

Our ensemble of attribution methods computes attributions differently for sensors and actuators; the ensemble uses a raw average of attributions for sensors and a β-average of attributions

for actuators, where β represents the relative weight of ML-based attribution methods. We compute our ensemble over three attributions: the normalized MSE, the normalized SM attribution, and the normalized LEMNA attribution.

$$s_{AVG_\beta} = \begin{cases} s_{MSE} + s_{SM} + s_{LEMNA} & \text{if sensor} \\ s_{MSE} + \beta(s_{SM}) + \beta(s_{LEMNA}) & \text{if actuator} \end{cases}$$

We compare our ensemble of attribution methods to the raw average of attributions: the left half of Figure 4.7 shows the AvgRank for our ensemble with different β-values; we compare averaging strategies for CNNs using the best-guess (Figure 4.7a) and practical (Figure 4.7c) timing strategies (described in Section 4.4.1). For all datasets and timing strategies, the best-performing value of $\beta \in [1.5, 3.25]$ (i.e., higher weight for ML-based attribution methods for actuators) outperforms the raw average (i.e., when $\beta = 1$). We also compare our ensemble to the raw average with GRUs and LSTMs; in all cases the best-performing $\beta \geq 1$, which shows that our ensemble outperforms the raw average.

We next compare our ensemble attribution method to individual, best-performing attribution methods: the raw-error ranking (MSE), SM, and LEMNA. Figure 4.7b shows the AvgRank for various attribution methods (individual and ensemble) for the CNN model over 150 timesteps, beginning with the start of the anomaly. Similar to what was found in Section 4.4.1, we find that ML-based attribution methods are initially less accurate, but their accuracy improves over time; conversely, a ranking of raw errors is initially accurate but becomes less accurate over time. Our ensemble attribution method combines strengths of each individual method and produces the lowest AvgRank over most timesteps. Our ensemble attribution method also outperforms all individual attribution methods when used with a practical timing strategy (Figure 4.7d), showing that it can also be used in practice when ground-truth timing is not known. This finding holds when our evaluation is repeated with GRUs and LSTMs.

Finally, we determine what the best-performing configuration for our ensemble attribution method is in practice: Figure 4.7d shows that when using our ensemble attribution method with $\beta = 2.5$, and attribution are computed between 25 and 50 seconds after the anomaly is detected, the AvgRank is lowest. Thus, we observe this configuration of our ensemble attribution method to be the best-performing attribution method across all attribution methods evaluated in this work.

Finding 10: An ensemble of attribution methods outperforms all individual attribution methods at identifying which feature was manipulated in an ICS anomaly.

4.5 Survey: ICS operator perceptions of attributions

Our experiments showed that attribution methods do not achieve perfect accuracy: the feature with the highest attribution score was actually the manipulated feature for less than 39% of all attacks (see Section 4.3.1). This raises the question: are imperfect attributions for anomaly detectors useful to ICS operators?

In this section, we describe the methodology and results of a preliminary survey of ICS operators, which was conducted concurrently with our experiments to better understand whether and

how attributions would be helpful for responding to anomalies. We sought operators' perspectives on the following questions:

1. How do ICS operators respond to anomaly-detection alerts and how would attributions fit into their workflow?
2. Assuming imperfect attribution performance, would operators find it more useful to be shown fewer features, but with a higher chance of omitting the feature that caused the anomaly; or more features, with a lower chance that the manipulated feature will be missed?

Survey methodology

First, we asked participants to describe the type of ICS they have experience with and their role in operating ICS. Next, to surface how operators integrate anomaly detectors into their workflows, we asked participants what kinds of anomaly detectors they have experience with, the benefits and challenges of using them, and what their role is in diagnosing the root causes of an anomaly.

Then, we asked participants to evaluate the tradeoff between the number of reported features and attribution accuracy. We showed participants a sample output for an attack in the SWaT dataset: a subset of the sensors and actuators, their values, and their attribution scores (in descending order). We then asked the participants to rate on a five-point Likert scale how useful the output would be in an attack scenario. We varied the number of features shown in the output, between the top two, five, ten, 20, or all 34 features[10]. We also varied the error rate (the percentage of attacks where the attacked feature is not included in the subset of features shown in the output) between a "low", "medium", and "high" level, which changes depending on the number of features shown (See Table 4.8). The error rates were based on our initial estimates of attribution accuracy; later we observed that the "high" error rate roughly matches the empirical error rate of raw-error rankings for the LSTM-based detection model using practical timings on SWaT, and the "medium" error rate roughly matches the empirical error rate of the best-performing LSTM-based ensemble attributions on SWaT. Lastly, we asked participants to explain the reasoning behind their ratings, and how they would integrate attributions into their workflow. The full survey text is available in Appendix A.

Participant recruitment and ethics

We recruited participants by sending email flyers to employees at organizations that run ICS and by sending private messages via LinkedIn to people with job titles relating to ICS security (e.g., OT Security Architect). We recruited seven participants in total. Participants were compensated with $5 gift cards. Our survey was approved under Exempt Review by our institutional review board.

[10]This survey was performed with an earlier version of SWaT with a feature selection step, and thus only 34 features were used.

Table 4.7: List of survey participants: their participant code, the type of ICS they operate, and their role at their organization.

	ICS Type	Role/Title
P1	Distributed control system	Cybersecurity, design and acquisition
P2	(Not disclosed)	Security engineer
P3	Unmanned vehicle ground control	Operator supervisor
P4	Various	Consulting and Research
P5	Electrical power generation and transmission	SCADA Engineering, System Integration and Security
P6	Electric Transmission	Security Engineer
P7	Various (manufacturing and distribution)	CISO

Table 4.8: Participants' ratings for the usefulness of hypothetical attribution outputs, varying the number of features shown and the error rate (1 being "not at all useful" and 5 being "extremely useful"). "Error rate" is the likelihood that the manipulated feature is not in the output. Bolded values indicate that the average rating was "moderately useful" or above.

# Output Features	**Usefulness (1-5)**		
Error rate: (H, M, L)	**(High)**	**(Medium)**	**(Low)**
2 (70%, 40%, 20%)	1.29	2.00	2.86
5 (50%, 30%, 10%)	1.57	2.57	**4.14**
10 (40%, 20%, 5%)	1.57	2.86	**4.29**
20 (30%, 10%, 5%)	2.14	**3.14**	**4.43**
All 34 (0%)			2.43

Survey results

Participants worked on a variety of ICS, ranging from electric transmission to controls for unmanned vehicles (see Table 4.7 for a summary of their roles and types of ICS operated). All had experience with rule-based anomaly detection, and four reported using ML-based anomaly detection.

Participants described anomaly detection as the first step in the attack mitigation process, followed by manual investigation and correction (P1, P2, P4, P6). Some challenges reported by participants included (i) excessive false positives that led to "wild goose chases", requiring manual mitigation by operators (P1, P2, P5), and (ii) a lack of data and context in alerts raised by detectors, which made it difficult to trace root causes of anomalies (P4, P6). Prior work has found that, more broadly, SOC analysts have similar challenges with security alerts [9].

Participants provided several examples of how attributions would be integrated into their workflow. Attributions could help provide context on the relationship between system components (P3, P4) and inform follow-up diagnostic steps, such as running tests and consulting runbooks (P2, P4). Attributions could also be integrated with other data sources such as control system logs (e.g., SCADA) in a security information and event management (SIEM) system (P5, P7).

Participants perceived attribution outputs with more features and low error rates to be more useful (Table 4.8). Participants reported attributions in the low-error-rate condition to be "very useful" (average score 4.14–4.43), if 20, ten, or five features were shown. For the medium-error-rate condition, roughly corresponding to our ensemble model's performance, participants reported attributions to be "moderately useful" if 20 features were shown (3.14) and "slightly useful" (2.00–2.86) for fewer features. For the high-error-rate condition, roughly corresponding to raw-error ranking, participants reported attributions to be between "slightly useful" and "not at all useful" (1.29–2.14). Lastly, attributions showing all 34 features were reported as only "slightly useful" (2.43) by participants.

Participants mentioned several considerations regarding attribution accuracy and quantity of information. P1 and P3 generally preferred having more information available to the operator. P7 preferred seeing less information, as having too many false positives would require too much effort to investigate. P5 and P6 said that operators and organizations would be unlikely to trust models with high error rates. P2 and P4 explicitly weighed the tradeoff between error rates and the amount of information shown:

> *A balanced trade-off is needed. Often having [a] list of max 10 [sensors] with minimal error rate is more useful than having less with high error rate. Depends on the needed follow-up testing effort to identify the one culprit finally. –P4*

These results suggest that attributions could help ICS operators respond to anomalies, even without perfect accuracy. Attributions could provide a starting point for operators when investigating anomalies; for this use case, operators reported that an attribution method that performs as well as our proposed ensemble method would be moderately helpful, and preferred to see attribution scores for the top 10–20 features to balance accuracy and the amount of information shown.

Finding 11: ICS operators are likely to find our current best-performing attribution methods for anomaly detection models to be moderately useful when responding to incidents, even if the single manipulated sensor or actuator cannot be identified with high accuracy.

4.6 Discussion and recommendations

In this section, based on our findings, we provide recommendations for researchers and practitioners to use attribution methods for ICS anomalies.

4.6.1 Recommendations for researchers

Evaluate attributions on a diverse set of complex ICS attacks. The accuracy of attribution methods depend heavily on ICS attack properties (Section 4.4.2). Despite the wide range of potential attack strategies, public ICS datasets predominantly contain high-magnitude, constant-valued manipulations [7, 48], and prior work evaluates attributions on only a small number of

attacks from these datasets [58, 69]. When developing ICS anomaly detection and attribution methods, evaluations should be performed on a complex and diverse set of ICS attacks. This would ensure that attributions generalize across attack strategies and perform well on attacks that are most difficult to attribute with currently existing methods (low-magnitude, categorical-actuator-based attacks).

Design attribution methods specifically for ICS anomaly detection. When tested on full ICS datasets, we found that prior attribution strategies performed less well than previously suggested, and that our adapted ML-based attribution methods performed less well than anticipated (Section 4.3.1). Attributing ICS anomaly detection presents unique challenges: attributions are time-dependent (Section 4.4.1) and affected by additional feature dependencies (Section 4.4.2). Furthermore, the results of our survey (Section 4.5) suggest that ICS operators would prefer attributions to show a list of 10–20 features to provide context for their investigation, rather than just the top few features.

Future work that designs attribution methods for ICS should be designed to directly address the aspects of ICS anomalies that make their attributions uniquely challenging, such as considering separate designs for sensors and actuators. These methods should also be evaluated with operators' preferences and workflows in mind, rather than optimizing solely for top feature accuracy.

4.6.2 Recommendations for practitioners

Consider attributions in workflows beyond the real-time detection case. Although it may seem most intuitive to compute attributions in real time at the moment when anomalies are detected, this strategy is suboptimal for attribution accuracy.

We found that that attributions are most effective when computed with an input that closely follows the start of the anomaly (e.g., 25 seconds): this is when the input to the attribution method includes some data on how the ICS has responded to the initial manipulation, but before the input is dominated by the manipulation's side effects (which may increase with time). However, in many attacks, the anomaly detector does not generate alerts at this ideal point in time.

Furthermore, we found that ML-based attribution methods can identify manipulated features, even when the input contains insufficient information for the anomaly-detection model to generate an alert (Section 4.4.1). Hence, to overcome the limitations of anomaly detection in providing timely detection for optimal attribution, we suggest using attribution methods in post-hoc settings, using tools like a data historian, which would allow the operator to leverage their domain expertise to explore optimal timings for attributions.

Use an ensemble of attribution methods. The optimal choice of attribution method differs based on the ICS, anomaly-detection model, and properties of the attack being attributed. A "silver bullet" solution does not yet exist for attributions of ICS anomalies. We found that, on average, a weighted ensemble of attributions from raw reconstruction error, saliency maps, and LEMNA outperforms all individual attribution methods (Section 4.4.4).

4.7 Summary

In this chapter, we evaluate how effective prior and newly adapted attribution methods are when used to identify the manipulated feature in an ICS attack. We performed the first broad evaluation of ICS anomaly attribution, comparing across anomaly-detection methods, model architectures, and datasets. We found that ICS anomaly attribution is dependent on several factors. We examine these factors and identify challenges related to the timing of anomaly detection and differences in feature types. We ultimately develop a strategy that uses an ensemble of attribution methods and show that it outperforms all individual attribution methods.

Chapter 5

CYPRESS: a structurally sparse model for ICS anomaly detection

In Chapter 3 and Chapter 4, I describe work that evaluates general-purpose model architectures (e.g., CNNs, LSTMs) for ICS anomaly detection and attribution. These approaches are general applications of models that maintain a uniform internal structure for input that correspond to different ICS components.

However, in reality, the dynamics of an ICS are not uniform. ICS components contain an underlying connected structure, based on physical and logical relationships. We hypothesize that detection and attribution of ICS anomalies can be improved with approaches that learn representations that better represent these relationships found in ICS. In this chapter, I describe work that investigates how spatial and logical relationships found in ICS can be embedded as structured representations in machine-learning models. These structured representations limit the relationships that can be learned from data when training a machine-learning model, and makes anomaly-detection models more efficient and effective for anomaly detection and attribution.

This work described in this chapter is in submission to the *35th USENIX Security Symposium* (USENIX Security 2026).

5.1 Introduction

In Chapter 3 and Chapter 4, we evaluated ML-based anomaly-detection models for detection and attribution respectively. Many of the best performing approaches were based on deep learning, which trains models with millions of parameters based on patterns found in ICS process data [98, 148]. However, these approaches are inefficient, are difficult to explain, and are susceptible to evasion attacks. The work described in Chapter 4 found that raw-error rankings were inaccurate for anomaly attribution; in over 60% of attacks, the feature with the highest error did not correspond to the ICS component that was manipulated.

Furthermore, factors such as interpretability, robustness, and efficiency are strong requirements for adopting anomaly-detection approaches [89]. Thus, deep-learning-based approaches may not be suitable for adoption in many ICS settings because of constraints on computational resources and skilled personnel [18, 106].

To overcome these problems, we propose and evaluate CYPRESS, a novel model architecture that learns representations of ICS defined by a structurally sparse set of inter-feature relationships. Rather than learning relationships from an unconstrained set of possible connections (as is done for most deep-learning models), CYPRESS learns structurally sparse representations that limit relationships between input features. CYPRESS requires only light-weight guidance (i.e., groupings of components) to learn representations of ICS from time-series process data.

Unlike deep learning models, CYPRESS uses far fewer model weights and only learns model weights that correspond to specified relationships between ICS components. We evaluate the detection performance, attribution performance, and robustness of CYPRESS across attacks from three ICS datasets, and we show that CYPRESS is more efficient, explainable, and robust than deep-learning-based approaches for ICS anomaly detection.

5.2 Model architectures for anomaly detection

In this section, we describe model architectures used in prior work for anomaly detection. In Section 5.2.1, we describe data description models, which are structurally sparse anomaly detection models used in the image domain that serve as a design inspiration for CYPRESS. In Section 5.2.2, we describe relevant model architectures used for ICS anomaly detection and how CYPRESS differs from these architectures in its internal representations of ICS.

5.2.1 Data description models

In the image domain, a data description model directly predicts scores for every input feature (i.e., pixels), which can then be used for anomaly detection [82, 103]. Data description models are structurally sparse—only a limited and fixed set of inputs can affect each output. Deep support-vector data description (Deep SVDD [103]) learns a sparse, compact hypersphere in its output layer that computes scores for input features, and fully convolutional data description (FCDD [82]) learns sparse convolutional kernels that compute scores for input image pixels.

Table 5.1: ML model architectures used in prior work for ICS anomaly detection, and how they represent (i) time relationships and (ii) sensor and actuator relationships.

	Time relationships	**Sensor and actuator relationships**
DNN [3]	Fully connected	Fully connected
CNN [70] GRU [77] LSTM [98]	Constrained by pre-defined structure	Fully connected
GDN [29] FSN [54]	Constrained by pre-defined structure	Constrained by weight pruning
CYPRESS (ours)	Constrained by pre-defined structure	Constrained by pre-defined structure

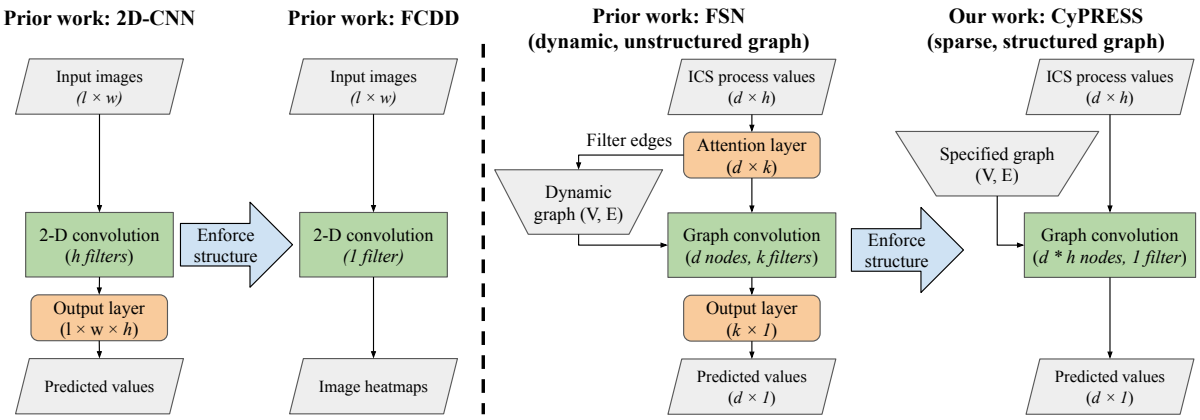


Figure 5.1: A comparison of the internal structures of CNNs, FCDD, FSNs, and CYPRESS (from left to right). In the image domain, FCDD is a structured alterative to CNNs and only learns relationships between neighboring pixels end-to-end. Similarly, in the ICS domain, CYPRESS is a structured alternative to FSNs that relies only on graph convolutions, ensuring that only a structured set of inter-feature relationships are learned.

SVDD and FCDD constrain connections between input features and output scores to those within a structurally sparse set. For example, when FCDD predicts scores for each pixel that describes how that pixel contributes to an anomaly, each output pixel's score is only influenced by its neighboring input pixel values.

We propose that data description models contain properties that are desirable for ICS anomaly detection, and so we design CYPRESS with similar assumptions. We assume that each predicted process value should only be influenced by a limited set of inputs from contextually relevant process values. Unlike deep-learning models, which learn weights for dense, fully connected representations, CYPRESS learns structurally sparse representations that correspond to the physical and logical relationships present in ICS.

5.2.2 Models used for ICS anomaly detection

In this section, we describe ML models used in prior work for ICS anomaly detection, and we compare how these models internally represent inter-feature relationships learned from ICS process data. Within the two-dimensional, d-by-h input of these ML models, there are two types of relationships between features that can be learned: *component relationships* (i.e., connections between sensors and actuators in d dimension) and *time relationships* (i.e., connections between timesteps in h dimension). In Table 5.1, we compare the ways in which these models used in prior work represent relationships between ICS features.

Neural networks. A fully connected deep neural network (DNN) connects all input features to all internal feature representations and learns a full set of weights for each of these connections [3]. Thus, the model can learn any relationship between any input features, leading to spurious correlations.

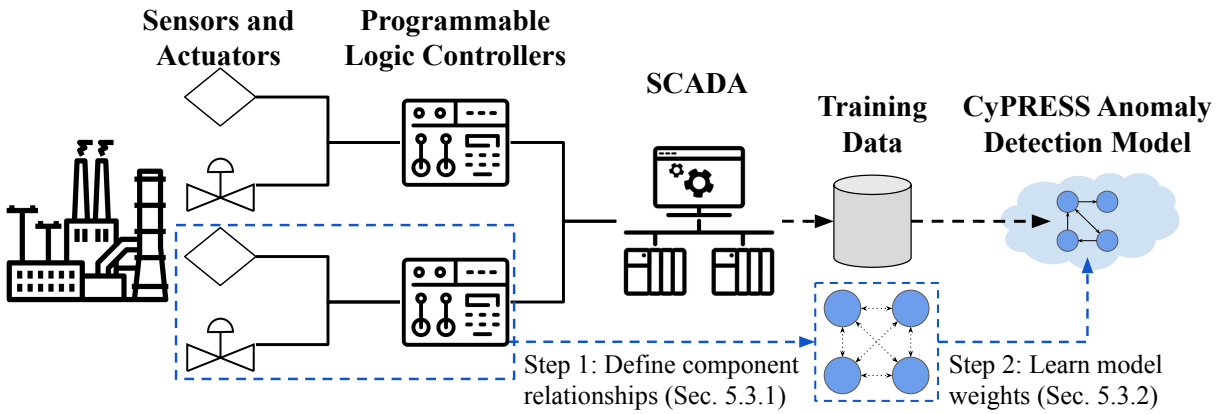


Figure 5.2: An illustration of the graph specification and training processes for CYPRESS. First, in step 1 we define a set of component relationships from the ICS, such as all components that share a PLC, and use this set to specify a fully connected, time-aware graph (described in Section 5.3.1). Then, in step 2 we use this specified graph to train CYPRESS, which learns the strengths of these connections from process-level training data (described in Section 5.3.2).

