Comparison of Industrial Control System Anomaly Detection Methods

Piotr Sobonski
Collins Aerospace Applied Research and Technology Ireland Limited
Enterprise Engineering
Cork, Cork, Ireland
piotr.sobonski@collins.com

Utz Roedig
University College Cork
School of Computer Science and Information Technology
Cork, Cork, Ireland
u.roedig@ucc.ie

Abstract

Industrial Control System (ICS) are used to produce goods that must be free of errors. Examples are medicines, medical equipment or vehicle parts. It is essential in such production environments to detect an attack which may aim to compromise goods. While Anomaly Detection (AD) is common to protect Information Technology (IT) infrastructure, it is not yet widely used to protect Operational Technology (OT) elements such as ICS and ultimately production. In this work we analyze the usefulness of different AD algorithms in the context of ICS. We aim to determine if simple statistical methods such as K-Means clustering (K-Means), Density-Based Spatial Clustering of Applications with Noise (DBSCAN), Stochastic Gradient Decent (SGD) or Support Vector Machine (SVM) are sufficient or if more advanced Machine Learning (ML) algorithms such as an Autoencoder are necessary to achieve a useful performance. Specifically, we consider real-world constraints such as limited available attack examples in training data and variations in background conditions. We use an evaluation framework called Anomaly Detection Evaluation Framework (ADEF) to model an autoclave manufacturing use case and possible attacks. Using ADEF we benchmark different AD algorithms. Our results show that simple methods perform very well, that large amount of attack examples are unnecessary and that fluctuations in environmental conditions pose a significant challenge.

CCS Concepts

• **Security and privacy** → *Intrusion detection systems.*

Keywords

ICS security; ICS simulation; ICS anomaly detection; ICS attacks; CPS security

ACM Reference Format:
Piotr Sobonski and Utz Roedig. 2024. Comparison of Industrial Control System Anomaly Detection Methods. In *Proceedings of the 2024 Workshop on Re-design Industrial Control Systems with Security (RICSS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3689930.3695211

1 Introduction

Produced goods must be free of errors, especially if a faulty product can endanger lives. For example, medicines, medical equipment or vehicle parts fall in this category. Cyber attacks are besides general faults in a production line a reason why products can be compromised. It is therefore desirable to understand when a production line is under attack. As an attacker will usually try to inflict damage without being noticed, detection is challenging.

As ICS are at the heart of most production processes it is the aim to augment these with AD capability. While it is possible to monitor network traffic to ICS components to detect an attack, this is considered unsatisfactory; the OT side should not be treated the same as classical IT infrastructure. Instead, it is necessary to implement AD within the OT environment and to monitor closely the behavior of process control.

If the intended behavior of all ICS control loops is known a priori it is possible to detect when a process starts to operate out of intended bounds. However, this is not a realistic assumption in industrial settings for two reasons: (i) The security team operates independent of the process engineering team and there is limited exchange of information (i.e. specification of the process). (ii) The security system uses an independent (sensing) infrastructure and is not coupled with the ICS driving production. Thus, practical AD in ICS usually makes use of an approach where the AD uses production cycles that are known to be attack free to learn the system behavior. Thereafter, the AD monitors the process and generates an alarm when a potential attack is observed.

AD within an ICS can make use of a variety of detection algorithms. Relatively simple statistical methods such as K-Means[14], DBSCAN[6], SGD[4] or SVM[10] is used or more advanced ML algorithms such as an Autoencoder[2] is employed. However, it is an open question if advanced ML algorithms are required or if more simplistic approaches are sufficient. It is important to determine how much effort is required to achieve acceptable detection performance. Production environments can comprise thousands of control loops that require monitoring and resources to train and execute models are limited. It is challenging to answer this question as real-world constraints in production environments must be considered. There might be unknown attacks that are not present in training data. Also, there will be variations in the environment that add noise to training data.

In this work we consider an autoclave manufacturing process as example use case. Materials undergo a heat treatment process in a chamber. The heating and cooling process must follow a specific profile. An attacker may aim to influence this cycle as it then may produce parts that pass visual inspection but do not comply otherwise (e.g. material strength).

The scientific contributions are as follows:

- *Autoclave Process and Attack Scenarios:* We provide an autoclave process description and a description of attack scenarios. The process and attacks are coded for the publicly available Anomaly Detection Evaluation Framework (ADEF)[1]. The process and attack descriptions within ADEF provide researchers with a reference scenario to study AD in an ICS context. Researchers may use the reference scenario to study the performance of other algorithms and compare their findings with results reported in this paper.
- *AD Performance Evaluation for an autoclave:* We provide a comprehensive evaluation of simple statistical AD methods such as K-Means, DBSCAN, SGD and SVM and compare these with a more advanced AD method using an Autoencoder. Specifically, we consider real-world constraints such as variations in the environment and limited amount of known attacks available for training.

In the next section we describe related work. In Section 3 we describe the autoclave process and attacks on this process used within this study. Section 4 introduces the ADEF framework that is used in this work to evaluate performance of AD models. Section 5 describes the evaluation of the AD methods using ADEF and the autoclave use case. Section 6 concludes the paper.