1-D time constraints. Much like their traditional 2-dimensional image-based counterparts [71], 1-dimensional convolutional neural networks (CNNs) use convolutional kernels to constrain relationships learned along the h dimension; by doing so, CNNs only learn relationships between neighboring timesteps [70]. Models based on gated recurrent units (GRUs) and long-short-term-memory units (LSTMs) enforce similar constraints along the h-dimension [77, 148]. Although these models limit which time-series relationships are learned, they are unconstrained in the d dimension, which means that they can learn any relationship between sensors and actuators. In Section 5.4, we empirically validate the presence of spurious relationships between components in these models.

Graph neural networks. Graph decision networks (GDNs [29]) and fused sparse autoencoder graph networks (FSNs [54]) are sparse models used in prior work for ICS anomaly detection. GDNs and FSNs train an attention layer and use its top k weights for inference, which constrains component relationships in an unstructured way [29, 54]. However, these attention layers are fully connected, so although only a sparse set of component relationships is used for inference, the set of potential component relationships that can be learned is unconstrained. The only way to enforce sparsity into GDNs and FSNs is by explicitly specifying the value of the parameter k, which defines the number of connections (i.e., inter-feature relationships) that can be learned for each feature. In Section 5.4, we show that, despite being sparse, GDNs and FSNs are still susceptible to learning spurious relationships.

5.3 CYPRESS: Cyber-Physical Representations with Sparse Structures

In this section, we describe a new model architecture that is sufficiently sparse to avoid learning spurious correlations but with enough capability to learn the feature relationships essential for detecting ICS anomalies. We call our model architecture CYPRESS (Cyber-Physical REpresentations with Sparse Structures). In contrast to other models used in prior work, CYPRESS enforces *structured* constraints along both the time dimension and the component dimension (shown in Table 5.1) with graph convolution layers. Figure 5.1 shows a comparison of CYPRESS with other models. We note in particular that CYPRESS does not use any fully connected layers, meaning that all of its learned representations are structured by graph convolutions.

Defining a CYPRESS model requires a specified set of relationships between input features. In Section 5.3.1, we describe strategies for manually specifying these relationships or automatically generating them from process data. In Section 5.3.2, we then describe our method for training CYPRESS, which learns structurally sparse representations of ICS based on these defined relationships. Figure 5.2 shows an overview of these processes.

5.3.1 Specifying inter-feature relationships

Much like how FCDD uses convolutional layers to represent relationships between input pixels, CYPRESS uses graph convolution layers to represent relationships between process-level input features along the time and component dimensions. These graph convolution layers are defined by a set of inter-feature relationships, and in this section we describe how such inter-feature relationships are specified.

Component relationships. We first describe how the relationships between components (i.e., d sensors and actuators) are represented in CYPRESS. Similarly to prior work [29, 54], we represent each component as a node in a connected graph, and we represent relationships between components as directed edges between these nodes; these relationships can be physical (e.g., the flow sensor of a pipe and the level sensor of the tank that the pipe flows into) or logical (e.g., a sensor value that is used by a PLC to change an actuator). Component relationships can be manually specified or automatically generated; we suggest several methods for defining them in Section 5.5.1 and evaluate them in Sections 5.6.1–5.6.2. Figure 5.2 illustrates how these component relationships could be specified by defining edges between all sensors and actuators that share a common PLC.

Time relationships. Once we create a graph that represents the relationships between components, we translate this graph into one that includes time relationships to account for CYPRESS's input history h (i.e., we convert a graph of d nodes into one with $d \times h$ nodes). We translate each component relationship $d_i \rightarrow d_j$ to a temporal relationship $d_{i,t} \rightarrow d_{j,t+1}$ for h timesteps. For example, translating a component relationship across 10 timesteps creates nine distinct edges in the time-relationship graph (e.g., $[d_i \rightarrow d_j]$ to $[d_{i,0} \rightarrow d_{j,1}, d_{i,1} \rightarrow d_{j,2}, ..., d_{i,8} \rightarrow d_{j,9}]$. Once all component relationships are translated into time relationships, we use the $d \times h$ graph to define

graph convolution layers in CYPRESS, which learns weights *only* for the nodes and edges in this graph.

5.3.2 Learning weights in CYPRESS

For the $d \times h$ graph, we define a node weight v_i for each graph node, and we define an edge weight $w_{i,j}$ for each edge that defines the relationship strength between a pair of graph nodes (i, j). These weights are used during inference in a message passing algorithm often used in graph-convolution layers [112, 136], which aggregates input process values x, node weights v, and edge weights w across node neighbors $N(i)$ to predict an output process value x'_i:

$$x'_i = \sum_{j \in N(i)} \frac{x_j \cdot v_j \cdot w_{i,j}}{\sqrt{|N(i)|}\sqrt{|N(j)|}}.$$

Since CYPRESS uses a generic message passing algorithm used for graph convolution layers, CYPRESS is fully differentiable and can be implemented with off-the-shelf libraries for graph convolution. CYPRESS matches the input and output structure used by other model architectures for ICS anomaly detection (described in Section 5.2.2) and can therefore also be trained with reconstruction-based losses.

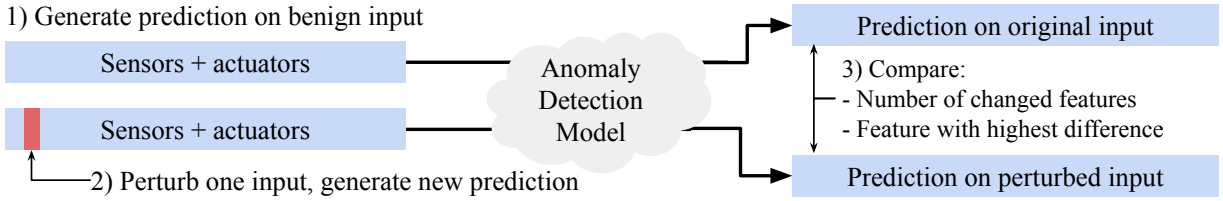
CYPRESS is *sparse*, as it learns weights for graphs with only a limited number of edges, and CYPRESS is *structured*, as it only learns weights defined by this graph, and no nodes or edges are created during training. Effectively, CYPRESS only learns a fixed set of weights from ICS process data, setting an upper bound on the number of inputs connected to each predicted output.

Next, in Section 5.4, we show that CYPRESS models produce outputs that are sparser and more accurate for attributing input manipulations, compared to deep-learning models. Furthermore, in Section 5.5, we show that CYPRESS is more efficient than deep-learning models such as GRUs and LSTMs. CYPRESS learns fewer parameters by several orders of magnitude, and performing inference with CYPRESS is up to $50\times$ faster. .

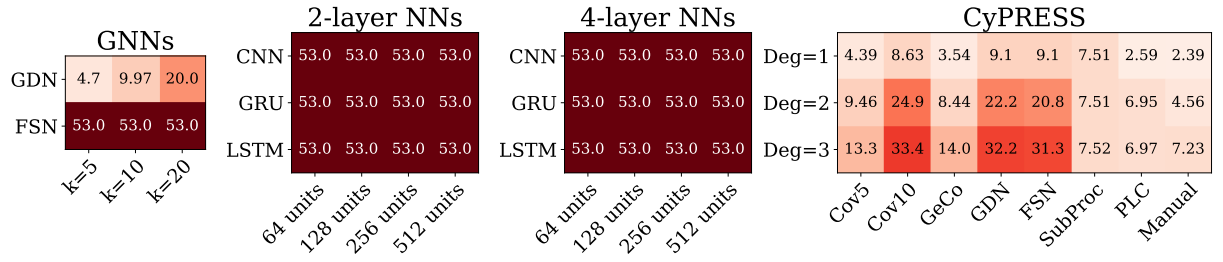
5.4 Analyzing spurious relationships learned by ICS anomaly-detection models

Deep-learning models based on fully connected model architectures, such as CNNs, GRUS, and LSTMS, learn dense representations of ICS. However, one problem with fully connected neural networks is that they are susceptible to spurious correlations—relationships learned from data that are not inherent to the underlying task [88, 145]. In practice, these misleading predictions can lead to errors in detection and diagnosis, as features that are not related to an underlying anomaly will appear to be anomalous.

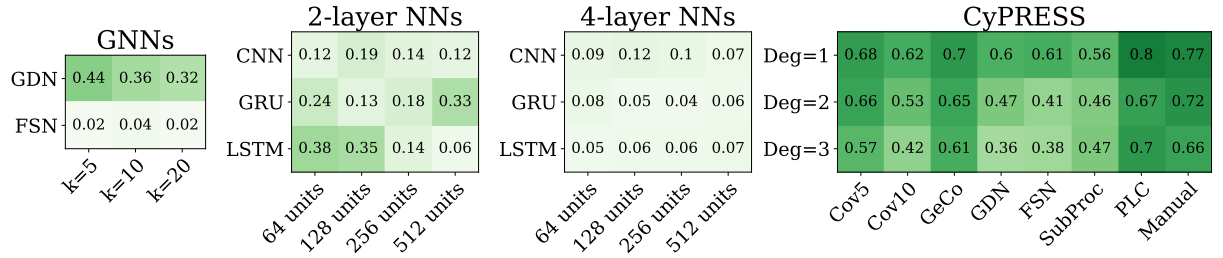
To measure the prevalence of spurious relationships in ICS anomaly detection models, we design a counterfactual benchmark test. In our counterfactual test, we perform a structured set of perturbations on benign input data. We first take a time-series segment of input data from training data, feed it to the reconstruction model, and record the model's output. Then, we take



(a) Overview of the steps used in our counterfactual test: we collect a benign input and its corresponding prediction (step 1); we perturb one input feature and generate a new prediction (step 2); and we compare the two predictions and calculate their difference (step 3). We measure the number of changed features and if the feature with the highest difference matches the feature that was perturbed.



(b) The average number of features changed from our counterfactual tests for various models.



(c) Match rates from our counterfactual test for various models.

Figure 5.3: We perform a counterfactual test for learned spurious relationships: given a benign input, we produce counterfactual input and observe the difference in prediction. Our test reveals that most state-of-the-art forecasting models fail to make predictions that (i) are sparse in change and (ii) align with input manipulations.

the input data segment, select a single input value, and increase it by three standard deviations (i.e., a 99.7th percentile event in the standard, normal distribution). We then feed the perturbed input data segment to the model again and compute the difference between the original output and the new output on the perturbed data. After repeating across the full space of possible model inputs, we compute the average for (i) the number of features that change after one perturbation and (ii) the rate at which the feature with the highest difference matches the feature that was originally perturbed (i.e., match rate). Figure 5.3a shows an overview of how our counterfactual test is performed. Since the predominant paradigm is to compute reconstruction errors based on model predictions, a desirable model should produce an output with high reconstruction errors for the feature that was originally perturbed.

We compute counterfactuals across a selection of state-of-the-art models from prior work, trained on the TEP dataset (using the methodology described in Section 5.5. For CyPRESS, across various configurations, such as degree of connection and graph specification (e.g., man-

ual, PLC-based, covariance-based, etc.), CYPRESS outperforms baseline models on our counterfactual test. Figure 5.3c shows the match rate of our test; we find that for graph-based models (i.e., GDN [29] and FSN [54]) and deep-learning models (i.e., CNN [70], GRU [77], and LSTM [148]), there are relatively few cases in which the feature with the highest difference matches the original manipulation. For all models, less than 40% of all tests result in a match, and many models have match rates below 10%. In contrast, CYPRESS produces outputs that match at higher rates, with many models achieving a match rate above 60%.

Figure 5.3b shows the average number of features that are changed after our counterfactual test; with GDNs, the number of features that can change is defined by the hyperparameter k, which defines the number of edges that are extracted from the attention weights to define the graph convolution. For all other models, all 53 features (out of 53 possible features in TEP) change. Whereas all baseline models produces outputs in which *every output feature* changes value, CYPRESS produces outputs in which the changes in output are localized to far fewer features. CYPRESS upper bounds the number of features that can change from any given input and uses fewer trained parameters; thus, CYPRESS is less susceptible to learning spurious relationships from data.

5.5 Evaluation setup

In this section, we describe how we implement baseline models (Section 5.5.1), and how we implement CYPRESS (Section 5.5.2).

5.5.1 Baseline models

As a baseline for our evaluation, we implement and train a variety of models used in prior work for anomaly detection, including lightweight ML models (AR [52, 133], GeCo [142]), deep-learning models (CNNs [70], GRUs [77], and LSTMs [148]), and graph-based models (GDN [29], FSN [54]).

We train models on the SWaT [48], TEP [17, 32], and CTown [37, 92] datasets (described in Section 2.2). All models are trained with unsupervised learning, which means that they are trained only on benign data from normal ICS operations. We define x_t as the full vector of d process values at a given time t. Given an input sequence of process values $(x_{t-1}, x_{t-2}, ...x_{t-h})$, we train each model $F()$ with a forecasting task, which predicts the next vector of process values and minimizes the mean-squared-error L2 loss between the observed process values and their prediction:

$$x'_t = F_w(x_{t-1}, x_{t-2}, ...x_{t-h})$$
$$w^* = \arg\min_w ||x_t - x'_t||^2$$

To train and implement GeCo models, we use the code provided by its original authors [142]. To implement the various baseline models, we use implementations provided by the original authors for GDNs and FSNs [29, 54], or we implement the models ourselves by using the predefined layer types in PyTorch for AR, CNNs, GRUs, and LSTMs. We train multiple instances

of each model across multiple hyperparameters, varying the number of layers and the number of units per layer for CNNs, GRUs, and LSTMs, or varying the connectivity factor k for GDNs and FSNs. We train all models with an input history of 10 timesteps. Following our methodology in Chapter 3, we train each model with the Adam optimizer for 50 epochs on 80% of the benign dataset, use 20% of the benign dataset to compute validation error for early stopping, and train five models for each set of model hyperparameters to account for training variance. In all of our reported results in Section 5.6, we report the average across these five models.

5.5.2 CyPRESS

Like the baseline models, we design and implement CyPRESS for a forecasting task: we train CyPRESS to predict the next process-value vector given an input history of the 10 previous process values. We implement CyPRESS based on our description in Section 5.3.1, using graph convolution layers from the Pytorch Geometric library[1]. CyPRESS is parameterized by the number of graph convolution operations that are performed during inference (i.e., the graph degree) and the method used for graph specification; we propose a variety of manual and automatic methods for graph specification.

Manual graph specification. We first define CyPRESS model with manually specified graphs (CPR-Manual). We create graph representations of CTown and TEP by inspecting the PLC logic in their simulators and by inspecting the descriptions of their physical processes[2]. We manually create a graph representation of SWaT based on prior publications that describe the connectivity of different components and the inputs and outputs for each PLC [5, 48].

Automatic graph specification. To automatically define graphs for CyPRESS, we draw inspiration from the taxonomy provided by Erba et al.[35], which defines the types of inconsistencies that should be detected by ICS anomaly-detection models. For statistical consistency, we define edges based on the top-5 or top-10 features with the highest covariance observed in the training data (CPR-Cov5 or CPR-Cov10) or based on the top-10 highest-valued attention weights from our trained graph-based models (CPR-GDN and CPR-FSN). To test for spatial consistency, we define edges based on predefined groups of PLCs and subprocesses[3] (CPR-PLC and CPR-SubProc). Lastly, we compare our automatic strategies to an invariant-inspired approach, in which we define edges based on the features that are selected for each function template in the pre-trained GeCo model (CPR-GeCo) [142].

Detecting anomalies. To perform anomaly detection with CyPRESS (and with baseline models), we must define a threshold for predicting an anomaly based on reconstruction errors. In

[1] `https://pytorch-geometric.readthedocs.io/en/2.6.1/`

[2] PLC logic is explicitly defined in these simulators, so we can precisely create edges based on logical relationships. However, the physical relationships are estimated from a process diagram in TEP and from EPANet configuration files in CTown.

[3] For CTown and SWaT, subprocess groups and PLC groups are identical. For TEP, we define subprocess groups based on the names of components in the simulator.

this work, we consider two commonly used strategies for thresholding: MSE-based thresholds and CUSUM. To use an MSE-based threshold, we compute the anomaly-detection model's per-feature reconstruction error on the validation data. Following prior work, we then use each feature's maximum validation error as a detection threshold [70, 98]. As an extension to MSE-based thresholds, other prior works use the cumulative sum of errors (CUSUM) [23, 133, 142], a stateful detection thresholding strategy. For each timestep t, we compute the absolute reconstruction error $e_t = |x_t - x'_t|$ for each feature and compute its rolling cumulative sum during testing $CUSUM_t = \max(0, CUSUM_{t-1} + e_t - \delta)$. δ is defined as the drift parameter, which serves as a constant decay factor for the accumulated CUSUM errors. Based on implementations used in prior work, we set the CUSUM detection threshold using benign validation data: we set the drift parameter δ to the average reconstruction error on validation data plus one standard deviation, and we set the detection threshold per feature to the maximum CUSUM value on the validation dataset.

Cost of CyPRESS. After implementing CyPRESS, we compare the size and computational cost of CyPRESS compared to baseline models.

Table 5.2: On left, the number of trainable parameters for each model trained on the TEP dataset. Models such as AR and GeCo use fewer features (i.e., 200–600) and deep-learning models use more features (i.e., 77K–16M) compared to CyPRESS, which uses 2000–6000 features. On right, the amount of time taken to perform a single inference pass (based on the TEP dataset). A single inference pass with CyPRESS (0.43–0.81ms) is almost always faster than any deep-learning model (0.8–22.3ms) and is $\approx$ 5–50$\times$ faster than the best-performing GRU and LSTM models.

		Number of Trainable Parameters		
	GeCo	203		
	AR	583		
CyPRESS	Manual	2,286		
	SubProc	5,004		
	PLC	2,466		
	Cov5	3,348		
	Cov10	5,598		
	GeCo	2,898		
	GDN	5,841		
	FSN	5,841		
	GDN	4,673		
	FSN	1.5M		
		128-unit	*256-unit*	*512-unit*
CNN	*2-layer*	77K	252K	898K
	4-layer	176K	647K	2.5M
	8-layer	374K	1.4M	5.6M
GRU	2-layer	176K	647K	2.5M
	4-layer	374K	1.4M	5.6M
	8-layer	770K	3.0M	11.9M
LSTM	*2-layer*	233K	858K	3.3M
	4-layer	497K	1.9M	7.5M
	8-layer	1.0M	4.0M	15.9M

		Inference Time (ms)		
	GeCo	0.08		
	AR	1.42		
		Deg = 1	*Deg = 2*	*Deg = 3*
CyPRESS	Manual	0.43	0.55	0.81
	SubProc	0.47	0.59	0.71
	PLC	0.44	0.58	0.66
	Cov5	0.43	0.56	0.68
	Cov10	0.47	0.62	0.73
	GeCo	0.43	0.56	0.76
	GDN	0.46	0.62	0.74
	FSN	0.45	0.62	0.74
		k = 5	*k = 10*	*k = 20*
	GDN	2.46	2.50	2.66
	FSN	8.01	7.99	8.23
		128-unit	*256-unit*	*512-unit*
CNN	2-layer	0.80	0.88	0.96
	4-layer	0.97	1.14	1.40
	8-layer	1.38	1.56	2.28
GRU	2-layer	2.45	2.18	2.73
	4-layer	3.00	3.61	4.63
	8-layer	5.20	5.87	8.11
LSTM	2-layer	15.04	15.44	19.54
	4-layer	14.65	15.58	18.95
	8-layer	15.22	22.26	22.23

Table 5.3: The average F1 score across all attacks on CTown (left), TEP (middle), and SWaT (right) for all the anomaly-detection models evaluated in this work. F1 scores are averaged across attacks and across five repeated trials. We highlight models that are "on par" (if it falls within 0.03 of the maximum score for that dataset). Various configurations of CYPRESS, GRUs, and LSTMs perform best across datasets.

		CTown			**TEP**			**SWaT**		
	AR			0.756			0.819			0.772
	GeCo			0.566			0.819			0.788
		Deg = 1	*Deg = 2*	*Deg = 3*	*Deg = 1*	*Deg = 2*	*Deg = 3*	*Deg = 1*	*Deg = 2*	*Deg = 3*
	Manual	0.736	0.793	0.803	0.848	0.834	0.876	0.748	0.767	0.622
	SubProc	-	-	-	0.792	0.829	0.820			
	PLC	0.761	0.907	0.895	0.748	0.777	0.775	0.758	0.766	0.771
	Cov5	0.747	0.892	0.930	0.817	0.835	0.831	0.237	0.788	0.791
	Cov10	0.761	0.901	0.935	0.869	0.819	0.844	0.784	0.795	0.804
	GeCo	0.742	0.779	0.808	0.863	0.878	0.856	0.770	0.803	0.799
	GDN	0.792	0.897	0.830	0.750	0.728	0.794	0.771	0.805	0.803
	FSN	0.758	0.891	0.931	0.779	0.755	0.728	0.783	0.786	0.739
		k = 5	*k = 10*	*k = 20*	*k = 5*	*k = 10*	*k = 20*	*k = 5*	*k = 10*	*k = 20*
GNNs	GDN	0.897	0.904	0.879	0.689	0.669	0.686	0.374	0.779	0.771
	FSN	0.878	0.902	0.885	0.780	0.763	0.785	0.785	0.781	0.782
		128-unit	*256-unit*	*512-unit*	*128-unit*	*256-unit*	*512-unit*	*128-unit*	*256-unit*	*512-unit*
CNN	*2-layer*	0.846	0.873	0.759	0.671	0.647	0.613	0.741	0.716	0.741
	4-layer	0.862	0.858	0.822	0.745	0.665	0.670	0.736	0.593	0.729
	8-layer	0.872	0.688	0.721	0.722	0.760	0.706	0.766	0.769	0.757
GRU	*2-layer*	0.936	0.943	0.940	0.766	0.757	0.791	0.799	0.798	0.801
	4-layer	0.942	0.944	0.945	0.714	0.593	0.550	0.797	0.802	0.764
	8-layer	0.883	0.902	0.924	0.773	0.752	0.764	0.794	0.771	0.789
LSTM	*2-layer*	0.944	0.945	0.945	0.730	0.833	0.857	0.795	0.799	0.799
	4-layer	0.942	0.939	0.944	0.774	0.743	0.685	0.795	0.812	0.806
	8-layer	0.919	0.938	0.938	0.789	0.784	0.755	0.795	0.801	0.799

Table 5.2 shows the number of trainable parameters and the average inference time for each model used in this work. To measure the inference time required by various ML models, we perform an experiment with a commodity desktop (Intel Xeon E-2136 3.30GHz CPU, 32 GB RAM); we measure the average time taken to perform a single inference pass with the model, across five trials. Overall, CYPRESS is far more efficient than GRUs and LSTMs; it uses over $1000\times$ fewer model parameters and performs inference up to $50\times$ faster.

5.6 Evaluation results

We first evaluate the detection performance of CYPRESS and baseline models (Section 5.6.1). Next, following the methodology used in prior work [42], we evaluate the attribution performance of CYPRESS and baseline models (Section 5.6.2). Finally, we evaluate how CYPRESS compares to baseline models against attackers with increased system access and stealthier attack strategies (Section 5.6.3).

Table 5.4: The average range-F1 scores across all attacks on CTown (left), TEP (middle), and SWaT (right) for all the anomaly-detection models evaluated in this work. Range-F1 scores are averaged across attacks and across five repeated trials. We highlight models that are "on par" (if it falls within 0.1 of the maximum score for that dataset). Various configurations of CYPRESS, GRUs, and LSTMs perform best across datasets.

		CTown			**TEP**			**SWaT**		
	AR			0.879			0.617			0.146
	GeCo			0.852			0.656			0.262
		Deg = 1	*Deg = 2*	*Deg = 3*	*Deg = 1*	*Deg = 2*	*Deg = 3*	*Deg = 1*	*Deg = 2*	*Deg = 3*
CyPRESS	Manual	0.855	0.898	0.883	0.756	0.743	0.752	0.151	0.155	0.150
	SubProc	0.721	0.564	0.689	0.719	0.702	0.720	0.189	0.189	0.154
	PLC	0.721	0.564	0.689	0.537	0.688	0.735	0.189	0.189	0.154
	Cov5	0.799	0.718	0.661	0.625	0.717	0.717	0.159	0.175	0.173
	Cov10	0.728	0.647	0.604	0.698	0.751	0.697	0.157	0.138	0.122
	GeCo	0.801	0.675	0.760	0.722	0.729	0.750	0.139	0.145	0.129
	GDN	0.774	0.625	0.696	0.658	0.639	0.731	0.170	0.104	0.113
	FSN	0.746	0.656	0.679	0.738	0.714	0.701	0.154	0.154	0.144
		k = 5	*k = 10*	*k = 20*	*k = 5*	*k = 10*	*k = 20*	*k = 5*	*k = 10*	*k = 20*
GNNs	GDN	0.635	0.671	0.745	0.625	0.597	0.511	0.177	0.235	0.229
	FSN	0.490	0.551	0.450	0.492	0.540	0.450	0.197	0.171	0.172
		128-unit	*256-unit*	*512-unit*	*128-unit*	*256-unit*	*512-unit*	*128-unit*	*256-unit*	*512-unit*
CNN	*2-layer*	0.882	0.793	0.583	0.591	0.567	0.525	0.312	0.225	0.269
	4-layer	0.617	0.605	0.629	0.693	0.579	0.593	0.261	0.202	0.226
	8-layer	0.824	0.579	0.728	0.461	0.487	0.535	0.235	0.197	0.171
GRU	*2-layer*	0.622	0.869	0.702	0.736	0.736	0.734	0.383	0.369	0.372
	4-layer	0.868	0.823	0.968	0.660	0.682	0.656	0.398	0.362	0.299
	8-layer	0.641	0.823	0.789	0.732	0.600	0.648	0.272	0.250	0.250
LSTM	*2-layer*	0.862	0.931	0.859	0.669	0.811	0.838	0.357	0.356	0.345
	4-layer	0.894	0.872	0.968	0.763	0.759	0.806	0.310	0.338	0.326
	8-layer	0.883	0.977	0.976	0.573	0.504	0.618	0.263	0.284	0.316

5.6.1 Anomaly detection

We first compare the anomaly-detection performance of CYPRESS with the performance of baseline models. For each dataset and each trained model, we use both the MSE-based and CUSUM thresholding strategies described in Section 5.5.1 to predict time-series anomalies from real-time process data. Similar to prior work, we find that the performance of thresholding strategies varies and that the best choice of thresholding strategy is not consistent across datasets [23, 41, 75, 133, 142]. We use a per-feature CUSUM thresholding strategy for the CTown dataset and a per-feature MSE-based thresholding for the TEP and SWaT datasets.

Results with point-based metrics. Table 5.3 shows the point-F1 scores for all model configurations on each dataset.

Although many of the best F1 scores for each dataset (e.g., 2-layer, 512-unit LSTM for CTown; the 4-layer, 256-unit LSTM for SWaT) are produced by deep-learning models, many configurations of CYPRESS can perform similarly as well or even better (e.g., CPR-GeCo on TEP). We define a model as "on par" if it falls within 0.03 of the maximum F1 score for the given dataset (highlighted in Table 5.3); many configurations of CYPRESS, GRUs, and LSTMs are on par with the best-performing models. In particular, multiple configurations of CPR-Manual,

CPR-Cov5, CPR-Cov10, Cov-FSN, and CPR-GeCo achieve scores that are on par with deep-learning models.

CYPRESS uses far fewer parameters than GRUs and LSTMs by several orders of magnitude (i.e., thousands of parameters compared to millions, as shown in Table 5.2). Despite using far fewer parameters, CYPRESS can detect anomalies with performance that is on par with larger, more complex anomaly-detection models.

Finding 12: CYPRESS achieves comparable detection performance to deep-learning-based approaches while using up to $1000\times$ fewer trained parameters.

Results with range-based metrics. Table 5.4 shows the range-F1 scores (based on our definition in Section 3.5.2) for all models, configurations, and datasets. Similar to our findings in Chapter 3, we find that there is a higher variance across range-F1 scores, and that many models' range-F1 scores are lower than their corresponding point-F1 scores. As with our analysis in Section 5.6.1, we define a model as "on par" if it achieves a score that is within 0.1 (rather than 0.03, due to higher variance) of the maximum score for that dataset, and we highlight these scores in Table 5.4. In general, large deep-learning models such as LSTMs generally perform best, and some configurations of CYPRESS can perform similarly.

We make some additional observations regarding CYPRESS and anomaly detection with range-based metrics. First, we note that our results are in line with prior works [59, 142, 143] which find that point-based metrics like the F1 score can be misleading; several models with high F1 scores produce relatively low range-F1 scores. This is caused by the fact that, despite using time-series inputs, ML-based anomaly detection models are not trained to make contiguous predictions, and additional thresholding techniques are often required to encourage such behavior. Second, we note that our results are also in line with prior work that finds that anomaly-detection on the SWaT dataset is unstable in practice [70, 75, 129, 135, 148]; because all attacks on SWaT are performed within one single ICS execution (as opposed to CTown and TEP, in which each attack is performed in a unique execution), several prior works have noted issues with feature drift, data cleaning, and mislabeled anomalies as a result. We emphasize that these issues with the SWaT dataset affect all methods (i.e., CYPRESS and baselines), and that methods for feature cleaning and optimal thresholding tuning are not the focus on this work.

5.6.2 Anomaly attribution

Next, we compare the anomaly-attribution performance of CYPRESS to that of GRUs and LSTMs (the best performing models for detection), using the AvgRank metric and methodology from Chapter 4. Since each model detects each anomaly with different timings and accuracies and we want to compare attribution performance systematically, we compute attributions based on the ground-truth start of each anomaly. As suggested by our findings in Chapter 4, this simulates a setting in which attributions are used for post-hoc anomaly diagnosis. For each labeled attack in each dataset, we capture the first time-series window of anomalous data and use the reconstruction errors from each anomaly detection model as an attribution score. Figure 5.4 shows

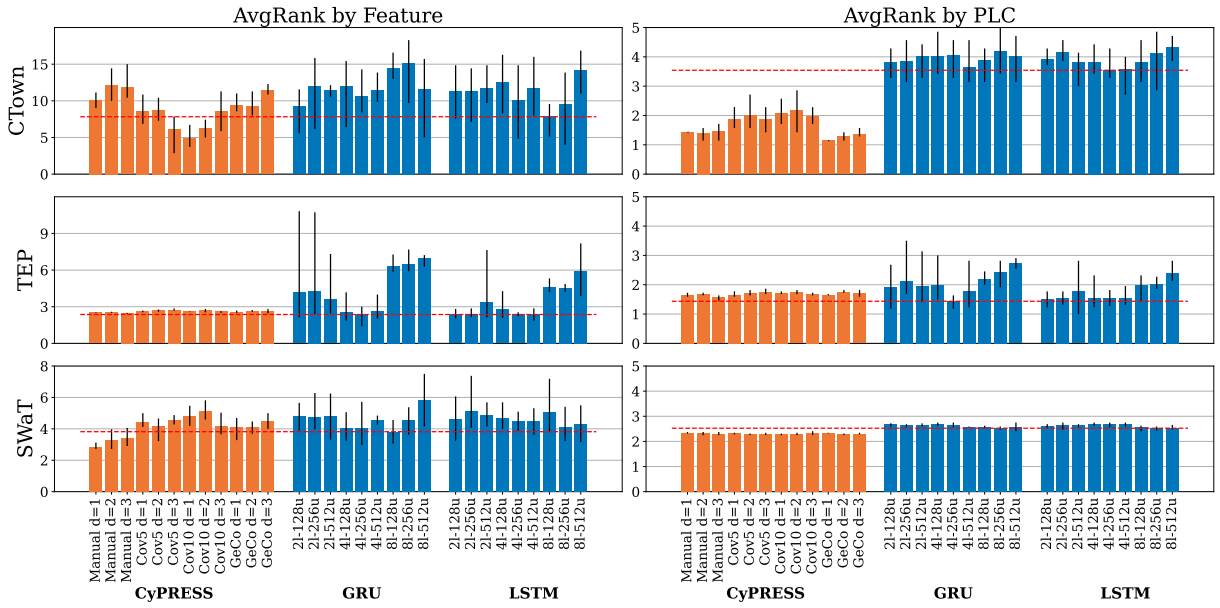


Figure 5.4: The AvgRank (lower is better) based on individual features (left) and based on PLCs (right) using each model's reconstruction errors for CTown (top), TEP (middle), and SWaT (bottom). Lower scores indicate that the model's outputs are more accurate for identifying the underlying manipulated component or PLC in an anomaly. CYPRESS (in orange, on left) produces lower rankings than GRUs and LSTMs (in blue, on right).

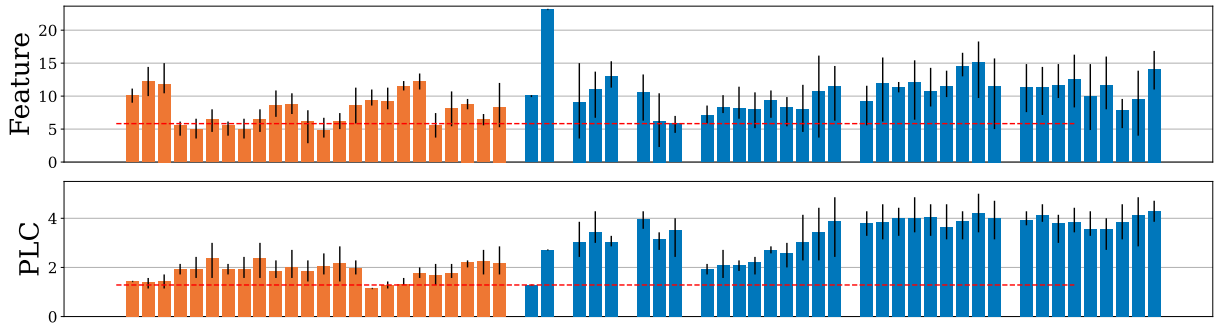
the average ranking of the attacked feature (top of each subfigure) and the average ranking of the PLC containing the attacked feature (bottom of each subfigure) for CYPRESS, GRUs, and LSTMs for the CTown, TEP, and SWaT datasets (lower is better).

CYPRESS produces attributions with better rankings for attacker-manipulated features. For instance, the feature AvgRank for CYPRESS on TEP falls within 2.4–2.8, whereas many deep-learning models produce a feature AvgRank above 5 (e.g., the 8-layer, 512-unit LSTM produces an AvgRank of 5.9). This trend also holds true for the SWaT dataset (bottom left of Figure 5.4). We also find that the PLC AvgRanks produced by CYPRESS are lower than those of GRUs and LSTMs. and the attribution performance is consistent across various CYPRESS configurations. In other words, the best-performing attributions can be drawn from most CYPRESS models.

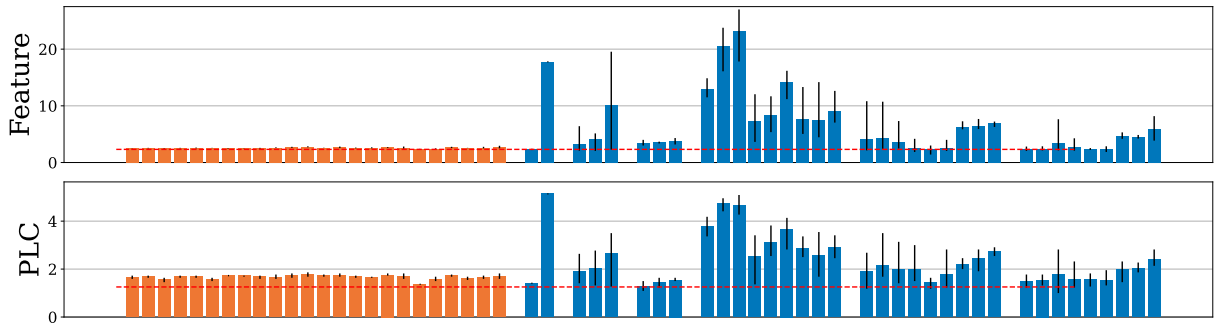
CYPRESS also produces attributions that are more consistent, particularly on the TEP and SWaT datasets. We find that both the feature AvgRank and PLC AvgRank have lower variance that those from deep-learning models; the maximum and minimum values are shown on the bars in Figure 5.4.

For CTown, we find that the feature AvgRank of CYPRESS is less consistent, and the attribution performance similar to that of deep-learning models. However, the PLC AvgRank is much lower, and CPR-GeCo models in particular are the most accurate (e.g., the PLC AvgRank is within 1.1–1.4).

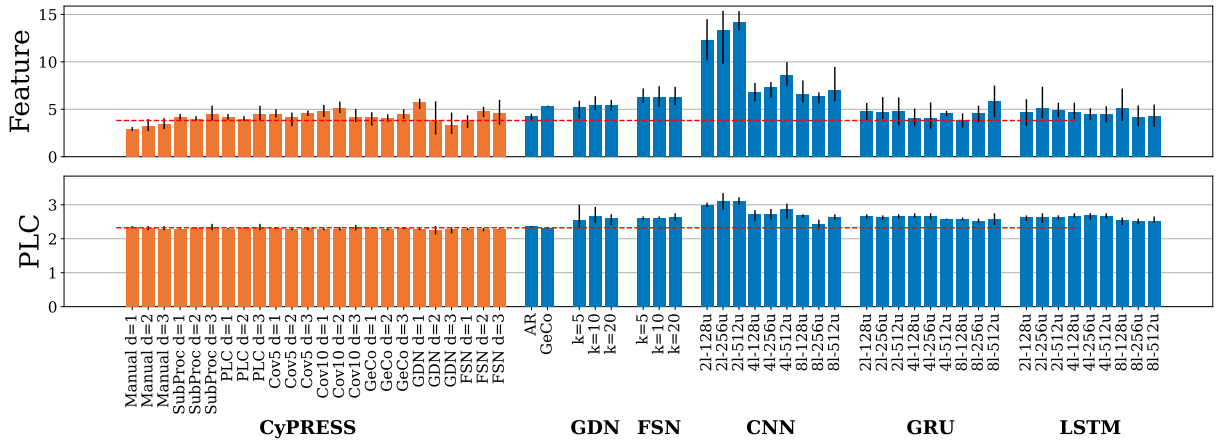
Figure 5.5 shows the full set of results for all models on the CTown (top), TEP (middle), and SWaT (bottom) datasets. In general, CYPRESS produces attributions (both when ranking features and when ranking PLCs) that are more accurate than baseline models.



(a) Full attribution results on CTown.



(b) Full attribution results on TEP.



(c) Full attribution results on SWaT.

Figure 5.5: For all three datasets (CTown at top, TEP at middle, SWaT at bottom), we compute the AvgRank (lower is better) based on individual features (top) and based on PLCs (bottom) using the reconstruction errors produced by various models. Lower scores indicate that the model produces outputs that are better aligned to the underlying manipulated component or PLC in each anomaly. We find that CyPRESS models (light shading, on left), produce lower rankings than baseline models (dark shading, on right).

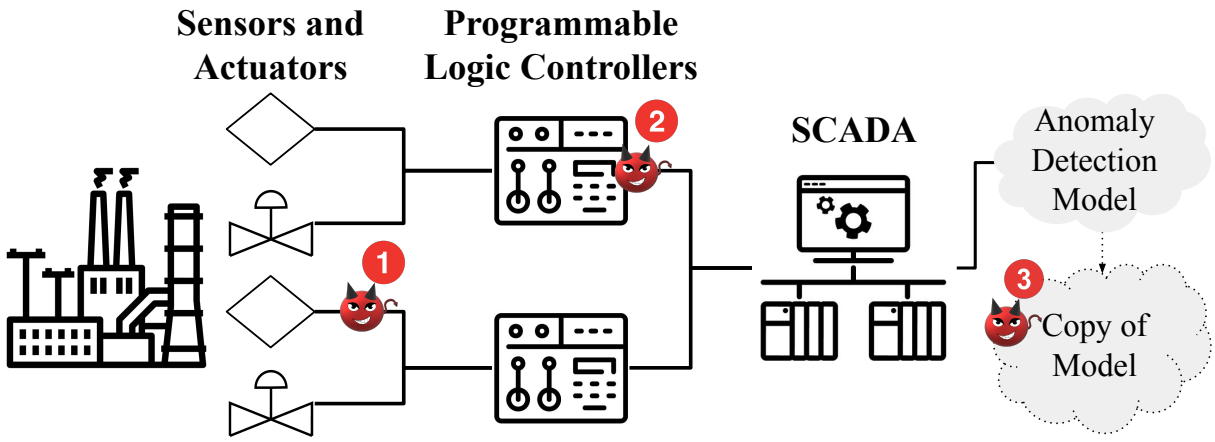


Figure 5.6: An overview of an ICS and the threat models we consider: attacks at the sensor level (#1), attacks at the PLC level (#2) and attacks that leverage knowledge of the anomaly-detection model (#3).

Finding 13: Compared to the best-performing baseline models, CYPRESS is more accurate when used to attribute anomalies to their manipulated components or manipulated PLCs.