2 Related Work

The development of new AD methods or more broadly of new Outlier Detection (OD) methods in the ICS domain is a very active research area (for example, Hooi [9]). This AD research has resulted in a number of libraries and frameworks useful in the ICS domain.

The Python Outlier Detection (PyOD) by [18] and Supervised Outlier Detection (SuOD)[17] libraries are generic ML models that can be tailored for an industrial use case. Alternative examples include the Time-series Outlier Detection System (TODS) [11] library which provides time series anomaly detection ML methods. More advanced models using deep learning methods are implemented in DARTS [7]. This library provides state-of-the-art implementations of the Autoregressive Integrated Moving Average (ARIMA) series of methods including deep learning approaches. TranAD [15] provides deep transformer network based anomaly detection and a diagnosis model to analyze multivariate production data sets and reduces significantly training time in comparison to listed baselines while increasing accuracy. The Unsupervised Anomaly Detection on Multivariate Time Series (USAD) [1] library illustrates unsupervised learning examples by using adversely trained Autoencoders on multivariate time series data. The Python Graph based Outlier Detection (PyGOD) [12] library uses graph based anomaly detection methods with aim to support critical industrial systems. Robust interpretability is provided by the graph based time series multivariate anomaly detection approach by Zhao et al. [16].

Our work does not aim to add to this growing pool of available AD methods. Instead, our work aims to provide comparison of such methods in the ICS context. Rather than finding the best performing method we shed light on the performance difference of simple statistical methods and sophisticated ML based approaches in a realistic use case. The aforementioned existing libraries can be integrated within the ADEF framework and used with the autoclave process if required.

A number of works compare the performance of AD methods in the ICS context. Schmidl et al. [13] provides a very comprehensive evaluation of the state-of-the-art AD ML models against open source time series data sets. A review of OD for time series data is provided by Bl'azquez-Garc'ia et al. [3]. The work provides an OD taxonomy and elaborates on detection techniques; implementations in form of software libraries are referenced. A very detailed survey by Braei and Wagner [5] encompasses AD on time series data using statistical methods K-Means, DBSCAN, Local Outlier Factor (LOF), Isolation Forest, one class SVM, Autoregressive Moving Average (ARMA), ARIMA) and deep learning models Multi Layer Perceptron (MLP), Convolutional Neural Network (CNN), Residual Neural Network (Resnet), Long Short-Term Memory (LSTM), Gradient Recurrent Unit (GRU) and Autoencoders.

While these existing works provide basic comparison of AD methods in an ICS context, the insights gained are limited to the specific data sets and use cases considered. In our work we focus on the impact of limited attack samples for training and variations in the environment between training and inference. Insights regarding these aspects cannot be extracted from existing studies. We provide for our work the autoclave process and attack description for the ADEF framework such that researchers can adjust our experimentation setup to their needs. We hope that this will increase the usefulness of our contribution.

3 The Autoclave Use Case

The autoclave process is used in wide range of industries such as automotive and aerospace to create composite material with desired physical properties that cannot be achieved using standard materials. The process uses high pressure and temperature to enhance product properties. Typically this process uses layered composite materials combined with adhesive to create under high pressure and temperature material with superior physical properties. At the same time the produced parts are lightweight in comparison to equivalent parts made of homogenous material. The precise control of temperature, pressure, and time ensures the part achieves the desired material properties, including high strength-to-weight ratio and durability. Control of the parameters of the autoclave process is usually carried out by an ICS.

3.1 Threat Model

An attacker may aim to interfere with the autoclave such that the resulting product is compromised. The attacker is likely to interfere in a way such that produced parts pass a simple (visual) inspection but do not exhibit the required properties. For example, the part may not have the required strength but it looks ok. To detect such

[1] Anomaly Detection Evaluation Framework (ADEF) Framework-https://github.com/sbnsix/adef

fault, a destructive test is required which can only be applied to some product samples. The attacker may know which parts undergo such testing and might not attack the production of these. Thus, AD may be used to ensure integrity of the production process and, thus, resulting parts.

We consider that the attacker compromises the ICS to implement an attack. An alternative approach would be for an attacker to directly tamper with the physical autoclave; we do not consider this case. The attacker may alter control loops by modifying configurations such as change of set points, change of timings or by falsifying sensor inputs.

As we assume that the attacker can manipulate the ICS for an attack it is not sensible to place AD functionality in the ICS as the attacker would then be able to disable it. AD should be carried out by an independent system with independent sensors. As the AD is independent from the ICS it is usually necessary to learn the normal process behavior.

3.2 Autoclave Process

Fig. 1 illustrates the temperature time series representing an autoclave process. The autoclave profile consists of three phases: a) heating - the step in which the temperature inside the chamber reaches the required processing temperature; b) part treatment - upon reaching the required temperature the parts go through the autoclave process where physical properties and/or part dimensions are established, c) cooling - the temperature and chamber pressure returns back to the ambient background to enable part removal from the chamber.

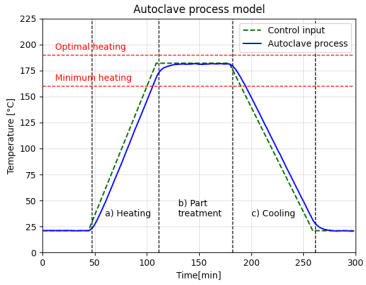

Figure 1: Autoclave process comprising three stages of thermal processing: a) heating - raising the temperature inside the autoclave chamber, b) part treatment - physical properties and part dimensions are established, c) cooling - restoring ambient temperature to remove parts.

3.3 Autoclave Attacks

We created seven different attacks to the autoclave process that are used to evaluate various AD methods. Fig. 2 illustrates the attacks an attacker may use to disrupt the autoclave process. Areas in which the attack is present is shown in red; the blue area shows normal operation; the green line shows the ideal cycle.

Other forms of attacks may exist; this list is not (and cannot be) complete. However, these provide a sample selection of reasonable attacks an adversary may execute.

The attacks are the following:

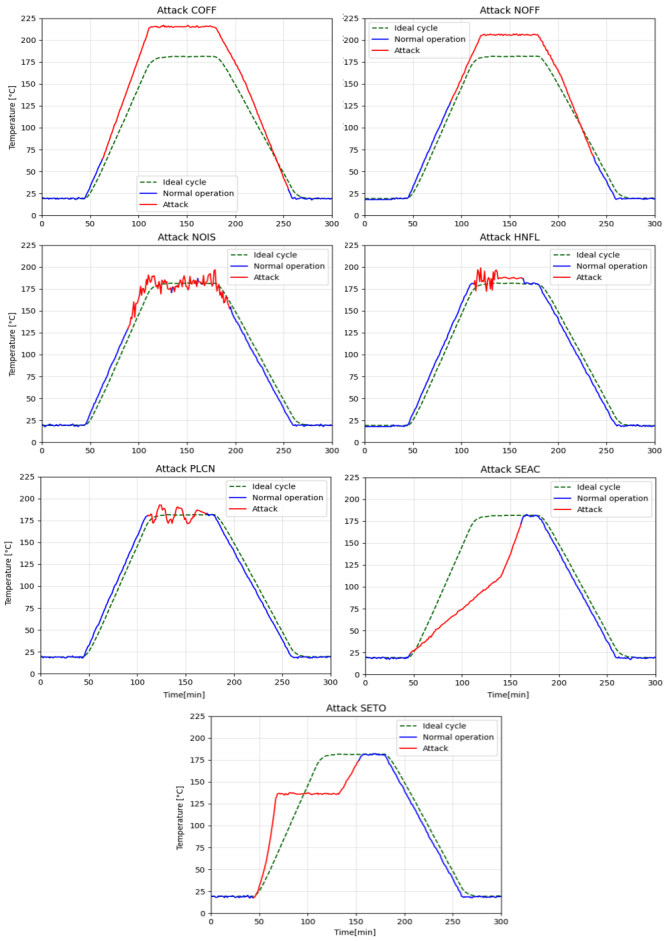

Figure 2: Attacks classes for ICS autoclave process: 1) Constant Offset (COFF) - Constant offset, 2) Noise Offset (NOFF) - Noise offset, 3) Noise Addition (NOIS) - Noise, 4) High Noise and Flat Line (HNFL) - High noise and flat line, 5) PLC noise (PLCN) - Programmable Link Controller (PLC) noise, 6) Sensor Active Control (SEAC) - Sensor active control, 7) Sensor Temperature Off (SETO) - Sensor temperature off.

(1) COFF - Constant offset attack - The attacker modifies the target temperature used for the autoclave process. The temperature is too high or too low.
(2) NOFF - Noise offset - This attack is a mixture of COFF and NOIS attacks.
(3) NOIS - Noise - The attacker controls the heater component and injects a thermal variation (thermal noise). The outcome of this attack will be a lower quality material.
(4) HNFL - High noise and flat line - The attacker modifies the heater component and injects initially a variation in the temperature profile followed by a constant temperature different to the desired set point.
(5) PLCN - PLC noise - The attacker adds noise to the sensor reading of the temperature in the autoclave leading to variations in the temperature in the autoclave.
(6) SEAC - Sensor Active Control - The attacker modifies the data the temperature sensor is providing.

(7) SETO - Sensor Temperature Off - The attacker prevents the temperature sensor to send updated values; the control process operates on outdated data.

The autoclave process and all attacks are implemented in the ADEF framework. Within the framework it is possible to create sequences of autoclave production cycles and to inject some cycles that correspond to the aforementioned attacks. Each attack can be parameterized (e.g. the offset for COFF; the level of noise for NOIS and so on).

4 Anomaly Detection Evaluation Framework (ADEF)

We use in this work the public available ADEF[2]. The framework enables benchmarking of different AD algorithms considering specific ICS use cases. In this work we consider the aforementioned autoclave process and the outlined attacks as the use case.