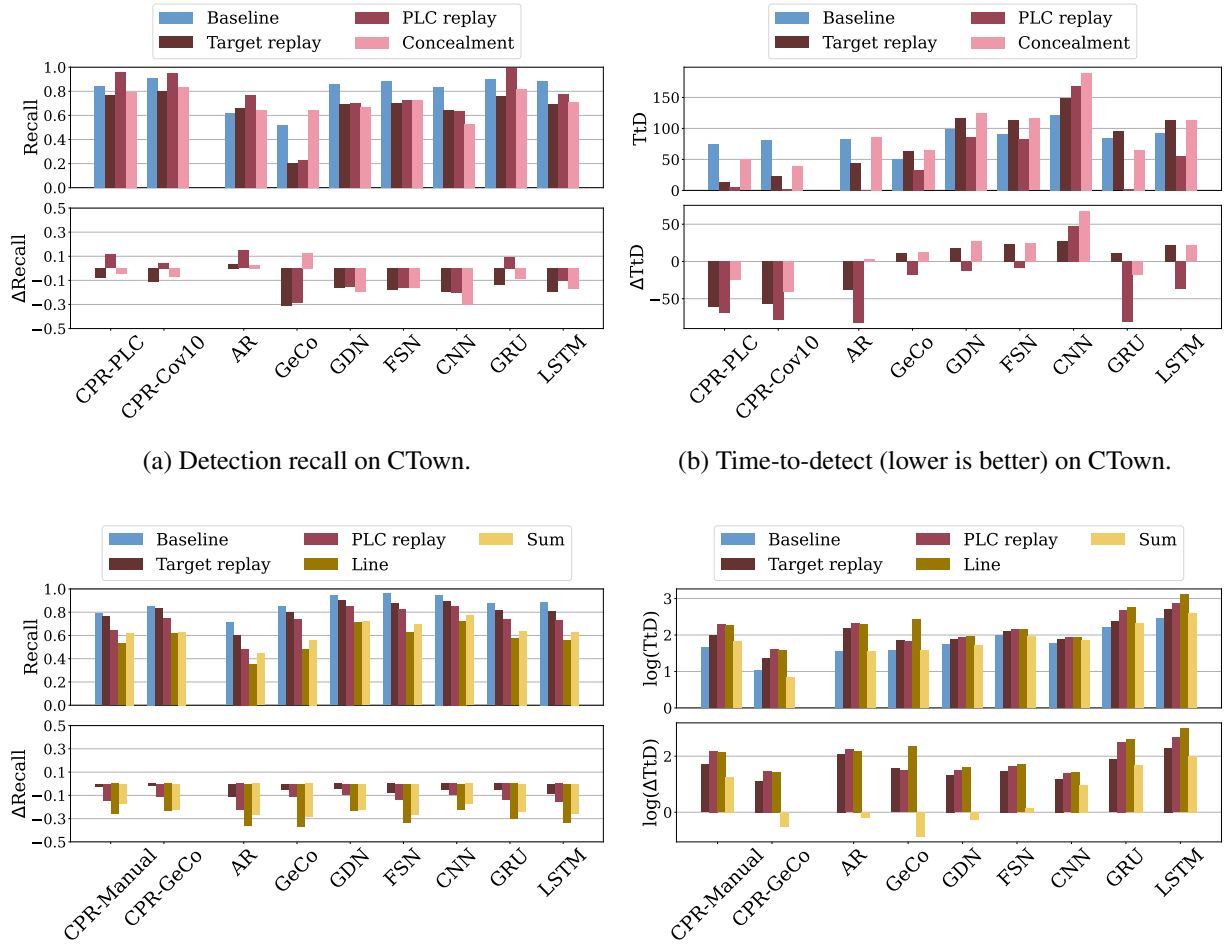
5.6.3 Robustness to stealthy attack strategies

We evaluate CYPRESS's robustness to stealthier attack strategies, such as using variable manipulation patterns rather than constant-valued manipulations [42], concealing manipulated values with replay attacks [35, 37], and using learning-based evasion attacks [36, 37]. For each of these stealthier attacks, we describe the additional levels of attacker access required, based on the examples shown in Figure 5.6.

To evaluate performance against stealthier attacker strategies, we must first select against a set of well-performing models for detection. For different types of model (e.g., CNNs, GRUs, CYPRESS), we select a well-performing set of models based on the detection results from Section 5.6.1: we select models with high recall, F1, and range-F1 scores. We note that, since we do not have access to a simulator for SWaT and only have access to an attack dataset, we are unable to meaningfully perform replay attacks for evaluation. Thus, we evaluate robustness only with the CTown and TEP datasets.

Generic replay attacks [35]. We first evaluate against generic replay attacks, which assume that the attacker replays a subset of benign data to the anomaly-detection model while performing their attack. For example, an attacker with sensor-level access (1 in Figure 5.6) will concurrently transmit their harm-inducing values to the physical process while sending stale benign data to the anomaly-detection model.

To simulate generic replay attacks, we systematically capture and replay benign data for all attacks in the CTown and TEP datasets, using methodology from prior work. For each baseline, non-stealthy attack, we perform two types of replay attack: one attack that only replays benign



(a) Detection recall on CTown.

(b) Time-to-detect (lower is better) on CTown.

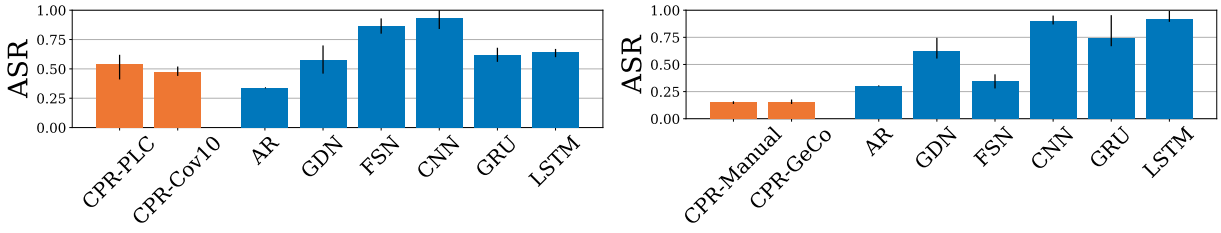
(c) Detection recall on TEP.

(d) Time-to-detect (lower is better) on TEP.

Figure 5.7: The average detection recall (left) and average time-to-detect (right, lower is better) after performing a stealthy attack (target replay, PLC replay, concealment, line manipulations, or sum manipulations) against best-performing models on CTown (top) or TEP (bottom). Absolute values are shown at the top of each subplot and differences between each stealthy attack and its baseline counterpart are shown at the bottom of each subplot. Although stealthy attacks cause all models to perform worse, CYPRESS is more effective at detecting stealthy attacks and produces a smaller performance degradation compared to baselines.

data from the target component ("Target replay" from location 1 in Figure 5.6), and one attack that replays benign data from all features that share a PLC with the target component ("PLC replay" from location 2 in Figure 5.6).

We evaluate anomaly detection models against generic replay attacks and report the change in recall and time-to-detect (TtD) in Figures 5.7a–5.7b. Compared to most baseline models on CTown, CYPRESS is more robust to generic replay attacks with only small changes in recall and faster time-to-detect (e.g., about 50 seconds faster for PLC replay). However, we find that replay attacks are effective at evading all models trained on TEP, including CYPRESS and deep-learning models.



(a) Attack success rate for evasion attacks on the CTown dataset.

(b) Attack success rate for evasion attacks on the TEP dataset.

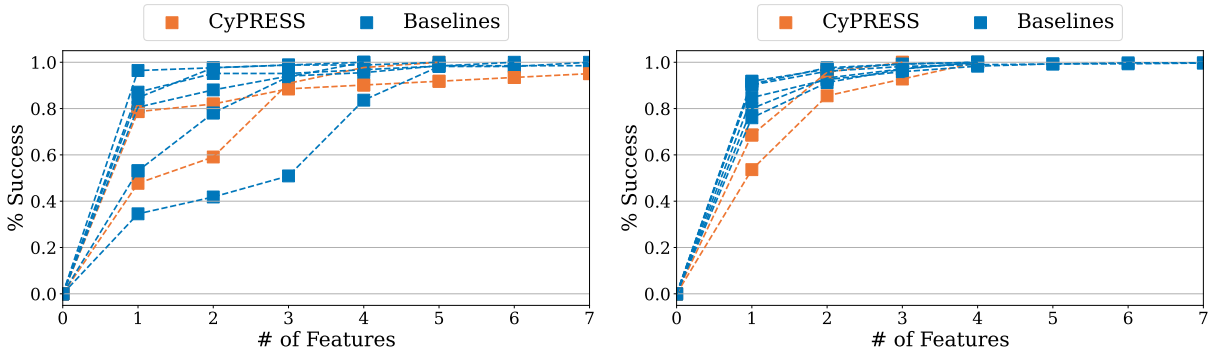
Figure 5.8: The attack success rates (ASR, lower is better) for optimized evasion attacks on the CTown (left) and TEP (right) datasets against various best-performing models. CYPRESS consistently produces a lower ASR than GRUs and LSTMs.

Concealment attacks [37]. To perform a concealment attack, the attacker manipulates two parts of the ICS concurrently, similar to a generic replay attack. Erba et al. provide modified versions of the attacks in the CTown dataset with concealment [37]. For each attack, different strategies such as network spoofing, PLC value spoofing, or a man-in-the-middle attack are used to conceal the target feature with a different, attacker-determined value; we present all of these strategies from prior work as a generic category of attack called "Concealment".

We evaluate each anomaly detection model against the provided concealment attacks and report the change in recall and detection latency in Figures 5.7a–5.7b, shown in red. Whereas most baseline models (CNNs, GRUs, LSTMs, GDNs, and FSNs) show a large drop in recall (i.e., recall drops by 0.1), the best-performing CYPRESS models only produce a minimal decrease in recall and reduction in time-to-detect.

Finding 14: Compared to deep-learning models, CYPRESS is more robust to concealment attacks, maintaining higher recall and faster times to detect.

Variable data injection [42]. (Chapter 4) In contrast to the constant-valued manipulations performed in our baseline datasets, we also explored the impacts of using variable patterns in data injection, based on our description in Section 4.4.3. These attacks assumes the same level of device access as the baseline attacks (1 in Figure 5.6), but replaces the constant-valued manipulation with stealthier counterparts: by either replicating the original sensor noise in the manipulation by adding a fixed value to the original signal ("Sum") or by performing the manipulation over a prolonged, linearly increasing period ("Line"). Figures 5.7c–5.7d show the results on the TEP dataset. In general, variable data injection attacks are the most difficult to detect; Figure 5.7c shows that the recall drops by over 15% for all baseline and CYPRESS models. However, CYPRESS is least affected by variable attack strategies: despite comparable decreases in recall, Figure 5.7d shows that the CPR-Manual model detects the evasion attacks with the lowest detection latency (e.g., 19.6 seconds for cumulative sum attacks and 36.5 seconds for line attacks).



(a) CDF of outcomes for evasion attacks on the CTown dataset.

(b) CDF of outcomes for evasion attacks on the TEP dataset.

Figure 5.9: The number of features required for successful evasion attacks (lower is better), for the CTown (left) and TEP (right) datasets. Compared to most baselines, CYPRESS is lower for most numbers of features, which indicates that fewer attacks can be performed with this level of access; an attacker must compromise a greater number of features in order to evade detection.

Optimized evasion attacks [36]. Finally, we evaluate the strongest attacker who has read access to the inputs and the outputs of the model itself, either by using a passive wiretap or by obtaining a copy of the anomaly detection model (3 in Figure 5.6). From this point of access, the attacker can search through potential manipulations by iteratively perturbing feature values and querying the model's output. The attacker can then repeat this process indefinitely until they generate an optimal reduction in output MSE or if they determine that no solution exists. In generating our optimized perturbations, we assume that the attacker is still limited to one point of write access, such as a single PLC[4] (2 in Figure 5.6).

We follow the methodology proposed by Erba et al. [36] to perform optimized evasion attacks. From each attack in each dataset, we sample 20 true positives detected by each model and attempt to modify features within the compromised PLC. When modifying features, we assume that the attacker is unable to retroactively manipulate time inputs, so they must manipulate the entire time-series input for a single feature with a fixed value. To find these optimal values, we perform a linear scan of 100 candidate values between the minimum and maximum values observed in the training distribution (i.e., in normal operation). We then select the feature and value which produces the largest drop in reconstruction error, before iteratively repeating the search process. We terminate the search if no more drops in reconstruction error are found after searching through all values in the compromised PLC and all candidate values.

Figure 5.8 shows the overall attack success rate (ASR) for evasion attacks on models trained on CTown (left) and TEP (right) respectively. Overall, we find that best-performing detection models such as LSTMs can be easily evaded, whereas the attack success rate on CYPRESS is far lower (e.g., on TEP, 89% of true positive samples that were originally detected by the LSTM are now undetected, whereas only 15% of these samples are evaded for CPR-Manual).

In Figure 5.9, we show the cumulative distribution of the number of features that must be

[4]If the attacker has full access to all process values, they can trivially spoof the entire system to bypass detection.

compromised for successful evasion across all attacks. Even in cases where an attacker can successfully bypass CYPRESS, on average, the attacker needs an increased level of access to succeed (i.e., they must compromise and manipulate more components), compared to most baseline models. For example, when modifying samples in TEP attacks, 54% of successful evasions on CPR-GeCo occur after compromising one feature, and 91% of successful evasions on LSTMs occur after compromising one feature; When modifying samples in CTown attacks, 48% of successful evasions on CPR-Cov10 occur after compromising one feature, and 96% of successful evasions on LSTMs occur after compromising one feature.

Finding 15: Compared to deep-learning models, CYPRESS is more robust to optimized evasion attacks; attacks on CYPRESS have lower success rates and require more attacker effort.

5.7 Future work and limitations

Exploring the interpretability of CYPRESS. Prior work extracts equations from GeCo's learned function templates and uses them as an aid for operator interpretation [142]. These equations follow a similar form to system identification [23, 46], which maps process outputs to combinations of process inputs in a tractable form. We propose that CYPRESS can be used similarly—we can convert edge weights used in CYPRESS's message passing algorithm into closed-form process-level equations. For example, we can extract the following function template from our CPR-Manual model trained on TEP ($s1$ refers to sensor #1 and $a2$ refers to actuator #2 in the anonymized TEP process)

$$\begin{aligned} x_{s1,t} = & \, x_{s1,t-1} \cdot v_{s1,t-1} \cdot w_{(s1,t-1)\rightarrow(s1,t)} \\ & + x_{a2,t-1} \cdot v_{a2,t-1} \cdot w_{(a2,t-1)\rightarrow(s1,t)} \\ & + x_{s1,t} \cdot v_{s1,t} \cdot w_{(s1,t)\rightarrow(s1,t)} \end{aligned}$$

Alternatively, this function template could also be substituted or combined with the corresponding process data and weight values to provide additional context for operators:

$$\begin{aligned} x_{s1,t} = & \, (-0.19) \cdot (-1.85) \cdot (0.76) \\ & + (-0.03) \cdot (2.44) \cdot (-2.39) \\ & + (-0.01) * (-0.02) \cdot (-0.02) \end{aligned}$$

A promising area of future work would be to investigate the best methods for converting anomaly-detection models into interpretable forms, such as determining if operators find process-level equations interpretable, if operators find graph-based information useful in practice, and if operators prefer certain types of visualization.

Exploring mixtures of anomaly detection models. In Chapter 3 and Chapter 4, we explored methods for tuning and configuring anomaly-detection methods for different objectives, such as

higher precision, higher recall, faster detection, or more accurate attribution. We found that CYPRESS performs well when applied for attribution in post-hoc diagnosis (Section 5.6.2) and detects stealthy attacks more quickly than deep-learning models (Section 5.6.3). Thus, we propose that CYPRESS can be used in conjunction with other approaches; for example, a deep-learning model could be used to detect anomalies with high recall in real time and CYPRESS could help diagnose these detected anomalies. Alternatively, CYPRESS could be used as an additional layer of defense against evasion attacks, detecting attacks that are missed by deep-learning-based approaches. We argue that CYPRESS provides useful functions for ICS anomaly detection, despite not solely achieving the highest F1 and TaF1 scores (Section 5.6.1). A promising area of future work would be to explore approaches that use systems of multiple anomaly-detection models for different objectives, including CYPRESS.

Limitations of using CYPRESS in practice. One limitation of using CYPRESS in practice is that operators must specify feature groupings for training. We explored several alternatives, ranging from manual specification of the ICS to automated specification from statistical measures. In practice, organizations that manage and operate anomaly-detection systems may prefer certain types of specification, based on their level of ML expertise, data availability, and their trust in automated methods. We intend for CYPRESS to be a general-purpose solution that can support various organizational preferences. In Chapter 6, we show that organizations vary widely in their needs and requirements for ML-based approaches. A promising area of future work would be to investigate how various forms of guidance for CYPRESS impact deployability and trust in practice and to investigate ways to better automate ICS specification to further ease adoption.

5.8 Summary

In this chapter, we propose CYPRESS, a novel model architecture for ICS anomaly detection. Unlike deep-learning-based approaches commonly used in prior work, CYPRESS learns structurally sparse representations of ICS based on inter-feature relationships. These relationships between ICS components can be manually specified (e.g., from a process diagram) or automatically inferred (e.g., from learned invariants). We show that CYPRESS achieves detection results that are on par with larger deep learning models while using far fewer trained parameters; CYPRESS avoids learning spurious relationships and can produce attributions that are more accurate in identifying manipulated components; and CYPRESS is more robust to stealthy evasion attacks.

Chapter 6

Examining practitioner's perspectives of ML-based tools for ICS alarms

The other works described in this thesis measure the effectiveness of existing anomaly-detection approaches (Chapter 3 and 4) and propose adaptations to them (Chapter 4 and 5), but these works are based on experimental systems used exclusively in research, rather than real ICS in practice.

It remains unclear how effective such ML-based anomaly-detection approaches would be when deployed and used in practice, and a strong consideration of both organizational and end-user needs is required to build this understanding [13].

In this chapter, I describe an interview-based study of practitioners that monitor and control ICS. We ask practitioners about the systems that use to detect anomalies and raise alarms, the tasks they perform with alarm data, and their perspectives on adopting AI[1] to support these tasks. Based on their responses, we determine if current anomaly-detection approaches proposed in research are suitable for ICS environments and make recommendations to improve the adoption of machine-learning-based tools for protecting ICS. The work described in this chapter is published in the *Proceedings of the Twenty-First Symposium on Usable Privacy and Security* (SOUPS 2025) [43].

6.1 Introduction

In practice, organizations typically use expert-defined rules to detect ICS anomalies [13, 14, 20, 87]. Researchers commonly propose approaches based on machine learning or artificial intelligence (AI) to improve the state of the art in ICS anomaly detection [29, 31, 41, 70, 79, 85], but these approaches are rarely used in practice. A recent survey of ICS practitioners found that only 10% of ICS use any form of AI on process data [26], and reports of AI being deployed in ICS in practice have only recently emerged [28, 111].

To directly investigate why AI is not commonly used in ICS and to explore new opportunities for adopting AI to protect ICS, we conducted 18 semi-structured interviews with practitioners who work on monitoring, operating, and securing ICS in various industries and roles. Based on

[1]We use the terms "ML" and "AI" interchangeably in this chapter, as we found that some practitioners preferred the term "AI" in interviews.

these interviews, we identify tasks commonly performed for alarms in ICS as part of an alarm workflow. Alarm workflows often involve defining rules to detect anomalies, reading real-time data from the ICS and raising alarms, responding to alarms, and modifying alarm rulesets for efficiency and safety. In this work, we answer the following research questions:

- **RQ6.1**: What types of data and systems are used for alarms in ICS, and how suitable are they for AI?
- **RQ6.2**: What human tasks are performed with alarms in ICS, and how can AI support them?
- **RQ6.3**: In organizations that operate ICS, what logistical and cultural factors hinder AI adoption?

At the process level, detecting attacks is intertwined with detecting non-malicious anomalies. Hence, our investigation by necessity examines participants' perspectives on the detection of all anomalies, and not just those caused by attacks. One particular challenge in answering these research questions is that practitioners working with ICS typically do not have experience with AI, which limits their ability to provide details on how AI could be used to protect ICS. Thus, in our interviews, we first performed a needs assessment of current practices for alarms in ICS. We asked practitioners how they design, use, and maintain systems to raise alarms; how they coordinate alarm response; and what challenges they experience with alarms. Since practitioners in ICS typically do not use AI, we next asked participants about their perceived benefits and barriers to adopting AI in ICS.

Although most prior work that proposes AI for ICS security focuses on centralized AI models for detecting anomalies [41, 66], our findings suggest that other use cases for AI are likely to be more promising in practice. We found (**RQ6.1**) that data and systems for raising alarms are often not centralized, but historical data from alarms is. We also found (**RQ6.2**) that practitioners often struggle with tasks beyond detecting anomalies, such as diagnosing alarms and managing alarm rulesets. We therefore propose creating tools for diagnosing and managing alarms on centralized, historical data. Furthermore, we found (**RQ6.3**) important cultural barriers to deploying and using tools in ICS, such as general skepticism towards adopting new technology. We therefore recommend ways to navigate these barriers; for example, given the importance of trust in ICS, we recommend that tool designers build trust with practitioners by interactively demonstrating how AI-based tools make predictions.

6.2 Participants and methodology

We interviewed 18 practitioners who work with ICS in multiple industries and roles. In this section, we describe: how we recruited participants (Section 6.2.1), how we conducted semi-structured interviews (Section 6.2.2), how we analyzed interview responses (Section 6.2.3), our consideration of research ethics (Section 6.2.4), and limitations of our methodology (Section 6.2.5).

6.2.1 Participant recruitment and demographics

Our target population is practitioners who work on safeguarding and securing ICS by performing alarm workflow tasks or by managing or supporting alarm workflows. To recruit participants from this population, we used purposive sampling. We directly contacted individuals in our professional networks; we advertised on ICS security mailing lists; we emailed utility providers with public contact information; we posted flyers on LinkedIn and X (formerly Twitter); and we sent direct messages on LinkedIn to people whose roles matched ICS-related keywords such as "SCADA" or "Control System."

In our initial recruitment text, we used the terms "anomaly detection" and "security," and we failed to recruit participants; multiple organizations responded that they did not perform anomaly detection or did not have any security-relevant topics to discuss. Given the sensitive nature of cyber-attacks on critical infrastructure, we believe that participants were unwilling to discuss these topics or believed that they were not relevant to them. We then updated our recruitment text to state that we were interested in "monitoring tools" and "alarm response." We were then able to successfully recruit study participants and discovered that, in fact, they do use systems to detect anomalies and acknowledge that cybersecurity concerns can impact alarm response. This experience serves as a useful lesson that, when interacting with practitioners who work with ICS, using appropriate terminology is important.

Potential study participants then filled out a screening survey, which asked about their industry, day-to-day tasks, and experience with ICS, cybersecurity, and AI. We screened participants for those who demonstrated experience with operating, managing, or developing tools for alarm workflows. We also limited study participants to those located in the USA, although some participants reported on prior experiences from working in other countries.

Table 6.1 provides an overview of study participants' demographic information. Participants worked for two different types of organizations: *plant owners*, organizations that operate an ICS, and *vendors*, organizations that support alarm workflows for multiple ICS. Of the participants who worked for plant owners, six participants worked for local municipalities across five different US states.

Similar to challenges reported in related work [9, 134], we found it difficult to recruit operators who worked as the first point of contact in alarm response for an ICS. Our interviews revealed that ICS are often monitored 24/7 by operators who are often overworked. Thus, these operators likely could not provide the time to participate in an interview for research purposes. Although we could not recruit operators who worked as the first point of contact in alarm response at the time of recruitment, the participants in our study manage these operators, perform secondary alarm response tasks, or worked as operators in the past. Thus, participants were able to discuss operator perspectives through second-hand experience and prior first-hand experience.

As described in Section 6.2.3, we iteratively performed qualitative analysis after establishing an initial list of codes. We determined that recruitment was complete once we observed that no new qualitative codes directly pertaining to our research questions emerged (inductive thematic saturation [105]).

ID	Industry	Role	# Years Exp.		
			OT	Sec.	AI
P1	Electric (solar)	Manager	10	0	10
P2	Oil & Gas	Engineer	1	0	0
P3	Electric (grid)	Engineer	12	4	6
P4	Water	Manager	10	0	0
P5	Water	Engineer	2	0	0
P6	Water	Manager	15	10	0
P7	Oil & Gas	Manager	18	0	0
V8	Electric (gen.)	Consultant	50	24	0
P9	Manufacturing	Engineer	13	0	0
V10	Electric (gen.)	Engineer	20	15	4
P11	Electric (grid)	Manager	10	10	0
V12	HVAC	Engineer	5	2	0
P13	Oil & Gas	Engineer	4	0	0
P14	Manufacturing	Engineer	7	0	0
V15	Electric (grid)	Engineer	25	10	0
V16	Oil & Gas	Consultant	8	13	0
P17	Oil & Gas	Engineer	16	14	4
P18	Water	Manager	35	10	0

Table 6.1: Demographic information for the 18 participants in our study: their industry; their role; and their years of experience with operational technology (OT), cybersecurity (Sec.), and AI. 13 participants primarily worked for a plant owner that operates one ICS (marked "P"), while five participants primarily worked for a vendor or as a consultant that supports multiple ICS (marked "V").

6.2.2 Interview methodology

We conducted 60-minute, semi-structured interviews over Zoom. Participants filled out a consent form before starting the interviews. We recorded interviews with participants' consent or took notes if participants did not consent to recording.

We divided our interview into four sections. In part I, we asked the participant about their professional background and day-to-day responsibilities. In part II, we asked about current practices for alarms in ICS: how data is collected and alarms are raised, how alarm response is performed, and how these processes are managed. In part III, we asked about vendor tools and how they are adopted in ICS. Finally, in part IV, we asked participants about their perceptions of using AI in ICS. When interviewing participants who worked for vendors, instead of asking about the practices of a single ICS, we asked about common practices and trends observed from working with clients. To ensure question clarity, we performed a pilot test of our interview script with two researchers (who are not directly involved with this work) with experience in human-subjects research in ICS contexts. Our interview script can be found in Appendix B.

6.2.3 Analysis methodology

To prepare our data for analysis, we transcribed interview recordings using an automatic transcription service. We edited all transcripts and notes for correctness and anonymity by removing specifically identifying information related to people, places, and companies before deleting the

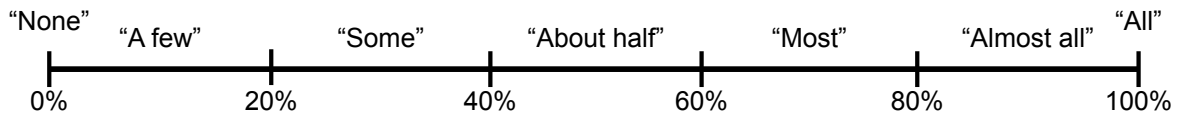


Figure 6.1: When reporting the proportion of participants in results, we use qualitative terms instead of raw counts or percentages. We convert percentages to terms based on the scale shown, using a mapping similar to that of prior works [34, 51].

original recordings.

To analyze our interview data, we used inductive thematic analysis [21]. After completing the first 16 interviews, two researchers iteratively and independently reviewed each transcript, creating a list of initial codes that captured concepts relevant to our research questions. Once all interviews were complete, the two researchers met to merge codes and group related codes into themes, producing our initial codebook. We then refined the codebook using an iterative, consensus-based approach to ensure that the two researchers shared a conceptual understanding of the codes and could apply the codes consistently. The two researchers independently coded two transcripts using the initial codebook, met to identify disagreements and update code definitions, and resolved all discrepancies in the codes. Using the refined codebook, both researchers independently coded four more transcripts, met to discuss codes, and found no substantial disagreements on the definitions or usage of the refined codes. After reaching consensus on the codebook and its application, one researcher subsequently coded the remaining interviews. To ensure consistent application of codes, another researcher reviewed these codes for correctness. Our final codebook and code counts are in Appendix C.

Following suggested practices in qualitative HCI research [86, 95], we do not compute inter-rater reliability metrics since the goal of our study is to identify emergent themes rather than to quantify the frequency of topics. We ensured consistency by refining the codebook when disagreements were found in double-coded interviews and reviewing each single-coded interview. Furthermore, we do not report the exact counts of participants when discussing results since our primary findings are qualitative and we gathered perspectives from a diverse but not necessarily representative sample of practitioners. We instead follow a common methodology from prior work and use qualitative terms to illustrate the prevalence of themes [34, 51], by mapping percentages to terms as shown in Figure 6.1.

6.2.4 Ethics

Our study was approved by our institution's Internal Review Board (IRB). Participants were compensated $60 USD for completing interviews, a similar rate to prior work that interviews domain experts [65, 83, 89]. Following best practices, we minimize participant harm by obtaining informed consent, anonymizing transcripts, and asking participants not to share confidential information about their organization or role [15]. We follow these principles to protect participants' individual privacy, to protect participants from potential repercussions from their employer, and to protect their employers from increased risk by disclosing sensitive information about cybersecurity practices.

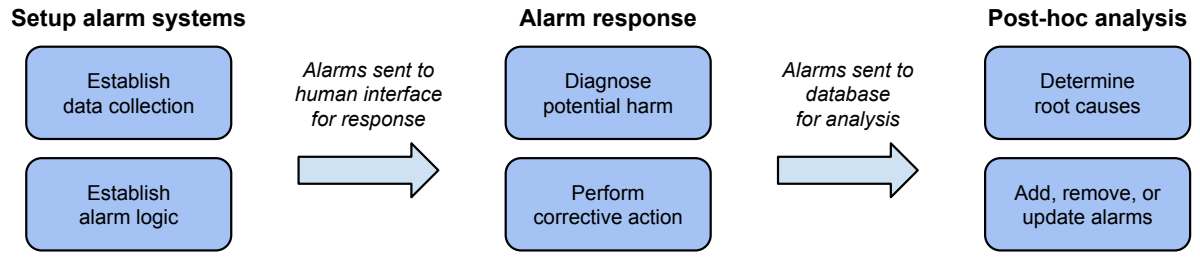


Figure 6.2: We found that alarms are managed through a set of processes in an alarm workflow. We show the different tasks in alarm workflows, categorized into three stages: (i) setting up systems for alarms, (ii) responding to alarms, and (iii) analyzing alarms post-hoc to determine root-causes and update alarm conditions.

We also weighed the benefits of publishing our results, which reveals security practices in ICS, against the risks of releasing more information to adversaries. We concluded that the risks were minimal since ICS are already commonly attacked [117], and industry surveys already disclose that AI-based tools are not commonly used in ICS [26].

6.2.5 Limitations

The responses of participants we interviewed may not fully represent the perspectives of current, first-response ICS operators, since they do not currently serve as the first response to ICS alarms. Some participants in senior roles had not worked as operators for several years, and their responses may not fully represent all operators due to organizational communication barriers and changes in the industry. Furthermore, all study participants were based in the USA, which may limit the applicability of our findings to other countries.

We describe participants' suggestions for AI adoption in Section 6.4. Most participants lacked experience with AI, and so potential misconceptions about the requirements and capabilities of AI may affect the feasibility of their suggestions.

6.3 Results: Current practices for alarms in ICS

In this section, we report our findings for how ICS operators use systems for alarms and perform alarm workflow tasks. Since AI is not commonly used in ICS and most practitioners who work with ICS lack experience with AI [26] (including the participants of our study, shown in Table 6.1), we use an indirect approach to investigate our research questions by first asking about current practices for alarms in ICS. As a pre-requisite for answering **RQ6.1**, we ask participants about systems that read data from an ICS and raise alarms (Section 6.3.1). To support our investigation of **RQ6.2**, we ask participants about human tasks performed for alarms (Section 6.3.2) and challenges with performing these tasks (Section 6.3.3). Finally, for our analysis of **RQ6.3**, we ask participants about ICS-specific factors that affect alarm workflows (Section 6.3.4) and adopting vendor tools in alarm workflows (Section 6.3.5). Although the individual processes used for each ICS vary, we identify common processes for alarms across ICS, and we show a categorization of these processes in Figure 6.2.

6.3.1 Systems for raising alarms

Anomaly-detection systems, whether AI-based or rule-based, rely on real-time data from the ICS, so understanding how this data is collected is critical to understanding how AI can be used for alarms in ICS.

What devices and systems are used? Referring to the devices described in Section 2.1, almost all participants who worked for plant owners use PLCs. Almost all participants also use a level 2 system to coordinate multiple PLCs—some participants use a DCS and some participants use SCADA. About half of participants report using one or more control rooms, which are centralized locations for operators to monitor ICS and delegate alarm response. In contrast, some participants reported that their organization does not use control rooms; operators instead interact with PLCs through co-located HMIs, which are distributed across the industrial process. Finally, some participants report that their organization uses a data historian, which stores process and alarm data in a central database for post-hoc analysis. We describe these post-hoc analysis tasks in Section 6.3.2.

Which organization manages these devices? Although plant owners use various devices for monitoring and controlling industrial processes, they do not necessarily program or manage these devices. Some participants work for plant owners who rely on vendors to manage their devices; a few participants reported that this was common in their industry. In contrast, some participants work for plant owners that employ their own staff to program and manage their devices. We describe how vendors affect alarm workflows in Section 6.3.5.

> *A lot of them actually contract out their PLC, SCADA, networking, some of the more high level stuff. Actually, almost every city that I know does that. –P6*

What behaviors are alarms used for? All participants reported using alarms to detect anomalies in process values, although the reasons for detection varied. Most participants mentioned safety: process alarms ensure the safety of the physical process or the process equipment. Some participants mentioned non-critical reasons, such as to ensure adequate production or to ensure that regulations are being met. Some participants also reported using alarms that were not related to process values; these include alarms to detect component failures (e.g., an unresponsive PLC), cybersecurity events, or physical security events. Finally, some participants reported special types of alarms, such as informational alarms, maintenance alarms, and alarms written for specific, prior incidents.

How are alarms defined? Most participants reported that alarms were defined by rules; alarms were raised if a process value exceeded an upper or lower limit. About half of participants additionally reported that more complex logic was used, such as using rates of change or combinations of rules.

> *We have combined conditions to generate a new alarm or suppress some alarms. For example, if we trip something, we don't need to see alarms from every downstream device. –P17*

Where is alarm logic implemented? Since various types of devices are used in alarm systems, the placement of alarm logic also varies. Most participants reported that alarm logic was written in PLCs; these alarms only evaluated conditions based on data available to the PLC. Some participants reported that alarm logic was instead written in DCS or SCADA; these alarms were often more complicated and required data from multiple PLCs. Finally, most participants also reported that alarms were written directly into level 0 devices (e.g., sensors), referred to as "safety systems." Safety systems can perform commands (e.g., shutting off a valve) without human involvement or inter-device communication and often send alarms to higher-level systems for diagnosis.

Where are alarms displayed? Although alarm logic is written into devices at levels 0–2, alarms are not necessarily displayed in these devices. Some participants reported that alarms from PLCs were forwarded to DCS, SCADA, or HMI. Devices in levels 0–1, such as safety systems and PLCs, often lack an operator interface, so alarms from these devices are forwarded to human operators in a level 2 system.

> *The SCADA is pulling from the PLC and if there's an alarm, it's going to display that on the SCADA itself. –P14*

However, a few participants reported that alarms were not always forwarded. In some cases, alarms could be raised without visibility to a level 2 system.

> *Some of those alarms will not go to SCADA, at least not directly. [. . .] If a relay causes a breaker to open, the relay knows why it opened, SCADA would see the breaker open, but if you were looking at your SCADA logs, you would never get any indication as to why that breaker opened. –V15*

Takeaways for AI. Participants reported using a variety of devices for raising alarms, which can range in data availability and computational power. Additionally, some plant owners rely on vendors to manage these devices. These differences make it unclear who would manage an AI-based tool for alarms, and where it should be deployed in an ICS. We also found that alarm conditions use logic and implementations that may differ from AI-based anomaly detection. For example, alarms use various data modalities (e.g., network and process data) and custom logic that may not correspond to learnable patterns in a dataset.

6.3.2 Human tasks in alarm workflows

We asked participants about human tasks performed in alarm workflows, such as responding to alarms and managing alarm rulesets. We investigated if and how humans performing these tasks could be supported by AI.

Who responds to an alarm? Most participants reported that an on-site operator is the first to respond to an alarm. For a majority of alarms, operators are able to respond appropriately, either by performing the required remediation action or by acknowledging the alarm as a non-issue. If the proper response could not be determined or performed, operators would then escalate to higher levels of authority for help. A few participants who worked for plant owners also reported

contractual agreements with vendors for alarm response. A few participants reported that they served as the second or third point of alarm response for an ICS.

The alarm does not go away or it gets worse, then you escalate up to the next line. I'll definitely get involved in troubleshooting and trying to figure out stuff like that. –P13

How is the response to an alarm determined? Most participants reported using structured protocols to remove ambiguity in alarm response. About half of participants reported that alarm response was dictated by pre-defined categories for alarms.

The operator knows if an alarm comes in color red, you have to address that right away. If it comes in this color, you just call this person. If it comes in this color, you don't even have to do anything. –V15

However, structured protocols do not cover all cases of alarm response. Most participants reported that alarm diagnosis sometimes relies on operator expertise—operators diagnose alarms by correlating them with auxiliary data in an intuitive, unstructured process. Despite using structured alarm response protocols and auxiliary data sources, less-experienced operators can struggle with alarm diagnosis.

I think that's probably our greatest challenge: training the staff that's still fairly new and still learning the processes what the appropriate level of response is. –P18

How are alarms analyzed post-hoc? In cases where the real-time alarm diagnosis and response was incorrect, organizations analyze historian data for root-cause analysis. About half of participants reported that their organizations have specific teams or roles for asynchronous, post-hoc alarm analysis.

Another team is looking through our alarm history and identifying where we have ongoing issues or where we didn't respond to something the way we should have. –P7

A few participants reported analyzing alarms post-hoc for alarm management. Participants mentioned the ANSI-ISA 18-2 standard on alarm management [57] and discussed alarm management tasks such as reviewing and updating alarm conditions to reduce operator fatigue. Organizations perform alarm management by reviewing historical alarm data through regular, cross-functional meetings to ensure that existing alarms are effective. If needed, alarms are added, removed, updated, or re-categorized.

We have alarm management expectations. Once a week, as an engineering team, we meet and review all of the alarms that came in over the last week, and try and figure out, was this a good, useful, real alarm? [. . .] And you can make changes to the alarm set points or things like that. –P13

Some participants did not explicitly mention "alarm management" but reported other processes for managing alarm rulesets. Some organizations regularly test alarms, some organizations use tools to analyze alarm data, and some organizations allow operators to update alarms themselves.

We use an alarm analysis tool. [. . .] It's doing SQL queries to find repeat offenders or numbers per hour. –P7

Finally, a few participants explicitly suggested that, since their historian data was centralized and labeled, an AI-based tool could be trained to help with alarm analysis.

So we now have 1000s upon 1000s of examples of: the data was doing this at the time, it led to this root cause analysis, and it led to this action. And I think that's something that we can begin to look at applying deep learning algorithms to, because we have the necessary data to start training that. –P1

Takeaways for AI. Although most alarm response is performed through structured protocols, we found that some tasks for alarms rely on intuition and expertise, such as alarm diagnosis and alarm management. These tasks involve post-hoc analysis of alarm data and already include automated processes and tools, which suggests openness towards using AI for these tasks.

6.3.3 Challenges with alarms

In this section, we describe common challenges reported by participants when working with alarms. Participants reported challenges across alarm workflows, from correctly diagnosing alarms to deciding if alarm rulesets were effective.

Nuisance alarms. Most participants discussed "nuisance" alarms, which do not correspond to genuine harm, do not convey useful information, and are not actionable. Operators are burdened with recognizing and acknowledging nuisance alarms to remove them from user interfaces, and some participants reported that it can be difficult to distinguish nuisance alarms from genuine alarms. When asked about the frequency of alarms at the ICS they worked with, participant responses ranged from two to 250 alarms per hour. A few participants mentioned a growing awareness about alarm fatigue and its potential to increase the likelihood of operator errors.

Processes may have some variances that are going to cause nuisance alarms. Eventually, they run into, "Oh, I've ignored too much. Now I've got myself into a hole." So we try to be very judicious about what we what we alarm about. –V10

Some participants reported using strategies to mitigate nuisance alarms. These include alarm management (described in Section 6.3.2), defining guidance for alarms (e.g., "alarm philosophy"), using tools to suppress nuisance alarms, and using tools to identify nuisance alarms. One participant described their experience reviewing and updating alarm rulesets at their organization, reporting that it required significant amounts of time and labor.

We looked at every single alarm that we have, and then went through and wrote troubleshooting guidance, and then challenged if you need the alarm, and then what the alarm point should be. And that was a significant year and a half of, at least 10 hours a week. –P13

Challenges in alarm diagnosis. Even if an alarm is determined to be genuine, alarm diagnosis can still be challenging. Some participants reported limited access to data as a challenge for alarm diagnosis. In some cases, due to missing coverage in monitoring or logging, operators would need to physically travel to properly diagnose an alarm, limiting their ability to respond quickly.

It could be 10 minutes to an hour or two, in some parts of the country, of drive time before you even get eyes to see what's going on. –P11

A few participants reported that too much information could overwhelm operators and hinder diagnosis. Providing data to operators requires a balance between sending sufficient diagnostic information and avoiding information overload.

We weren't rationalizing what we were bringing in, trying to bring everything back that we could. And so getting any value, really, out of those alarms was difficult. –P7

Takeaways for AI. Participants reported challenges with alarm workflow tasks, particularly for alarm management and alarm diagnosis. We suggest that AI could help mitigate these challenges and improve these tasks, and we describe participant perspectives on using AI for alarm workflow tasks in Section 6.4.2.

6.3.4 Factors that affect alarm workflows

In this section, we describe factors specific to ICS that affect how alarm workflows are designed and executed. We identify opportunities for improvement and pitfalls that should be avoided when using AI for alarm workflows in ICS.

Limited resources in ICS. Plant owners often have small OT and cybersecurity budgets [97], introducing constraints on technology and personnel. Most participants discussed how these constraints made alarm workflow tasks more difficult.