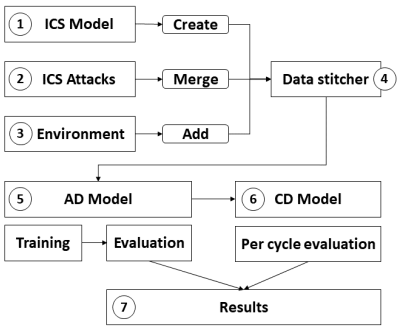

Figure 3: acADEF simulation process flow: 1) acICS Model-the ICS Model creates initial data representation of ICS system inside simulator, 2) acICS attacks-ICS attack generator for ICS, 3) acICS environment-environment model or external disturbances data that is influencing ICS model, 4) Data stitcher-process that combines all ICS inputs to create finalized data set that is used in ADEF framework, 5) AD Model-AD model training and evaluation step where AD model is prepared, 6) CD Model-Cycle Detector (CD) model evaluation step using windowing technique, 7) Results-generator experimental results in compressed form (images, presentations etc.)

Fig. 3 depicts ADEF simulation process flow. In the first step, a sequence of production cycles considering the use case (here the autoclave) are produced. Each production cycle follows the shape shown in Fig. 1 with a degree of variation to reflect a realistic environment. Next, production cycles reflecting attacks are generated. Both inputs are then combined at a stitcher module to produce two sets of data: training and evaluation data. Here it can be controlled how many production cycles are under attack and what type of attacks are used. If there are any additional environment based inputs they can be merged as supplied input file with the datasets. In case of the autoclave, relevant environment data is the background temperature that varies throughout the year and changes start and endpoints of heat treatment cycles.

Once formation of training and evaluation data is completed the AD models considered are trained. The training phase is using hyper parameter search to find the best AD model configuration. An AD model usually provides a decision (attack or no attack) for each incoming time series point of each production cycle. However, in a practical industrial setting it is of higher interest to judge if the entire production cycle should be considered compromised. For this purpose we use a Cycle Detector (CD). For the CD we consider windowing; when the AD model has detected an attack for a number of consecutive time series points of the production cycle we consider the cycle to be compromised. The CD improves in most cases on the AD performance.

Finally, the performance of AD and CD are evaluated using a set of metrics.

4.1 Evaluation Metrics

The ADEF computes accuracy and delay metrics. Accuracy metrics show how well AD and CD detect attacks; Delay metrics show how long it takes before either AD or CD notice a production cycle compromise.

Detection accuracy metrics.

- **Area Under the Curve (AUC)**-AUC computed from the Receiver Operating Characteristic (ROC) curve for AD and CD.
- **Equal Error Rate (EER)**[8]-Equal Error Rate is measuring model error rate value for given experiment,
- **F1 score**-F1 score for AD and CD. The metric is used to verify correct model behavior against under-fit and over-fit problems.
- **Accuracy**-measures AD and CD model accuracy.

Detection delay metrics.

- **Detection delay** Δ_1-The metric is computed in number of time series sample points between the start of an attack and detection. This metric is computed for standard AD models where per point detection is made.
- **Detection delta** Δ_2-The metric is computed in number of time series sample points between the start of the attack and the CD identifying a production cycle as compromised.

5 AD Performance in the Autoclave Use Case

We conduct three different experiments in which we compare the performance of the following AD algorithms: (1) Threshold; (2) DBSCAN; (3) SGD, (4) K-Means; (5) SVM; (6) Autoencoder.

The algorithm (1) Threshold serves as a baseline. For this algorithm we assume that the AD system is provided with an exact description of the production cycle. Using a threshold the AD simply compares if a time series point in a cycle ventures too far. As previously described, this ideal scenario is not realistic as AD and ICS are usually not tightly coupled. Thus, all other algorithms (2) to (6) use the provided training data to learn how a normal production cycle should looks like. Simple statistical methods such as (2) to (5) do not require much training data while (6) requires a reasonable amount of training data.

The three experiments, illustrated in Fig 4, have the following aims. In Experiment 1 we establish a baseline; during training all 7 types of attacks are present and 100 training cycles are available. In Experiment 2 we provide the same 100 training cycles and all 7 types of attacks are present. However, the evaluation data set contains modified cycles where a change in environment temperature throughout the year is incorporated. In Experiment 3 the training

[2]Anomaly Detection Evaluation Framework (ADEF) Framework-https://github.com/sbnsix/adef

set does not contain all 7 types of attacks. Thus, the AD algorithms are confronted with unknown attacks in the test data.