About half of participants who worked for a plant owner reported personnel constraints, including understaffed teams and limited technical skills. These constraints limited plant owners' abilities to improve their processes for alarms, as they were occupied with critical operations and alarm response.

I've got one assistant now and I feel like I could keep three assistants busy. There's this triage of things that would be valuable to do versus things that are urgent. –P4

Participants also reported challenges with managing or using technology in ICS. Some participants reported challenges with managing OT networks, which caused problems with data visibility and trust. A few participants reported concerns with device capabilities, in terms of both computation and networking. A few participants also reported challenges with updating and replacing ICS devices. Given these technical constraints, some participants expressed doubt that adopting advanced tools for alarms would help.

They wrote a little report on it. Here's some potential alternatives. [...] But the alternatives may not work any better because the problem may really be that our network is unstable. –P4

Perceptions of safety in IT vs OT. About half of participants reported tensions between informational technology (IT) and operational technology (OT) professionals; such tensions have been studied in prior work [44, 140, 146]. We found that these tensions can add friction to alarm workflow tasks when IT professionals interact with OT systems.

People that came up from the IT side, their mental model of digital systems, it's not working, I'll just reset it. I can't do that if I'm running a power grid. [...] Because IT capabilities are continuing to get pushed closer to physical systems, people are coming from the IT side. That's where I think a majority of the mental model change needs to happen. –P11

We found that safety, rather than security, was the focus for most participants. About half of participants acknowledged that ICS security was relevant to their work, since attacks on ICS could affect systems used for alarms. However, some participants reported that their organization does not use OT security tools, instead reporting that they believed security was the responsibility of IT.

> *A lot of the industrial control equipment, your PLCs and stuff, are pretty vulnerable. If somebody can get to them over a network, we feel like we're already essentially screwed. So the emphasis is more on the IT side and keeping those things off the network. –P4*

Of the participants who work for vendors, almost all reported that concerns about security emerged in their discussions with plant owners, which affected their practices for monitoring and connectivity.

> *Some people will not connect [relays] to SCADA because there's a worry if your SCADA system had a problem, mainly a cyber problem. [. . .] By isolating it completely, you've got another level of confidence. –V15*

Government regulations. In some industries, government regulations mandate which monitoring, alarms, and security practices must exist. Of the four participants who worked for plant owners in the water industry, half reported that they wrote alarms for certain process values because they were required by regulation. Most participants who worked in the electric industry reported on how NERC CIP (North American Electric Reliability Critical Infrastructure Protection) standards [123] impact their systems for alarms. In some cases, regulations would cause plant owners to consider the tradeoff between increased monitoring and the additional cost of required compliance.

> *Does the device have connectivity? If it does have connectivity then you've got to follow these extra rules. But if you can say the device has no connectivity, then you don't have to answer those next questions. –V15*

A few participants also reported on differences in regulation across industries. One participant commented specifically on how regulation could affect AI adoption, stating that NERC CIP would serve as a barrier for adopting AI in the electric industry. Developers of AI-based tools will therefore need to meet NERC CIP regulations for adoption in the electric industry.

> *If it's like oil and gas, where it's a well-known, well-documented process, they're more likely to embrace AI type stuff. If it is a power-gen aeroderivative turbine, a lot of times it's going to be very human centric, because they cannot document the AI-ness for NERC CIP. –V10*

Takeaways for AI. Several factors can limit how alarm systems are used and how alarm workflow tasks are performed: difficulty updating devices, mismanaged OT networks, understaffed and undertrained teams, cultural friction between IT and OT professionals, and restrictive government regulations. Tool designers must successfully navigate these barriers for effective adoption in ICS alarm workflows.

6.3.5 Adopting vendor tools in ICS

In Section 6.3.1, we reported that some plant owners have contracts with vendors for monitoring and alarm response. This suggests that vendors could potentially be the driving factor towards

adopting AI for alarm workflows in ICS. Since most existing tools are not based on AI, we asked participants about how vendor tools in general are evaluated and adopted into ICS. We discuss participant responses when asked specifically about adopting AI in Section 6.4.

What barriers hinder adoption? Almost all participants reported that adopting vendor tools in ICS is difficult. Participants reported concerns with vendor tools such as high cost, requirements for skilled personnel, lack of customization, and a lack of trust.

You need personnel, you need people trained on new technology, if you want to put in new technology. And that was a problem with certain brands. –P5

Some participants reported that plant owners prefer to keep their systems homogeneous under a single vendor; many vendors provide solutions across the ICS stack for control logic, alarms, and OT networks.

There's better stuff out there that we could be using, but our investment in <Brand A> versus what it would cost and the amount of work it would take me to switch over to <Brand B> is pretty unfeasible. –P6

What values are important for adoption? Participants reported that quantitative criteria were not frequently used to evaluate vendor tools. Some participants reported that metrics like accuracy or F1-scores, commonly used in ICS anomaly detection research, were not meaningful to them. Instead, almost all participants reported that tools were evaluated qualitatively. Values such as brand reputation, positive discussions with vendors, and positive recommendations were reported as most important. A few participants mentioned vendors that provide AI-based tools for ICS, but reported that they were mostly not yet trusted by the industry.

You could do all of this testing to say, what's the percentage of identified anomalies versus unidentified anomalies? There's things like that. Yeah, not at my level. –P8

Who decides to adopt new technology? Finally, we found that who made tool acquisition decisions varied. Some participants reported that a cross-functional team decided if a vendor tool was adopted, whereas some participants reported that practitioners who work with ICS, including the potential end-users, were often excluded from these decisions.

Some of that decision making, sadly it's going to be some really slick talking salesman talking to an engineer that will never work with that SCADA system. And then the people that use that SCADA system aren't going to have a say in what they're using. That's just the way it is. –P6

Takeaways for AI. Tool adoption in ICS is heavily based on trust and reputation. Detection-based metrics, which are used to compare AI models in research, are not used to motivate adoption. Tool designers and vendors should therefore develop new metrics that better convey trust and build their reputation with ICS insiders.

6.4 Results: Perceptions of AI

In the final part of our interview, we asked participants directly about their perceptions of using AI in ICS. We allowed participants to respond based on their own conceptual model of AI, but

since most study participants had no experience with AI (shown in Table 6.1), we acknowledge that these suggestions may not necessarily be practical.

We describe participants' conceptual models of AI (Section 6.4.1), perceived benefits of adopting AI (Section 6.4.2), and perceived barriers to adopting AI (Section 6.4.3). Combined with our findings of current practices and challenges (Section 6.3), these responses reveal opportunities and challenges for using AI to support alarm workflows in ICS and guide our recommendations in Section 6.5.

6.4.1 Conceptual models of AI

Given the lack of participant expertise in AI, we first establish and describe participants' mental models of AI. In this study, we define AI to be any technology that uses historical data to learn patterns and makes predictions on new data, such as neural networks [3, 69, 125], large language models (LLMs) [120], SVMs [10, 108], and confidence intervals [30]. We found that all participants demonstrated some understanding of what AI was and its capabilities. Most participants mentioned a specific model, such as LLMs, neural networks, or linear regression models. The remaining participants did not mention a specific model, but correctly referred to AI as technology that learns and makes predictions from data.

6.4.2 Perceived benefits of adopting AI in ICS

Some alarm workflow tasks involve processing large amounts of data and seeing complex patterns. Based on their understanding of AI and its capabilities, most participants thought AI was well suited for such tasks.

Making alarm workflow tasks more efficient. About half of participants believed that using AI would save operators' time. In Section 6.3.4, we reported that ICS operations teams are often understaffed and fatigued, and AI could help reduce this burden. About half of participants believed that AI could outperform humans at some tasks, such as seeing complex relationships in data or avoiding distractions.

> *If there was some kind of machine learning, it might notice things like, these alarms happen when you're running that motor over there which you wouldn't think is related. –P4*

A few participants who worked for plant owners reported trying LLMs for alarm diagnosis, outside of their organization's established alarm workflows. These participants tried using LLMs to diagnose previously resolved incidents. Participants reported positive experiences with LLMs, which were able to correctly diagnose the incidents and strengthened their belief that AI could provide value in similar circumstances.

> *In the prompt, I put the question: we know that a certain equipment was tripped, so we asked to find why it's tripped. . . . We spent like a whole bunch of hours, but this model for 10 seconds, and having 30% of information we had, gave us the cause of the trip. –P17*

6.4.3 Perceived barriers to adopting AI in ICS

In Section 6.3.4, we described barriers to adopting vendor technology in ICS. Echoing these findings and based on their understanding of what AI is and its requirements, participants reported their perceived barriers to adopting AI in ICS.

Limited compute and data availability. Participants reported concern that using an AI-based tool would require data and computational infrastructure beyond their organization's capabilities. About half of participants reported that data availability was a barrier to adopting AI because their data quality is too poor, their data is not sufficiently labeled, or that there would be IP issues with AI-based tools accessing their data.

> *The mistake would be: Hey, we have this incredible system that can detect all these problems. [. . .] But no site manager wants to set them up with the massive data requirements to make that product run. –P1*

Limited people with AI expertise. Some participants reported concerns with finding and hiring the specialized staff required to use AI-based tools. On the other hand, a few participants suggested that adopting AI could help with staffing issues, suggesting that using AI could attract a younger, more skilled workforce to ICS.

> *You need the person who knows how the process is controlled, how alarms are generated, and how an LLM works, which is not easy to find. –P17*

Low trust in AI. Most participants reported that, since ICS are so critical, they needed tools that were trustworthy. About half of participants reported concerns with AI's lack of transparency or tendency to make errors. These concerns made it difficult to convince plant owners to adopt AI-based tools.

> *The customer, he doesn't trust [AI] in control at all. [. . .] If you give them a method which has some problem with robustness, it can cost him millions if he got to shut down activity. –V12*

When asked about ways to improve AI in general, some participants recommended that AI reduce overconfidence and some participants recommended that AI be more transparent.

> *There are many wrong ways and there's a few good ways to implement [AI], and the good ways all involve: Here's how it works, here's what it's looking at, breaking it down, and putting a lot more transparency behind it. –P11*

Participants also suggested methods to build trust in AI-based tools before adoption: allowing practitioners to interactively test with real data, establishing benchmarks for AI-based tools in ICS, adding explanations to predictions, or ensuring that tools were only used as an assistant to human operators.

> *Around alarms, I would say, starting as an assistant, because there's no way it's going to have all of the experience and all of the information necessary to be 100%. But I think it could do a really good job of helping you. –P7*

Stage of Alarm Workflow	Available Input Data	Prediction Task(s)	Anticipated End-user
Anomaly detection	Real-time process data	Detect anomalies in real time	Operator
Alarm diagnosis	Alarm with context	Suggest real-time alarm response Predict root cause of alarm	Operator Lead operator
Alarm management	Set of prior alarms and actions	Fix a misconfigured alarm Improve alarm ruleset	Engineer Manager

Table 6.2: We suggest opportunities for AI-based tools to support different alarm workflow tasks. For different stages of an alarm workflow (shown in Figure 6.2), we list the expected input data available for AI, potential prediction tasks for AI, and anticipated end-users who would use these predictions.

6.5 Analysis and recommendations

In this section, we answer our research questions and provide recommendations for adopting AI to support alarm workflows in ICS. We discuss how AI could use existing data and systems for alarms (**RQ6.1**, Section 6.5.1), how AI could support humans in alarm workflow tasks (**RQ6.2**, Section 6.5.2), and how to navigate barriers that hinder AI adoption (**RQ6.3**, Section 6.5.3).

6.5.1 Deploying AI in systems for alarms

What data and systems are used for alarms in ICS, and are they suitable for AI? (RQ6.1) We found that the systems and practices for alarms vary across ICS (Section 6.3.1). Alarms operate on process data, network data, or data from cybersecurity tools; alarm logic is programmed into sensors, PLCs, SCADA, or DCS; and alarms can be forwarded to and displayed on various devices. Systems for alarms also vary in the degree of vendor involvement. These differences suggest that, although most prior work that proposes AI-based ICS anomaly detection assumes that all process-level data and compute are available for inference [66, 75], deploying a centralized AI model with real-time access to all process features is unlikely to be feasible for most ICS.

We found that organizations that work with ICS often centralize and store historical data, which may make it suitable for AI. With historical data, participants reported performing data analysis tasks, labeling data, and using automated tools that suggest a readiness for adopting AI (Section 6.3.2).

Recommendation: Consider the varying availabilities of data and infrastructure in ICS when deploying AI. Effectively using AI-based tools to support alarm workflows in ICS requires considering how an AI-based tool would be deployed: what data will be used for training and inference, where inference will be performed, and whether tools will be managed by plant owners or vendors. Designers of AI-based tools need to consider that it is likely that they would be training AI models on only a subset of process-level features.

In domains other than ICS, some deployment models of AI may already fit with the existing systems and practices of some ICS, such as decentralized algorithms for model training and inference (e.g., federated learning across IoT devices [93]) or accessing AI-based tools through a vendor (e.g., AI as a service [101]). Future work should investigate and develop deployment

models for AI that match the varying requirements of ICS environments.

Recommendation: Acknowledge the impact of government regulation on AI adoption. We found that differences in government regulation impact alarm workflow practices (Section 6.3.4). For example, in the electric industry, NERC CIP regulations impose security and documentation requirements on connected devices and collected data, which causes some plant owners to choose not to connect certain devices to networks. Developers of AI-based tools for the electric industry will similarly need to comply with NERC CIP regulations. Thus, adopting AI for ICS in industries with stricter regulations (e.g., electricity) will be more difficult than in others (e.g., oil and gas). Researchers and designers of AI-based tools may find new opportunities by focusing on deployment in industries with more flexibility for AI adoption.

6.5.2 Using AI to support alarm workflow tasks

What human tasks are performed for alarms in ICS, and can AI support them? (RQ6.2) We found that several tasks are performed for alarms in ICS (Section 6.3.2), as shown in Table 6.2. In particular, beyond real-time anomaly detection, humans analyze alarms post-hoc for alarm diagnosis and alarm management. Most prior work in AI for ICS security focuses on AI-based anomaly detection [46, 75, 85, 132], but participants reported challenges with alarm diagnosis and alarm management, which rely heavily on intuition and expertise (Section 6.3.3). Participants themselves also suggested that AI could support these tasks (Section 6.4.2). We therefore propose designing an AI-based tool to support alarm diagnosis or alarm management. Given an alarm (or set of alarms), an AI-based tool could help humans triage alarms, suggest potential remediation actions, or predict root causes; or given a larger (e.g., from the past month) dataset of alarm and response data, an AI-based tool could suggest alarms to be added, removed, or modified.

Recommendation: Design AI-based tools to assist humans, rather than act autonomously. As described in Section 6.4.3, we found that participants prefer that AI-based tools make suggestions rather than automate decisions. Furthermore, as described in Section 6.3.3, alarm diagnosis and alarm management are often performed post-hoc and are used to address rare and complex situations; a human-facing, AI-based assistant may be appropriate for these tasks where urgent action is not required. Researchers and designers of AI-based tools should therefore focus on interactively assisting humans. Prior work has explored methods to provide assistance through human-AI interaction [91, 144], and a promising area of future work would be to apply such methods to ICS alarm workflow tasks.

Recommendation: Design AI-based tools to produce unintrusive, actionable outputs. We found that nuisance alarms and operator fatigue often hinder alarm response, and diagnosing unclear and unactionable alarms is a common challenge (Section 6.3.3). We recommend that AI-based tool designers ensure that AI outputs are actionable (e.g., by using AI-based explanations to suggest actions with each prediction [6, 94]), and we recommend that AI-based tools

balance the desire to inform the user with giving the user the ability to avoid repeated notifications if they are judged to be incorrect (e.g., by allowing humans to make decisions without AI involvement [16, 22]).

6.5.3 Navigating barriers to AI adoption

What ICS-specific factors hinder AI adoption? (RQ6.3) We found that barriers from technology (e.g., limited device connectivity), personnel (e.g., insufficient training), and culture (e.g., tensions between IT and OT professionals) can limit alarm workflow design and execution in ICS (Section 6.3.4). Participants also reported reservations about adopting AI in ICS (Section 6.4.3), stemming from a general mistrust of new technology in ICS (Section 6.3.5). Vendors are said to fail at meeting the requirements of ICS by imposing high costs, not providing adequate customization, or not matching the culture of safety in ICS. To avoid making similar mistakes when deploying AI-based tools to protect ICS, we recommend ways for tool designers to navigate these barriers.

Recommendation: Design AI-based tools for the skill sets of practitioners who work with ICS. We found that alarm workflows involve multiple people with different tasks and specialties (Section 6.3.2). For example, alarm diagnosis could be performed by an operator who first sees the alarm, a manager determining the root cause of an alarm that was incorrectly responded to, or an engineer debugging the logic of a misconfigured alarm. AI-based tools should be designed for specific users performing these tasks, rather than generally for practitioners. Table 6.2 lists potential opportunities for AI to support different alarm workflow tasks.

Participants also reported that requirements on personnel would be a barrier to adopting AI-based tools (Section 6.4.3), and plant owners are unlikely to hire or train skilled end-users to use AI-based tools. Instead, AI-based tools should be tailored to the existing skill sets of practitioners working with ICS. Prior work in other domains has investigated how explanations of AI outputs can be modified based on levels of user expertise [24, 33], and a promising area of future work would be to apply such approaches to support practitioners in ICS.

Recommendation: To build trust in AI, focus on demonstrating transparency to users. We found that trust and reputation were more important than quantitative metrics when deciding to adopt new technology in ICS (Section 6.3.5). Practitioners who work with ICS often do not currently trust AI (Section 6.4.3), so AI-based tool designers must first build this trust. Participants suggested building trust by transparently demonstrating how AI-based tools make decisions. Furthermore, participants reported that OT professionals face challenges working with IT professionals, are weary of new technology being pushed into their environments, and can be excluded from tool-adoption decisions (Section 6.3.4). Thus, AI-based tool demonstrations should include practitioners who work with ICS and OT.

As a first step to build trust in AI, we recommend pilot projects that allow interactive testing of AI-based tools and focus on transparency of AI models. Prior work has developed interactive tools for prototyping AI-based tools by allowing users to modify data and observe changes in

predictions [137, 138], and future work should investigate whether such methods are effective for practitioners who work with ICS.

6.6 Summary

We investigated current practices for alarms in ICS and identified challenges and opportunities for AI to support these practices. After conducting semi-structured interviews with 18 practitioners who work on safeguarding and securing ICS in different roles, from performing alarm response to building tools for alarms, we identified opportunities to adopt AI-based tools to support alarm diagnosis and alarm management. Finally, we recommend ways for researchers and designers of AI-based tools to navigate barriers to adoption in ICS, such as considering AI models with access to only a subset of process-level features and interactively demonstrating model transparency to practitioners.

Chapter 7

Conclusion

The work described in this thesis proposes techniques and guidelines to make anomaly detection more effective for industrial control systems. Through a combination of qualitative and quantitative investigations, I identified various challenges to using machine-learning-based anomaly detection for ICS in practice and subsequently made recommendations towards mitigating these challenges. Across the work described in this thesis, a common takeaway is that, although general-purpose approaches can achieve state-of-the-art results in benchmark settings, these approaches can be made more effective by adapting their design and application to ICS domains. In our proposed approaches, I consider the selection of evaluation metrics, practical use cases, environment constraints, adoption values, and human factors that distinguish ICS anomaly detection from general detection tasks. I recommend that future work that proposes approaches for ICS security follow a similar strategy with ICS-specific design and adaptation. In summary, through a careful consideration of the values and constraints that make ICS anomaly detection unique, this thesis builds towards a future in which ICS and critical infrastructure are made more secure as the growth of the capabilities of machine learning and artificial intelligence continues.

Appendix A

Survey text used in Chapter 4

In this Appendix, we show the text used for our survey of ICS experts, which we report on in Section 4.5.

Part I - Background

1. What type of industrial control system (ICS) do you operate? (Free response)
2. What is your role in operating this ICS? (Free response)
3. Do you, or does your organization use an anomaly detection system to detect potential attacks on the ICS you operate? (Yes / No / Not Sure)
4. Do you have any current or prior experience working with anomaly detection systems? (Yes / No)
5. What method(s) does the anomaly detection system that you have experience with use to detect anomalies? (choose all that apply: Rule-based anomaly detection, Machine learning-based anomaly detection, Not sure, Other (please specify))
6. What kind of information does the anomaly detection system that you have experience with provide you, and what does the human interface look like? (Free response)
7. What aspects of the anomaly detection system that you have experience with are most useful to you when monitoring the system or responding to potential attacks? (Free response)
8. What aspects of the anomaly detection system that you have experience with are most challenging or confusing to work with when monitoring the system or responding to potential attacks? (Free response)

Part II - Simulated Output

For the last part of this survey, we describe a simulated attack on the Secure Water Treatment (SWaT) testbed, and outputs from a proposed, machine-learning based anomaly detection system. In this scenario, the anomaly detection system has detected an anomaly. In addition to alerting the operator to the occurrence of an anomaly, the system also reports a list of sensors and actuators that the model predicts to be the cause of the anomaly. Here is the output of the anomaly detection system for a simulated attack on SWaT: (Table A.1)

Table A.1: Sample output from the detector used in the survey of operators.