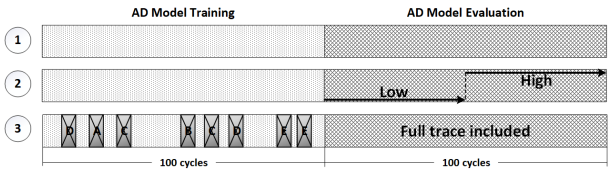

Figure 4: Evaluation trends created to evaluate AD models performance under different long term background temperatures scenarios. 1) Baseline experiment-a simple evaluation with 100 training cycles/100 detection cycles with all known attacks and no background temperature changes. 2) Model skew-in detection cycle trends up and down are added to detection cycles. This experiment aims to check how model is skewed in both directions. 3) Model knowledge-validation of known vs. unknown attacks. Selection of attacks during training phase are removed from training data set to observe whether performance degradation occur.

5.1 Experiment 1-Baseline

In this experiment we consider production cycles without skew due to environmental conditions and all possible attack patterns are visible in the training data. Fig. 5 provides a visualization of training and evaluation data set. The training and evaluation data sets contain 100 generated autoclave cycles of 300 minutes duration each. Each cycle is represented by 300 sample points (the process is tracked every minute). The attack-free elements of the cycle are shown in blue and attacked elements are shown in red. ADEF is configured such that 30% of the cycles are under attack. The type of attack is randomly selected with equal probability chosen from the set of all 7 attacks.

All AD models are trained (or configured with simpler models) using the training data set. As each AD model may have several configuration parameters ADEF automatically tests a large set of parameter combinations against the training data. The best configuration (based on the achieved AUC) for the Receiver Operating Characteristic (ROC) is then used for evaluation of the evaluation data set. It is not guaranteed that ADEF finds the best configuration for each AD model but performs a scan over a large search space. Fig. 6 shows the resulting ROC curve for all evaluated AD models in blue. The blue ROC is created by varying the decision threshold (usually one available configuration parameter for all algorithms) and evaluation the decision result for all 300000 data points (300 samples in 100 cycles).

The CD is evaluated using the found best AD configuration. The window size of the CD is increased from 0 to 300 minutes to create the CD ROC shown in red in Fig. 6. This shows the performance of the CD providing a decision on each of the 100 production cycles contained in the evaluation data. AD provides decisions for each point in a production cycle. CD provides a decision on the entire production cycle.

Tab 1 summarizes the evaluation results. The values for column AUC summarize the result visually provided in Fig. 6. The other columns provide the additional metrics introduced in the previous sections. For $\Delta 1$ and $\Delta 2$ we provide best case, average and worst case.

The results show that the SVM model is the performance winner over other algorithms and the much more complex Autoencoder. We chose a winner based on CD performance as we ultimately need

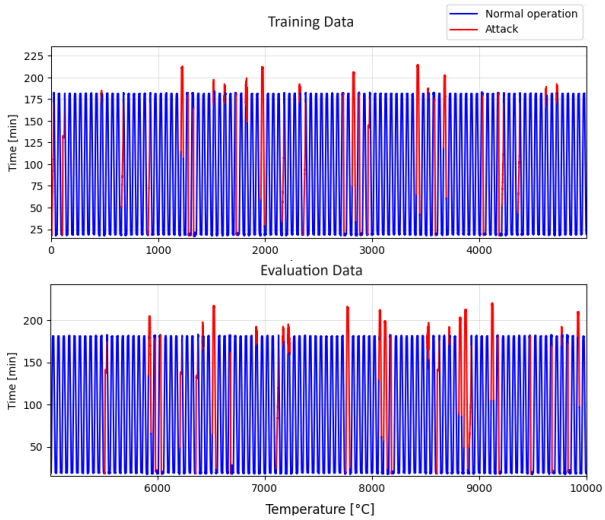

Figure 5: Baseline experiment input data for training and evaluation. (Left) 100 autoclave cycles with 30% faults used in training, (Right) 100 autoclave cycles with 30% faulty cycles used in evaluation phase.

Table 1: AD and CD Model performance metric achieved in baseline experiment.

Type	Model	Metric AUC	ACC	F1	EER	Δ1 [Min, Avg, Max]	Δ2 [Min, Avg, Max]
AD	THB	0.98	0.9	0.69	0.1	[1, 1.27, 2]	N/A
CD	THB	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
AD	AutoE	0.97	0.82	0.56	0.17	[0, 0, 0]	N/A
CD	AutoE	1.00	0.99	0.98	0.03	[2, 2, 2]	N/A
AD	DBSCAN	0.97	0.88	0.65	0.12	[1, 1, 1]	N/A
CD	DBSCAN	1.00	1.00	1.00	0.00	[2, 2, 2]	[13, 26.77, 45]
AD	K-Means	0.96	0.89	0.67	0.11	[1, 1.33, 2]	N/A
CD	K-Means	1.00	1.00	1.00	0.00	[1, 2.46, 9]	[3, 5.73, 7]
AD	SGD	0.93	0.9	0.69	0.1	[1, 1.53, 4]	N/A
CD	SGD	1.00	0.99	0.98	0.03	[1, 1.5, 2]	[172, 186.35, 208]
AD	SVM	0.98	0.9	0.69	0.1	[1, 1.33, 3]	N/A
CD	SVM	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]

to know if a production run is compromised. We use EER and $\Delta 2$ to select the best performing algorithm. While the EER of DBSCAN, K-Means and SVM is zero, SVM achieves a $\Delta 2$ of 2.5 minutes on average. With SVM a compromised cycle is detected on average after 2.5 minutes. It has to be noted that with the Autoencoder a compromise can only be detected after the cycle is completed (300 minutes) as the Autoencoder requires the full cycle data for a decision.

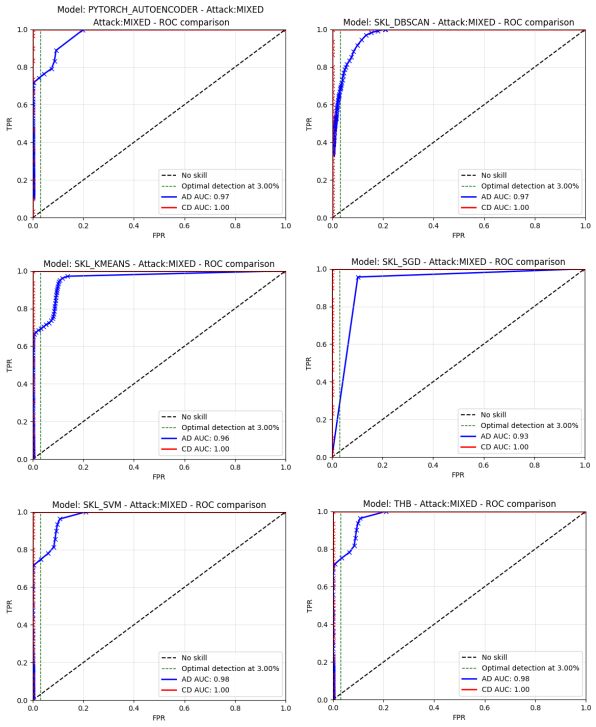

Figure 6: Baseline experiment. ROC curves for AD models (blue) and CD models (red) observed for mixed attack.

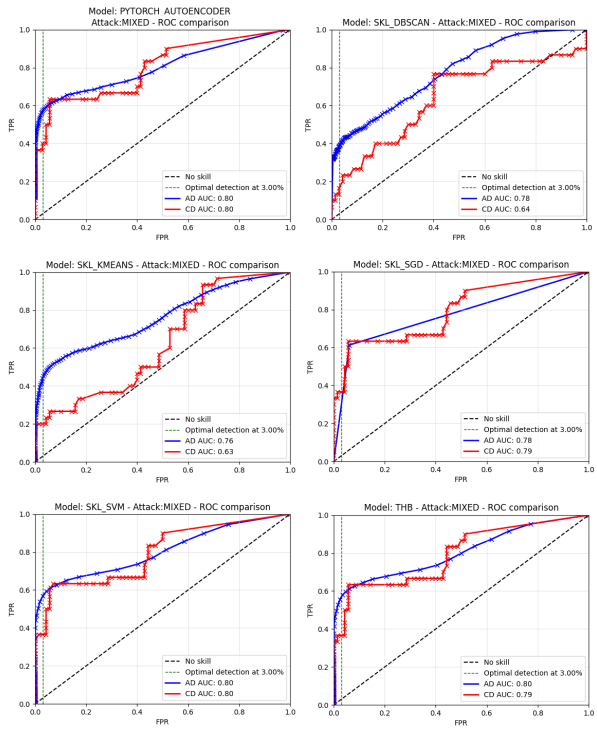

Figure 7: Model skew experiment. ROC curves for AD models (blue) and CD models (red) observed for mixed attack. Model AD performance drop clearly visible in comparison to baseline experiment on both types of AD and CD models. The CD model is under performing AD model performance.

5.2 Experiment 2-Changing Environment

In this experiment we consider a change in the environment. Training and evaluation data sets are generated the same way as described in Experiment 1 with the difference that the evaluation data set now reflects seasonal temperature changes. Two extreme background temperature inputs are added (winter and summer background temperatures) to the evaluation data set, while training data set remains without adverse events. Thus, the autoclave process will start and end at slightly different temperatures than visible to the algorithms during training. This is a realistic problem in industrial settings. Available collected training data may be collected at times with different environmental conditions. The aim of this experiment is to see if this has significant impact on the algorithm performance.

Fig. 7 showcases the obtained AD (blue) and CD (red) ROC curves. Tab 2 shows all performance metrics resulting from this experiment.

There is a clear visible performance drop and the CD is performing worse than the AD. The performance drop is significant and one can conclude from this experiment that training of AD must be frequently repeated when environmental conditions change. It is not practical in an industrial setting to once set up AD and hope it will perform well over a long duration subject to smaller changes in environmental conditions.

Table 2: AD and CD Model performance metric achieved in model skew experiment.