Feature	Current Value	Anomaly Score	Alert Level
DPIT301	19.59	10.11	HIGH
MV302	2.00	8.74	HIGH
P302	2.00	2.03	HIGH
FIT201	2.44	0.54	MEDIUM
P203	2.00	0.54	MEDIUM
P101	2.00	0.53	MEDIUM
MV101	2.00	0.49	MEDIUM
FIT101	2.67	0.48	MEDIUM
MV201	2.00	0.46	MEDIUM
P403	1.00	0.45	MEDIUM

Currently, the anomaly detection system is configured to show the top 10 most likely sensors and actuators to be responsible for the anomaly, out of 34 total sensors. However, it can be configured to show more or fewer sensors or actuators. The more sensors it shows, the more likely the sensor or actuator that is the root cause of the anomaly is in the list.

In this next part, we propose several alternative configurations of the system, with different numbers of sensors/actuators shown.

1. How useful would an anomaly detection system be if it showed 2 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale: not at all useful, slightly useful, moderately useful, very useful, extremely useful)
   • 70% • 40% • 20%
2. How useful would an anomaly detection system be if it showed 5 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale)
   • 50% • 30% • 10%
3. How useful would an anomaly detection system be if it showed 10 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale)
   • 40% • 20% • 5%
4. How useful would an anomaly detection system be if it showed 20 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale)
   • 30% • 10% • 5%
5. How useful would an anomaly detection system be if it showed all 34 sensors or actuators? (5 point Likert scale)
6. Please explain your responses above: do you think it would be helpful for the anomaly detection system to display more sensors and actuators with lower error rates, or fewer sensors and actuators with higher error rates? (Free response)
7. Which of these two options would you rather have, for this anomaly detection system? (Choose one)
   - The entire list of sensors and actuators, and anomaly scores and alert levels for all of

them

- A list of 10 sensors and actuators the anomaly detection model thinks are most likely to be the cause of the anomaly, which correctly identifies the root cause 95% of the time

8. If this anomaly detection system showed 10 sensors and actuators, what percent of the time does the anomaly detection system need to correctly include the cause of the anomaly for it to be useful? (0%-100%)
9. If you had an anomaly detection system like this, how would you use the information provided to find the root cause of the anomaly? How would you integrate it into your workflow? (Free response)

Appendix B

Interview scripts used in Chapter 6

The detailed questions used in our semi-structured interview script are provided below. When appropriate, we asked follow up questions to participants to encourage further elaboration.

Part I - Background information
Participant demographics:

- What type(s) of ICS do you work on?
- What is your job title?
- How many years of experience do you have with ICS/OT?
- Do you have a background in cybersecurity? Describe/how many years?
- Do you have a background in machine learning? Describe/how many years?

Specifics of ICS:

- What parts of the system are monitored and controlled by PLCs, SCADAs, etc?
- What types of data is collected, how is it collected, and how is it shown to a human?
- Could you describe your day-to-day responsibilities?

Part II - Questions about monitoring and alarms at your organization
Addressing issues in ICS:

- What are examples of anomalies or problems in the process that would raise alarms in your ICS?
- How are these issues detected?
- When an alarm is raised, what is your role in addressing the situation?
- How do you find out about these situations? (Are you watching an HMI? Do you get assigned work orders?)
- Who around you is involved in addressing these situations? (Are you managing people who address it? Are other people analyzing the situation and asking you to investigate?)

Anomaly detection tools:

- What tools or systems does your organization use to detect anomalies?
- Does your system primarily rely on rules, ML, or both?

- Does your operate on network information (e.g., packets), operations information (e.g., sensors and actuators), or both?

Responsibility around anomaly detection:

- Who sets up anomaly detectors, rules, or set points? You, other people in your org, outside vendors?
- What types of data sources are used in the monitoring system?
- Who is responsible for these sources? How many people in the org are in that role?
- What types of actions are commonly taken or expected, based on information from the monitoring system?
- Who is responsible for taking these actions? How many people in the org are in that role?

Responding to anomalies:

- How many alarms do you receive, and what proportion of them do you have to manually investigate?
- Do alarms need to be triaged? Is this process difficult?
- What starts an investigation of an alarm?
- How do you determine if the alarm in question is a false positive? Walk through your process in making this decision.
- Are any tools used to help with diagnosis, what information is provided by this tool?
- How is information from this tool used to determine if an alarm is a false positive?
- How is information from this tool be used to determine next steps for remediation?
- What other information would you need about an alarm to help determine if it is a false positive or not?
- Could parts of this decision be automated? Would you trust it?

General perspectives:

- What are the most helpful or useful aspects of the tools you use?
- What are the main challenges in detecting and debugging anomalous behavior?
- What are the main challenges in working with ICS in general?

Part III - Tool adoption

- How did your organization decide on the tools it uses for monitoring?
- Are there other roles in the organization who measure these things, experiment with tools, or deploy them?
- What properties or metrics were used to distinguish the tools you use from other alternatives?
- Are these decisions based on quantitative metrics? If no, then what is it based on?
- Once a tool is deployed, do you use any metrics or processes to ensure that it is useful for your organization?
- Suppose a vendor suggests that you try a new tool, what properties would it need to have for you to consider adopting it?

Part IV - Perceptions of AI

- What do you consider to be the pros and cons of using AI-based anomaly detection methods vs traditional methods for alarms in ICS?
- What improvements would need to be made to AI to make it more trusted by your organization?

Part V - Miscellaneous

- Is there anything else you wanted to tell us that we didn't ask about?

Appendix C

Qualitative codes used in Chapter 6

We provide our codebook in Table C.1 and Table C.2. The codes shown in Table C.1 are used to analyze the responses in part II of our interview script, and the codes shown in Table C.2 are used to analyze the responses in part III and part IV of our interview script; additional codes are included in Table C.2 for general themes that emerged, which were not specific to an interview question.

Name of Code	Description	# Matched
role > OTcybersecurity	Performs OT cybersecurity tasks in their role	5
role > alarmResponse	Performs alarm response tasks in their role	4
role > engineering	Performs engineering tasks in their role	11
role > manager	Performs management tasks in their role	6
role > operations	Performs operations tasks in their role	4
teamsize	Details about team size	6
architecture > usesPLC	Organization uses PLCs	12
architecture > usesDCS	Organization uses a DCS	5
architecture > usesSCADA	Organization uses SCADA	5
architecture > usesHMI	Organization uses HMIs	7
architecture > usesMainControlRoom	Organization uses a control room	6
architecture > usesSubControlRoom	Organization uses multiple control rooms	2
architecture > usesHistorian	Organization uses a data historian	7
architecture > detailsPLC	Details about how PLCs are used	10
architecture > detailsSCADA	Details about how SCADA/DCS are used	14
alarmArch > PLCs	Alarms come from PLCs	13
alarmArch > SCADA	Alarms come from SCADA/DCS	6
alarmArch > safetySystem	Alarms come from a safety system	11
alarmArch > external	Alarms come from an external tool	2
alarmDefn > bounds	Alarms defined as upper/lower bounds	12
alarmDefn > custom	Alarms defined as custom logic	10
alarmRole > operations	Operators configure alarms	3
alarmRole > team	A team configures alarms	3
alarmRole > vendor	A vendor configures alarms	4
alarmTypes > process	Alarms for unwanted process values	16
alarmTypes > communication	Alarms for communication issues	6
alarmTypes > componentFailure	Alarms for component failures	6
alarmTypes > cybersecurity	Alarms for cybersecurity issues	3
alarmTypes > physicalSecurity	Alarms for physical security issues	3
alarmTypes > other	Alarms for other types of issues	5
alarmResponse > triage	Details about triage process in alarm response	13
alarmResponse > controlRoom	Details about control rooms in alarm response	4
alarmResponse > severity	Details about severity levels in alarm response	9
alarmResponse > operations	Details about coordination with operators in alarm response	11
alarmResponse > humanFactors	Details about human factors in alarm response	6
alarmResponse > UI	Details about user interfaces in alarm response	5
alarmNumber	Details about number/rate of alarms	9
alarmPhilosophy	Details about what should be an alarm	9
alarmActionability	Details about actionability of alarms	4
alarmDiagnosis > challenges	Details about challenges when diagnosing alarms	12
alarmDiagnosis > nuisance	Details about nuisance alarms when diagnosing alarms	12
alarmDiagnosis > intuition	Details about need for intuition when diagnosing alarms	11
alarmDiagnosis > postHoc	Details about post hoc analysis of alarms	10
alarmDiagnosis > tools	Details about vendor tools when diagnosing alarms	8
alarmDiagnosis > ML	Details about using ML to diagnose alarms	3
alarmManage > meeting	Details about alarm management meeting	4
alarmManage > testing	Details about testing alarms	5
alarmManage > update	Details about updating alarms	5

Table C.1: Codes for responses in Part II of our interview script, which focuses on tasks and systems in alarm workflows. For each code, we provide: its name and structure, its description, and the number of participants matched to it.

Name of Code	Description	# Matched
tools > barriersCultural	Barriers to tool adoption from ICS culture	10
tools > barriersTechnical	Barriers to tool adoption from ICS technical limitations	6
tools > peopleExcluded	Details about people excluded in adoption	4
tools > peopleIncluded	Details about people included in adoption	5
tools > values	Details about important values for adoption	16
tools > metrics	Details about how metrics are used to evaluate tools	13
AI > positive	General positive perceptions of AI	15
AI > positive > saveTime	AI will save time	9
AI > positive > complex	AI performs complex tasks better than humans	9
AI > positive > exciting	AI is novel and exciting	3
AI > negative	General negative perceptions of adopting AI	17
AI > negative > trust	ICS would not trust AI	11
AI > negative > criticality	ICS are too critical for AI	10
AI > negative > complex	Parts of ICS are too complex for AI	5
AI > negative > transparency	AI decisions are not transparent	16
AI > negative > dataCost	AI requires data that we do not have	8
AI > negative > moneyCost	AI requires money that we do not have	4
AI > negative > peopleCost	AI requires people than we do not have	6
AI > negative > badAI	Negative perceptions of AI itself	5
AI > conceptual	Conceptual models of AI	18
AI > conceptual > LLM	Talked about LLMs	8
AI > conceptual > neuralNet	Talked about neural networks	5
AI > conceptual > linear	Talked about linear regression or classification	3
AI > conceptual > prediction	Talked about data for AI predictions generally	6
AI > conceptual > training	Talked about data for AI training generally	12
AI > useCase > assistant	Suggest to use AI as an assistant	7
AI > useCase > optimizeProcess	Suggest to use AI to optimize process	8
AI > useCase > optimizeAlarms	Suggest to use AI to optimize alarm workflows	8
AI > useCase > maintenance	Suggest to use AI for system maintenance	3
AI > recommendations	Recommendations for how AI should improve	8
external > alarms	An external party performs part of alarm workflow actions	1
external > implementation	An external party implements part of the alarm workflow	4
external > detailsExternal	Details about contracts with external parties	9
compareIndustry > cultural	Comparing ICS industries based on cultural differences	5
compareIndustry > longitudinal	Comparing ICS based on trends over time	6
compareIndustry > size	Comparing ICS based on their size	5
compareIndustry > technical	Comparing ICS industries based on technical differences	4
misc > cybersecurityPerceptions	Current cybersecurity perceptions in ICS	9
misc > cybersecurityPractices	Current cybersecurity practices in ICS	9
misc > regulations	Government regulations affecting ICS	7
misc > cultureClash	IT/OT culture clash	8
misc > painPeople	Personnel issues affecting ICS	8
misc > painTechnical	Technical issues affecting ICS	9

Table C.2: Codes for responses in Part III and Part IV of our interview scripts, which focus on vendor tool adoption (Part III) and perceptions of AI (Part IV). We also include codes for other miscellaneous themes, such as cross-industry and cross-functional pain points. For each code, we provide: its name and structure, its description, and the number of participants matched to it.

Bibliography

[1] Syed Ghazanfar Abbas, Muslum Ozgur Ozmen, Abdulellah Alsaheel, Arslan Khan, Z. Berkay Celik, and Dongyan Xu. SAIN: Improving ICS attack detection sensitivity via State-Aware invariants. In *Proceedings of the 33rd USENIX Security Symposium*, 2024. 1

[2] Maged Abdelaty, Roberto Doriguzzi-Corin, and Domenico Siracusa. DAICS: A deep learning solution for anomaly detection in industrial control systems. *arXiv:2009.06299*, 2020. 3.3.1, 3.1, 3.2, 3.3.2

[3] Ahmed A. Abokifa, Kelsey Haddad, Cynthia S. Lo, and Pratim Biswas. *Detection of Cyber Physical Attacks on Water Distribution Systems via Principal Component Analysis and Artificial Neural Networks*. 2017. 3.1, 3.2, 5.1, 5.2.2, 6.4.1

[4] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6, 2018. 2.5

[5] Sridhar Adepu and Aditya Mathur. Distributed attack detection in a water treatment plant: Method and case study. *IEEE Transactions on Dependable and Secure Computing*, 18(1), 2021. 5.5.2

[6] Sridhar Adepu, Nianyu Li, Eunsuk Kang, and David Garlan. Modeling and analysis of explanation for secure industrial control systems. *ACM Transactions on Autonomous and Adaptive Systems*, 17(3-4), 2022. 1, 6.5.2

[7] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, 2017. 2.2, 3.1, 4.2.1, 4.2.1, 4.6.1

[8] Chuadhry Mujeeb Ahmed, Gauthama Raman MR, and Aditya P Mathur. Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems. In *Proceedings of the 6th ACM Cyber-Physical System Security Workshop*, 2020. 1

[9] Bushra A. Alahmadi, Louise Axon, and Ivan Martinovic. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In *Proceedings of the 31st USENIX Security Symposium*, 2022. 2.6, 4.5, 6.2.1

[10] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. Enhancing one-class support vector machines for unsupervised anomaly detection. In *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*, 2013. 6.4.1

[11] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert systems with applications*, 186, 2021. 4.2.3

[12] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018. 1, 2.4, 2.4, 4.2.2, 4.2.3

[13] Mohammed Asiri, Neetesh Saxena, Rigel Gjomemo, and Pete Burnap. Understanding indicators of compromise against cyber-attacks in industrial control systems: A security perspective. *ACM Transactions on Cyber-Physical Systems*, 2023. 6, 6.1

[14] Rima Asmar Awad, Saeed Beztchi, Jared M. Smith, Bryan Lyles, and Stacy Prowell. Tools, techniques, and methodologies: A survey of digital forensics for SCADA systems. In *Proceedings of the 4th Annual Industrial Control System Security Workshop*, 2018. 6.1

[15] Michael Bailey, David Dittrich, Erin Kenneally, and Doug Maughan. The menlo report. *IEEE Security & Privacy*, 10(2), 2012. 6.2.4

[16] Gagan Bansal, Besmira Nushi, Ece Kamar, Walter S. Lasecki, Daniel S. Weld, and Eric Horvitz. Beyond accuracy: The role of mental models in human-AI team performance. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 7(1): 2–11, 2019. 6.5.2

[17] Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. Revision of the Tennessee Eastman process model. *IFAC-PapersOnLine*, 48(8):309–314, 2015. 2.2, 4.2.1, 4.2.1, 5.5.1

[18] Margret Bauer, Alexander Horch, Lei Xie, Mohieddine Jelali, and Nina Thornhill. The current state of control loop performance monitoring–a survey of application in industry. *Journal of Process Control*, 38, 2016. 2.6, 5.1

[19] Jack Beerman, David Berent, Zach Falter, and Suman Bhunia. A review of colonial pipeline ransomware attack. In *Proceedings of the IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops*, 2023. 1

[20] Deval Bhamare, Maede Zolanvari, Aiman Erbad, Raj Jain, Khaled Khan, and Nader Meskin. Cybersecurity for industrial control systems: A survey. *Computers & Security*, 89: 101677, 2020. 6.1

[21] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 2006. 6.2.3

[22] Zana Buçinca, Maja Barbara Malaya, and Krzysztof Z Gajos. To trust or to think: cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *ACM on Human-Computer Interaction*, 5(CSCW1), 2021. 6.5.2

[23] Alvaro A Cardenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011. 1, 2.4, 4.2.1, 4.4.3, 5.5.2, 5.6.1, 5.7

[24] Valerie Chen, Q Vera Liao, Jennifer Wortman Vaughan, and Gagan Bansal. Understanding the role of human intuition on reliance in human-AI decision-making with explanations. *ACM on Human-computer Interaction*, 7(CSCW2), 2023. 6.5.3

[25] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018. 1

[26] Jason D. Christopher. Sans 2024 state of ICS/OT cybersecurity. *SANS Institute*, 2024. 1, 6.1, 6.2.4, 6.3

[27] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 2.4

[28] Oakley Cox. Three ways AI secures OT & ICS from cyber attacks, 2024. `https://www.darktrace.com/blog/three-ways-ai-secures-operational-technology-ot-industrial-control-systems-ics-from-cyber-attacks`. 6.1

[29] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5): 4027–4035, 2021. 1, 2.4, 2.4, 4.2.1, 4.2.4, 5.1, 5.2.2, 5.3.1, 5.4, 5.5.1, 6.1

[30] Berend Denkena, M-A Dittrich, Hendrik Noske, and Matthias Witt. Statistical approaches for semi-supervised anomaly detection in machining. *Production Engineering*, 14:385–393, 2020. 6.4.1

[31] Willian Dimitrov and Svetlana Syarova. Analysis of the functionalities of a shared ICS security operations center. In *Big Data, Knowledge and Control Systems Engineering*, 2019. 6.1

[32] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993. 2.2, 4.2.1, 5.5.1

[33] Upol Ehsan, Samir Passi, Q Vera Liao, Larry Chan, I-Hsiang Lee, Michael Muller, and Mark O Riedl. The who in XAI: How AI background shapes perceptions of AI explanations. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024. 6.5.3

[34] Pardis Emami-Naeini, Henry Dixon, Yuvraj Agarwal, and Lorrie Faith Cranor. Exploring how privacy and security factor into IoT device purchase behavior. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019. 6.1, 6.2.3

[35] Alessandro Erba and Nils Ole Tippenhauer. Assessing model-free anomaly detection in industrial control systems against generic concealment attacks. In *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022. 5.5.2, 5.6.3, 5.6.3

[36] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In *Proceedings of*

the 36th Annual Computer Security Applications Conference, 2020. 3, 3.1, 3.3.1, 3.1, 3.2, 5.6.3, 5.6.3

[37] Alessandro Erba, Andres F. Murillo, Riccardo Taormina, Stefano Galelli, and Nils Ole Tippenhauer. On practical realization of evasion attacks for industrial control systems. In *Proceedings of the 2024 Workshop on Re-design Industrial Control Systems with Security*, 2024. 2.2, 5.5.1, 5.6.3, 5.6.3

[38] G Erion, JD Janizek, P Sturmfels, S Lundberg, and SI Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature Machine Intelligence*, 3, 2021. 2.5, 4.1, 4.2.3, 4.3.2

[39] Cheng Feng, Venkata Reddy Palleti, Aditya Mathur, and Deeph Chana. A systematic framework to generate invariants for anomaly detection in industrial control systems. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*, 2019. 1, 3, 3.1, 3.1, 3.3.1, 3.2, 3.3.2, 3.4.2, 3.5.3

[40] Pavel Filonov, Fedor Kitashov, and Andrey Lavrentyev. RNN-based early cyber-attack detection for the Tennessee Eastman process. *arXiv preprint arXiv:1709.02232*, 2017. 2.4, 4.2.2

[41] Clement Fung, Shreya Srinarasi, Keane Lucas, Hay Bryan Phee, and Lujo Bauer. Perspectives from a comprehensive evaluation of reconstruction-based anomaly detection in industrial control systems. In *Proceedings of the 27th European Symposium on Research in Computer Security*, 2022. 1, 3, 4.2.1, 5.6.1, 6.1

[42] Clement Fung, Eric Zeng, and Lujo Bauer. Attributions for ML-based ICS anomaly detection: From theory to practice. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium*, 2024. 1, 4, 5.6, 5.6.3, 5.6.3

[43] Clement Fung, Eric Zeng, and Lujo Bauer. Adopting AI to protect industrial control systems: Assessing challenges and opportunities from the operators' perspective. In *Proceedings of the Twenty-First Symposium on Usable Privacy and Security*, 2025. 1, 6

[44] Andrea Gallardo, Robert Erbes, Katya Le Blanc, Lujo Bauer, and Lorrie Faith Cranor. Interdisciplinary approaches to cybervulnerability impact assessment for energy critical infrastructure. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024. 2.6, 2.6, 6.3.4

[45] Luis Garcia, Ferdinand Brasser, Mehmet Hazar Cintuglu, Ahmad-Reza Sadeghi, Osama A Mohammed, and Saman A Zonouz. Hey, my malware knows physics! attacking PLCs with physical model aware rootkit. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium*, 2017. 1

[46] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys*, 51(4):1–36, 2018. 1, 5.7, 6.5.2