Type	Model	Metric					
		AUC	ACC	F1	EER	Δ1 [Min, Avg, Max]	Δ2 [Min, Avg, Max]
AD	Threshold Based Model (THB)	0.8	0.91	0.29	0.57	[1, 8.22, 29]	N/A
CD	THB	0.79	0.85	0.28	0.72	[1, 8.22, 29]	[41, 59.75, 83]
AD	AutoE	0.8	0.91	0.29	0.58	[1, 8.26, 29]	N/A
CD	AutoE	0.8	0.85	0.28	0.72	[1, 8.26, 29]	N/A
AD	DBSCAN	0.78	0.77	0.36	0.337	[1, 5.18, 14]	N/A
CD	DBSCAN	0.64	0.65	0.34	0.57	[1, 5.18, 14]	[24, 59.06, 88]
AD	K-Means	0.76	0.85	0.33	0.42	[1, 7.56, 28]	N/A
CD	K-Means	0.63	0.52	0.41	0.54	[1, 7.56, 28]	[1, 14.54, 34]
AD	SGD	0.78	0.91	0.29	0.57	[1, 8.22, 29]	N/A
CD	SGD	0.79	0.85	0.28	0.72	[1, 8.22, 29]	[40, 59.67, 83]
AD	SVM	0.8	0.91	0.29	0.57	[1, 8.26, 29]	N/A
CD	SVM	0.8	0.84	0.28	0.7	[1, 8.26, 29]	[41, 59, 82]

5.3 Experiment 3-Unknown Attacks

This experiment evaluates the impact of less examples of attacks available during training. We run the experiment the same way as described in Experiment 1 with the difference that not all 7 attacks present in the evaluation data set are also present in the training data set. The aim is to see how well AD and CD can cope with unknown attacks. In a practical industrial setup it has to be assumed that not all possible attacks can be enumerated and considered.

We carry out 4 experimental runs. For each run we increase the number of attack types present in the training data. We consider 1, 3, 5 and all 7 attacks present; thus, the experiment with 7 attacks present in the training data corresponds to Experiment 1, our baseline.

Table 3: AD and CD Model performance metric achieved in model knowledge experiment.

Model	Metric						
	#Attacks	AUC	ACC	F1	EER	Δ1 [Min, Avg, Max]	Δ2 [Min, Avg, Max]
THB	1	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
THB	3	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
THB	5	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
THB	7	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
AutoE	1	1.00	0.99	0.98	0.03	[2, 2, 2]	N/A
AutoE	3	1.00	0.99	0.98	0.03	[2, 2, 2]	N/A
AutoE	5	1.00	0.99	0.98	0.03	[2, 2, 2]	N/A
AutoE	7	1.00	0.99	0.98	0.03	[2, 2, 2]	N/A
DBSCAN	1	1.00	1.00	1.00	0.00	[2, 2, 2]	[13, 26.77, 45]
DBSCAN	3	1.00	1.00	1.00	0.00	[2, 2, 2]	[13, 26.77, 45]
DBSCAN	5	1.00	1.00	1.00	0.00	[2, 2, 2]	[13, 26.77, 45]
DBSCAN	7	1.00	1.00	1.00	0.00	[2, 2, 2]	[13, 26.77, 45]
K-Means	1	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[1, 5.38, 8]
K-Means	3	1.00	1.00	1.00	0.00	[1, 10.69, 24]	[11, 17.9, 29]
K-Means	5	0.99	0.99	0.98	0.03	[1, 10.14, 26]	[29, 44.11, 59]
K-Means	7	1.00	1.00	1.00	0.00	[1, 2.46, 9]	[3, 5.73, 7]
SGD	1	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
SGD	3	1.00	0.99	0.98	0.03	[1, 1.9, 3]	[164, 174.54, 184]
SGD	5	1.00	0.99	0.98	0.03	[1, 1.5, 2]	[175, 189.43, 208]
SGD	7	1.00	0.99	0.98	0.03	[1, 1.5, 2]	[172, 186.35, 208]
SVM	1	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
SVM	3	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
SVM	5	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]
SVM	7	1.00	1.00	1.00	0.00	[1, 1.5, 2]	[2, 2.5, 3]

Tab. 3 lists all results for the CD. We do not consider here the performance of the AD.

As we can see from the results the SVM model still performs best, even when we reduce the number of attack examples in the training data set. For the Autoencoder the number of attack examples does not matter as it is only trained on the data of attack free cycles (a property of the Autoencoder). However, the Autoencoder never reaches an EER of zero.

We can conclude that a well performing AD/CD can be configured using a very limited set of attack examples. It is not necessary to collect large data sets containing attacks and it is also possible to defend against unknown attacks to the system.

6 Conclusion

This section summarize our main discoveries and highlight impact done by this study. In this work we analyzed the performance of AD models in the context of an autoclave process. The aim was to understand if advanced ML algorithms are required or if more simplistic approaches are sufficient. We found that simple statistical methods such as DBSCAN and SVM outperform a more complex Autoencoder. These methods also work well when only a limited set of attack examples are present and they also can deal with unknown attacks. However, we found that variations in the environment have a significant impact on performance of all evaluated AD models. As environments change in industrial settings (such as environmental temperature considered here) are to be expected frequent training and calibration of the models is required. This poses the challenge of obtaining attack free training data continuously.