[47] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. Anomaly detection in cyber physical systems using recurrent neural networks. In *Proceedings of the 18th International Symposium on*

High Assurance Systems Engineering, 2017. 3.1, 3.2

[48] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *Proceedings of the 11th International Conference on Critical Information Infrastructures Security*, 2016. 2.2, 3.1, 3.3.2, 4.2.1, 4.2.1, 4.6.1, 5.5.1, 5.5.2

[49] Benjamin Green, Marina Krotofil, and Ali Abbasi. On the significance of process comprehension for conducting targeted ICS attacks. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy*, 2017. 1, 4.4.2

[50] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. LEMNA: Explaining deep learning based security applications. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018. 2.5, 4.1, 4.2.3

[51] Hana Habib, Sarah Pearman, Jiamin Wang, Yixin Zou, Alessandro Acquisti, Lorrie Faith Cranor, Norman Sadeh, and Florian Schaub. ït's a scavenger hunẗ: Usability of websites' opt-out and data deletion choices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020. 6.1, 6.2.3

[52] Dina Hadžiosmanović, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. Through the eye of the PLC: Semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014. 1, 2.4, 2.4, 4.2.2, 5.5.1

[53] Anna Hall and Vivek Agarwal. Barriers to adopting artificial intelligence and machine learning technologies in nuclear power. *Progress in Nuclear Energy*, 175, 2024. 1

[54] Siho Han and Simon S. Woo. Learning sparse latent graph representations for anomaly detection in multivariate time series. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. 1, 2.4, 2.4, 5.1, 5.2.2, 5.3.1, 5.4, 5.5.1

[55] Grant Ho, Mayank Dhiman, Devdatta Akhawe, Vern Paxson, Stefan Savage, Geoffrey M. Voelker, and David Wagner. Hopper: Modeling and detecting lateral movement. In *Proceedings of the 30th USENIX Security Symposium*, 2021. 2.4

[56] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8), 1997. 2.4

[57] Bill R. Hollifield. Understanding & applying the ANSI-ISA 18-2 alarm management standard. Technical report, Hexagon, 2023. 6.3.2

[58] Chanwoong Hwang and Taejin Lee. E-SFD: Explainable sensor fault detection in the ICS anomaly detection system. *IEEE Access*, 9:140470–140486, 2021. 4.1, 4.2.3, 4.2.3, 4.2.4, 4.4.1, 4.6.1

[59] Won-Seok Hwang, Jeong-Han Yun, Jonguk Kim, and Hyoung Chun Kim. Time-series aware precision and recall for anomaly detection: Considering variety of detection result and addressing ambiguous labeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019. 3.1, 3.5.1, 5.6.1

[60] Moses Ike, Kandy Phan, Keaton Sadoski, Romuald Valme, and Wenke Lee. SCAPHY: Detecting modern ICS attacks by correlating behaviors in SCADA and PHYsical. In

Proceedings of the 2023 IEEE Symposium on Security and Privacy, 2023. 1

[61] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun. Anomaly detection for a water treatment system using unsupervised machine learning. In *Proceedings of the 2017 IEEE International Conference on Data Mining Workshops*, 2017. 3.1, 3.2, 3.3.2, 3.4.1

[62] Albert T. Jones and Charles R. McLean. A proposed hierarchical control model for automated manufacturing systems. *Journal of Manufacturing Systems*, 5(1), 1986. 2.1

[63] Eunsuk Kang, Sridhar Adepu, Daniel Jackson, and Aditya P. Mathur. Model-based security analysis of a water treatment system. In *Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, 2016. 1

[64] Jonguk Kim, Jeong-Han Yun, and Hyoung Chun Kim. Anomaly detection for industrial control systems using sequence-to-sequence neural networks. In *Proceedings of the 5th Workshop on the Security of Industrial Control Systems and of Cyber-Physical Systems*, 2019. 3.1, 3.2, 3.3.2

[65] Taewook Kim, Hyomin Han, Eytan Adar, Matthew Kay, and John Joon Young Chung. Authors' values and attitudes towards AI-bridged scalable personalization of creative language arts. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, 2024. 6.2.4

[66] Abigail MY Koay, Ryan K L Ko, Hinne Hettema, and Kenneth Radke. Machine learning in industrial control system (ICS) security: current landscape, opportunities and challenges. *Journal of Intelligent Information Systems*, 60(2), 2023. 6.1, 6.5.1

[67] Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. Matched and mismatched SOCs: A qualitative study on security operations center issues. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019. 2.6

[68] Moshe Kravchik and Asaf Shabtai. Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and Privacy*, 2018. 3, 3.1, 3.1, 3.3.1, 3.2, 3.4.2, 3.4.2, 3.5.3

[69] Moshe Kravchik and Asaf Shabtai. Efficient cyber attacks detection in industrial control systems using lightweight neural networks. *arXiv:1907.01216*, 2019. 1, 2.4, 2.4, 3, 3.2, 3.3.2, 4.1, 4.2.1, 4.2.2, 4.2.3, 4.3.1, 4.3.1, 4.4.1, 4.6.1, 6.4.1

[70] Moshe Kravchik and Asaf Shabtai. Efficient cyber attack detection in industrial control systems using lightweight neural networks and PCA. *IEEE Transactions on Dependable and Secure Computing*, 19(4), 2022. 1, 2.4, 3, 3.1, 3.3.1, 3.1, 3.3.2, 3.4.2, 5.1, 5.2.2, 5.4, 5.5.1, 5.5.2, 5.6.1, 6.1

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 2.4, 5.2.2

[72] Marina Krotofil and Jason Larsen. Rocking the pocket book: Hacking chemical plants, 2015. 4.2.1, 4.4.3

[73] Marina Krotofil, Jason Larsen, and Dieter Gollmann. The process matters: Ensuring

data veracity in cyber-physical systems. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, 2015. 1

[74] David Kushner. The real story of Stuxnet. *IEEE Spectrum*, 53(3), 2013. 1

[75] Olav Lamberts, Konrad Wolsing, Eric Wagner, Jan Pennekamp, Jan Bauer, Klaus Wehrle, and Martin Henze. SoK: Evaluations in industrial intrusion detection research. *Journal of Systems Research*, 3(1), 2023. 1, 3.1, 5.6.1, 5.6.1, 6.5.1, 6.5.2

[76] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms–the Numenta anomaly benchmark. In *Proceedings of the 14th International Conference on Machine Learning and Applications*, 2015. 3.1, 3.5.1, 3.5.2, 3.5.2, 3.5.2

[77] D. Lavrova, D. Zegzhda, and A. Yarmak. Using GRU neural network for cyber-attack detection in automated process control systems. In *Proceedings of the 7th IEEE International Black Sea Conference on Communications and Networking*, 2019. 2.4, 2.4, 5.1, 5.2.2, 5.4, 5.5.1

[78] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015. 2.4

[79] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *Proceedings of the 28th International Conference on Artificial Neural Networks*, 2019. 3.1, 3.2, 6.1

[80] Qin Lin, Sridha Adepu, Sicco Verwer, and Aditya Mathur. TABOR: A graphical model-based approach for anomaly detection in industrial control systems. In *Proceedings of the 2018 Asia Conference on Computer and Communications Security*, 2018. 1, 3.1, 3.3.1, 3.2

[81] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 2011. 1, 2.1

[82] Philipp Liznerski, Lukas Ruff, Robert A Vandermeulen, Billy Joe Franks, Marius Kloft, and Klaus Robert Muller. Explainable deep one-class classification. In *Proceedings of the Ninth International Conference on Learning Representations*, 2021. 5.2.1

[83] Sara Ljungblad, Yemao Man, Mehmet Aydın Baytaş, Mafalda Gamboa, Mohammad Obaid, and Morten Fjeld. What matters in professional drone pilots' practice? an interview study to understand the complexity of their work and inform human-drone interaction research. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021. 6.2.4

[84] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017. 2.5, 4.1, 4.2.3

[85] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng Yao. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys*, 54(5):1–36, 2021. 1, 6.1, 6.5.2

[86] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. Reliability and inter-rater relia-

bility in qualitative research: Norms and guidelines for CSCW and HCI practice. *ACM on Human-Computer Interaction*, 3(CSCW), 2019. 6.2.3

[87] Stephen McLaughlin, Charalambos Konstantinou, Xueyang Wang, Lucas Davi, Ahmad-Reza Sadeghi, Michail Maniatakos, and Ramesh Karri. The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 104, 2016. 6.1

[88] Yifei Ming, Hang Yin, and Yixuan Li. On the impact of spurious correlation for out-of-distribution detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36 (9):10051–10059, 2022. 5.4

[89] Jaron Mink, Hadjer Benkraouda, Limin Yang, Arridhana Ciptadi, Ali Ahmadzadeh, Daniel Votipka, and Gang Wang. Everybody's got ML, tell me what else you have: Practitioners' perception of ML-based security tools and explanations. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy*, 2023. 2.6, 5.1, 6.2.4

[90] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. In *Proceedings of the 25th Annual Network and Distributed System Security Symposium*, 2018. 2.4

[91] Hussein Mozannar, Gagan Bansal, Adam Fourney, and Eric Horvitz. When to show a suggestion? integrating human feedback in AI-assisted programming. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(9):10137–10144, 2024. 6.5.2

[92] Andres Murillo, Riccardo Taormina, Nils Tippenhauer, and Stefano Galelli. Co-simulating physical processes and network data for high-fidelity cyber-security experiments. In *Proceedings of the Sixth Annual Industrial Control System Security Workshop*, 2021. 2.2, 5.5.1

[93] Dinh C Nguyen, Ming Ding, Pubudu N Pathirana, Aruna Seneviratne, Jun Li, and H Vincent Poor. Federated learning for internet of things: A comprehensive survey. *IEEE Communications Surveys and Tutorials*, 23(3), 2021. 6.5.1

[94] Megan Nyre-Yu, Elizabeth Morris, Michael Smith, Blake Moss, and Charles Smutz. Explainable ai in cybersecurity operations: Lessons learned from XAI tool deployment. In *Proceedings of the Usable Security and Privacy Symposium*, 2022. 2.6, 6.5.2

[95] Cliodhna O'Connor and Helene Joffe. Intercoder reliability in qualitative research: Debates and practical guidelines. *International Journal of Qualitative Methods*, 19, 2020. 6.2.3

[96] Sean Oesch, Robert Bridges, Jared Smith, Justin Beaver, John Goodall, Kelly Huffer, Craig Miles, and Dan Scofield. An assessment of the usability of machine learning based tools for the security operations center. In *Proceedings of the 2020 International Conferences on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data and IEEE Congress on Cybermatics*, 2020. 2.6

[97] Dean Parsons. SANS ICS/OT cybersecurity survey: 2023's challenges and tomorrow's defenses. *SANS Institute*, 2023. 6.3.4

[98] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, and

Félix J. García Clemente. MADICS: A methodology for anomaly detection in industrial control systems. *Symmetry*, 12(10), 2020. 1, 2.4, 3.1, 3.2, 4.2.1, 4.2.2, 5.1, 5.1, 5.5.2

[99] Raul Quinonez, Jairo Giraldo, Luis Salazar, Erick Bauman, Alvaro Cardenas, and Zhiqiang Lin. SAVIOR: Securing autonomous vehicles with robust physical invariants. In *Proceedings of the 29th USENIX Security Symposium*, 2020. 1

[100] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. 2.5, 4.1, 4.2.3

[101] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. MLaaS: Machine learning as a service. In *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications*, 2015. 6.5.1

[102] Robert M. Lee, Michael J. Assante and Tim Conway. Analysis of the cyber attack on the Ukrainian power grid. *Electricity Information Sharing and Analysis Center*, 388, 2016. 1

[103] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. 5.2.1

[104] Imran Sajjad, Daniel D. Dunn, Rajnikant Sharma, and Ryan Gerdes. Attack mitigation in adversarial platooning using detection-based sliding mode control. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, 2015. 1

[105] Benjamin Saunders, Julius Sim, Tom Kingstone, Shula Baker, Jackie Waterfield, Bernadette Bartlam, Heather Burroughs, and Clare Jinks. Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & quantity*, 52: 1893–1907, 2018. 6.2.1

[106] Martin A Sehr, Marten Lohstroh, Matthew Weber, Ines Ugalde, Martin Witte, Joerg Neidig, Stephan Hoeme, Mehrdad Niknami, and Edward A Lee. Programmable logic controllers in the context of industry 4.0. *IEEE Transactions on Industrial Informatics*, 17(5), 2020. 5.1

[107] Dmitry Shalyga, Pavel Filonov, and Andrey Lavrentyev. Anomaly detection for water treatment system based on neural network with automatic architecture optimization. *arXiv:1807.07282*, 2018. 3.1, 3.3.1, 3.2

[108] Wenli Shang, Peng Zeng, Ming Wan, Lin Li, and Panfeng An. Intrusion detection algorithm based on OCSVM in industrial control system. *Security and Communication Networks*, 9(10), 2016. 6.4.1

[109] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. 4.2.1

[110] Yasser Shoukry, Paul Martin, Yair Yona, Suhas Diggavi, and Mani Srivastava. PyCRA: Physical challenge-response authentication for active sensors under spoofing attacks. In

Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015. 1

[111] Alex Shultz. For the first time, artificial intelligence is being used at a nuclear power plant: California's diablo canyon. The Markup, 2025. https://themarkup.org/artificial-intelligence/2025/04/08/for-the-first-time-artificial-intelligence-is-being-used-at-a-nuclear-power-plant-californias-diablo-canyon. 6.1

[112] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 5.3.2

[113] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2.5, 4.1, 4.2.3

[114] Brian Singer, Amritanshu Pandey, Shimiao Li, Lujo Bauer, Craig Miller, Lawrence Pileggi, and Vyas Sekar. Shedding light on inconsistencies in grid cybersecurity: Disconnects and recommendations. In *Proceedings of the 2023 IEEE Symposium on Security and Privacy*, 2023. 2.6

[115] N. Singh and C. Olinsky. Demystifying Numenta anomaly benchmark. In *Proceedings of the 2017 International Joint Conference on Neural Networks*, 2017. 3.5.2

[116] Pooja Singh and Lalit Kumar Singh. Instrumentation and control systems design for nuclear power plant: An interview study with industry practitioners. *Nuclear Engineering and Technology*, 53(11), 2021. 2.6

[117] Joseph Slowik. Evolution of ICS attacks and the prospects for future disruptive events. *Threat Intelligence Centre Dragos Inc*, 2019. 1, 2.1, 6.2.4

[118] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 2.5, 4.1, 4.2.3, 4.3.2

[119] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4), 2009. 3.1

[120] Lei Song, Chuheng Zhang, Li Zhao, and Jiang Bian. Pre-trained large language models for industrial control. *arXiv preprint arXiv:2308.03028*, 2023. 6.4.1

[121] Kirti Soni, Nishant Kumar, Anjali S Nair, Parag Chourey, Nirbhow Jap Singh, and Ravinder Agarwal. Artificial intelligence: Implementation and obstacles in industry 4.0. In *Handbook of Metrology and Applications*. Springer, 2022. 1

[122] Keith Stouffer. Guide to industrial control systems (ICS) security. *NIST special publication*, 800(82), 2011. 2.1

[123] Eric Stranz and Stefan Nohe. NERC CIP in the real world on a real budget. In *Proceedings of the Minnesota Power Systems Conference*, 2016. 6.3.4

[124] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

2.5, 4.1, 4.2.3, 4.3.2

[125] Riccardo Taormina and Stefano Galelli. 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. 2.4, 2.4, 3, 3.1, 3.1, 3.2, 3.4.2, 3.5.3, 6.4.1

[126] Riccardo Taormina, Stefano Galelli, Nils Ole Tippenhauer, Elad Salomons, Avi Ostfeld, Demetrios G Eliades, Mohsen Aghashahi, Raanju Sundararajan, Mohsen Pourahmadi, M Katherine Banks, 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. 2.2, 3.1

[127] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. Precision and recall for time series. In *Advances in Neural Information Processing Systems*, 2018. 3.1, 3.5.1, 3.5.2, 3.5.2

[128] Liang Tong, Bo Li, Chen Hajaj, Chaowei Xiao, Ning Zhang, and Yevgeniy Vorobeychik. Improving robustness of ML classifiers against realizable evasion attacks using conserved features. In *Proceedings of the 28th USENIX Security Symposium*, 2019. 4.2.1

[129] Federico Turrin, Alessandro Erba, Nils Ole Tippenhauer, and Mauro Conti. A statistical analysis framework for ICS process datasets. In *Proceedings of the 2020 Joint Workshop on CPS&IoT Security and Privacy*, 2020. 3.3.2, 5.6.1

[130] Muhammad Azmi Umer, Aditya Mathur, Khurum Nazir Junejo, and Sridhar Adepu. Generating invariants using design and data-centric approaches for distributed attack detection. *International Journal of Critical Infrastructure Protection*, 28:100341, 2020. 1

[131] Muhammad Azmi Umer, Khurum Nazir Junejo, Muhammad Taha Jilani, and Aditya P. Mathur. Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations. *International Journal of Critical Infrastructure Protection*, 38:100516, 2022. 1

[132] Muaan ur Rehman and Hayretdin Bahşi. Process-aware security monitoring in industrial control systems: A systematic review and future directions. *International Journal of Critical Infrastructure Protection*, 47:100719, 2024. 1, 6.5.2

[133] David I. Urbina, Jairo A. Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016. 1, 2.4, 2.4, 4.4.2, 5.5.1, 5.5.2, 5.6.1

[134] Mathew Vermeer, Natalia Kadenko, Michel van Eeten, Carlos Gañán, and Simon Parkin. Alert alchemy: SOC workflows and decisions in the management of NIDS rules. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023. 2.6, 6.2.1

[135] Dennis Wagner, Tobias Michels, Florian CF Schulz, Arjun Nair, Maja Rudolph, and Marius Kloft. TimeseAD: Benchmarking deep multivariate time-series anomaly detection.

Transactions on Machine Learning Research, 2023. 5.6.1

[136] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds. *ACM Transactions on Graphics*, 38(5), 2019. 5.3.2

[137] Zijie J Wang, Alex Kale, Harsha Nori, Peter Stella, Mark E Nunnally, Duen Horng Chau, Mihaela Vorvoreanu, Jennifer Wortman Vaughan, and Rich Caruana. Interpretability, then what? editing machine learning models to reflect human knowledge and values. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022. 6.5.3

[138] Zijie J Wang, Jennifer Wortman Vaughan, Rich Caruana, and Duen Horng Chau. GAM coach: Towards interactive and user-centered algorithmic recourse. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023. 6.5.3

[139] Alexander Warnecke, Daniel Arp, Christian Wressnegger, and Konrad Rieck. Evaluating explanation methods for deep learning in security. In *Proceedings of the 2020 IEEE European Symposium on Security and Privacy*, 2020. 4.2.4

[140] Joseph Weiss, Rob Stephens, and Nadine Miller. Control system cyber incidents are real—and current prevention and mitigation strategies are not working. *Computer*, 55, 2022. 6.3.4

[141] Konrad Wolsing, Lea Thiemt, Christian van Sloun, Eric Wagner, Klaus Wehrle, and Martin Henze. Can industrial intrusion detection be SIMPLE? In *Proceedings of the 27th European Symposium on Research in Computer Security*, 2022. 2.4

[142] Konrad Wolsing, Eric Wagner, Luisa Lux, Klaus Wehrle, and Martin Henze. GeCos replacing experts: Generalizable and comprehensible industrial intrusion detection. In *Proceedings of the 34th USENIX Security Symposium*, 2025. 1, 2.4, 2.4, 5.5.1, 5.5.2, 5.5.2, 5.6.1, 5.6.1, 5.7

[143] Renjie Wu and Eamonn J Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 35(3), 2021. 3.1, 5.6.1

[144] Qian Yang, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. Re-examining whether, why, and how human-AI interaction is uniquely difficult to design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020. 6.5.2

[145] Wenqian Ye, Guangtao Zheng, Xu Cao, Yunsheng Ma, and Aidong Zhang. Spurious correlations in machine learning: A survey. *arXiv preprint arXiv:2402.12715*, 2024. 5.4

[146] Alberto Zanutto, Benjamin Oliver Shreeve, Karolina Follis, Jeremy Simon Busby, and Awais Rashid. The shadow warriors: In the no man's land between industrial control systems and enterprise IT systems. In *Proceedings of the Thirteenth Symposium on Usable Privacy and Security*, 2017. 6.3.4

[147] Carson Zimmerman. Cybersecurity operations center. *The MITRE Corporation*, 2014. 2.6

[148] Giulio Zizzo, Chris Hankin, Sergio Maffeis, and Kevin Jones. Adversarial attacks on

time-series intrusion detection for industrial control systems. In *Proceedings of the 19th International Conference on Trust, Security and Privacy in Computing and Communications*, 2020. 1, 2.4, 2.4, 3, 3.1, 3.1, 3.2, 3.3.2, 3.3.2, 4.2.1, 4.2.2, 4.3.1, 5.1, 5.2.2, 5.4, 5.5.1, 5.6.1