ACKNOWLEDGMENTS

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant numbers 18/CRT/6222. For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

References

[1] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. 2020. USAD: UnSupervised Anomaly Detection on Multivariate Time Series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD'20)*. Association for Computing Machinery, New York, NY, USA, 3395–3404. https://doi.org/10.1145/3394486.3403392

[2] Kamal Berahmand, Fatemeh Daneshfar, Elaheh Sadat Salehi, Yuefeng Li, and Yue Xu. 2024. Autoencoders and their applications in machine learning: a survey. *Artif. Intell. Rev.* 57 (2024), 28. https://api.semanticscholar.org/CorpusID:267466773

[3] Ane Bl'azquez-Garc'ia, Angel Conde, Usue Mori, and José Antonio Lozano. 2020. A Review on Outlier/Anomaly Detection in Time Series Data. *ACM Computing Surveys (CSUR)* 54 (2020), 1 – 33. https://api.semanticscholar.org/CorpusID:211075794

[4] Léon Bottou. 2012. *Stochastic Gradient Descent Tricks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 421–436. https://doi.org/10.1007/978-3-642-35289-8_25

[5] Mohammad Braei and Sebastian Wagner. 2020. Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *ArXiv* abs/2004.00433 (2020). https://api.semanticscholar.org/CorpusID:214743362

[6] Mete Çelik, Filiz Dadaşer-Çelik, and Ahmet Sakir Dokuz. 2011. Anomaly detection in temperature data using DBSCAN algorithm. In *2011 International Symposium on Innovations in Intelligent Systems and Applications*. Istanbul, Turkey, 91–95. https://doi.org/10.1109/INISTA.2011.5946052

[7] Julien Herzen, Francesco Lassig, Samuele Giuliano Piazzetta, Thomas Neuer, Leo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan Koscisz, Dennis Bader, Frederick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and Gael Grosch. 2022. Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research* 23, 124 (2022), 1–6. http://jmlr.org/papers/v23/21-1177.html

[8] Heinz Hofbauer and Andreas Uhl. 2016. Calculating a boundary for the significance from the equal-error rate. In *2016 International Conference on Biometrics (ICB)*. 1–4. https://doi.org/10.1109/ICB.2016.7550053

[9] Bryan Hooi. 2019. *Anomaly Detection in Graphs and Time Series: Algorithms and Applications*. Ph. D. Dissertation. Carnegie Mellon University, USA. https://doi.org/10.1184/R1/8337236.V1

[10] Jianmin Jiang and Lasith Yasakethu. 2013. Anomaly Detection via One Class SVM for Protection of SCADA Systems. In *2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2013, Beijing, China, October 10-12, 2013*. IEEE Computer Society, 82–88. https://doi.org/10.1109/CyberC.2013.22

[11] Kwei-Herng Lai, Daochen Zha, Guanchu Wang, Junjie Xu, Yue Zhao, Devesh Kumar, Yile Chen, Purav Zumkhawaka, Minyang Wan, Diego Martinez, and Xia Hu. 2021. TODS: An Automated Time Series Outlier Detection System. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 18 (May 2021), 16060–16062.

[12] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, George H. Chen, Zhihao Jia, and Philip S. Yu. 2022. PyGOD: A Python Library for Graph Outlier Detection. *arXiv preprint arXiv:2204.12095* (2022).

[13] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. 2022. Anomaly Detection in Time Series: A Comprehensive Evaluation. In *Proceedings of the VLDB EndowmentVolume 15, Issue 9, pp 1779–1797*, Vol. 15. 1779–1797. https://doi.org/10.14778/3538598.3538602

[14] Vardhan Shorewala. 2021. Anomaly Detection and Improvement of Clusters using Enhanced K-Means Algorithm. In *2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. 115–121. https://doi.org/10.1109/ICCCSP52374.2021.9465539

[15] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proc. VLDB Endow.* 15, 6 (feb 2022), 1201–1214. https://doi.org/10.14778/3514061.3514067

[16] Hang Zhao, Yujing Wang, Juanyong Duan, Congrui Huang, Defu Cao, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, and Qi Zhang. 2020. Multivariate Time-Series Anomaly Detection via Graph Attention Network. In *2020 IEEE International Conference on Data Mining (ICDM)*. 841–850. https://doi.org/10.1109/ICDM50108.2020.00093

[17] Yue Zhao, Xiyang Hu, Cheng Cheng, Cong Wang, Changlin Wan, Wen Wang, Jianing Yang, Haoping Bai, Zheng Li, Cao Xiao, Yunlong Wang, Zhi Qiao, Jimeng Sun, and Leman Akoglu. 2021. SUOD: Accelerating Large-Scale Unsupervised Heterogeneous Outlier Detection. arXiv:2003.05731 [cs.LG]

[18] Yue Zhao, Zain Nasrullah, and Zheng Li. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research* 20, 96 (2019), 1–7. http://jmlr.org/papers/v20/19-011.html