

Article

Decentralized Real-Time Anomaly Detection in Cyber-Physical Production Systems under Industry Constraints

Christian Goetz *  and Bernhard Humm 

Hochschule Darmstadt— Department of Computer Science, University of Applied Sciences,
64295 Darmstadt, Germany

* Correspondence: christian.goetz@yaskawa.eu

Abstract: Anomaly detection is essential for realizing modern and secure cyber-physical production systems. By detecting anomalies, there is the possibility to recognize, react early, and in the best case, fix the anomaly to prevent the rise or the carryover of a failure throughout the entire manufacture. While current centralized methods demonstrate good detection abilities, they do not consider the limitations of industrial setups. To address all these constraints, in this study, we introduce an unsupervised, decentralized, and real-time process anomaly detection concept for cyber-physical production systems. We employ several 1D convolutional autoencoders in a sliding window approach to achieve adequate prediction performance and fulfill real-time requirements. To increase the flexibility and meet communication interface and processing constraints in typical cyber-physical production systems, we decentralize the execution of the anomaly detection into each separate cyber-physical system. The installation is fully automated, and no expert knowledge is needed to tackle data-driven limitations. The concept is evaluated in a real industrial cyber-physical production system. The test result confirms that the presented concept can be successfully applied to detect anomalies in all separate processes of each cyber-physical system. Therefore, the concept is promising for decentralized anomaly detection in cyber-physical production systems.

Keywords: anomaly detection; cyber-physical production systems; cyber-physical systems; deep learning; unsupervised learning



Citation: Goetz, C.; Humm, B. Decentralized Real-Time Anomaly Detection in Cyber-Physical Production Systems under Industry Constraints. *Sensors* **2023**, *23*, 4207. <https://doi.org/10.3390/s23094207>

Academic Editors: Jun Wu, Zhaojun Steven Li, Yi Qin and Carman K.M. Lee

Received: 13 February 2023

Revised: 17 April 2023

Accepted: 21 April 2023

Published: 23 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Due to the rising complexity of modern processes in manufacturing, the application of cyber-physical systems (CPS) is increasing. A CPS can be described as a combination of an embedded system with sensors and actuators. The system interacts with these to monitor and control physical processes (Figure 1) [1]. Typically, the embedded system requires a communication interface to exchange data with other systems or a cloud. Many of these CPSs are networked to realize complex physical processes in the real world [2]. CPSs combine powerful information technology to monitor and control engineered systems [3].

Modern production systems, which include CPSs, are defined as cyber-physical production systems (CPPS) [4]. These systems are based on two main functionalities, advanced connectivity to ensure real-time data acquisition from the physical world and feedback from cyberspace. CPPSs break with the structure of the typical automation hierarchy to enable intelligent data management, real-time analytics, and enhanced computational capabilities. The control and field levels still exist to ensure the highest performance for critical loops, while the higher levels are more dynamic and decentralized [5].

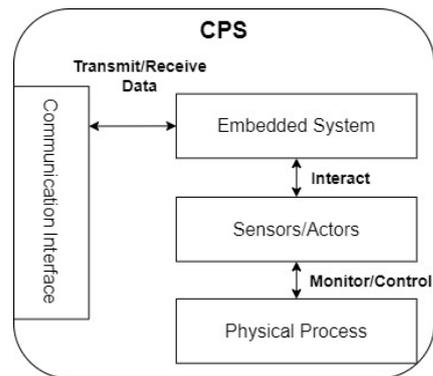


Figure 1. Abstract concept of a CPS [1].

Such a CPPS can be seen in Figure 2. The rotary table dispenser system consists of different CPSs working together to realize several physical processes, e.g., transportation or pick-and-place operations. The overall process involves picking small items from a rotating table and putting them into several containers which are moving around the machine on conveyor belts. After the container is filled and reaches the end position, it gets picked up by the production robot and emptied back onto the rotating table. Thereafter, the container is put in a central location from which the sliding robot places it into the container tray. When the container tray is full, both sliders move to the left side of the system, and the sliding robot sets the container back on the conveyor belt. The described system acts as a simulation of a similar real industrial process and is used as a demonstration unit in Yaskawa. In total, there are nine CPSs, each combining a mechanical and an embedded system. Seven CPSs are based on servomotors and servo controllers. Two CPSs consists of an industrial robot with a robot controller. A central control unit collects data from the different CPSs and regulates the main production process. Additional computational units provide the opportunity to integrate higher functions, e.g., resource planning, production analysis, and process control handling.

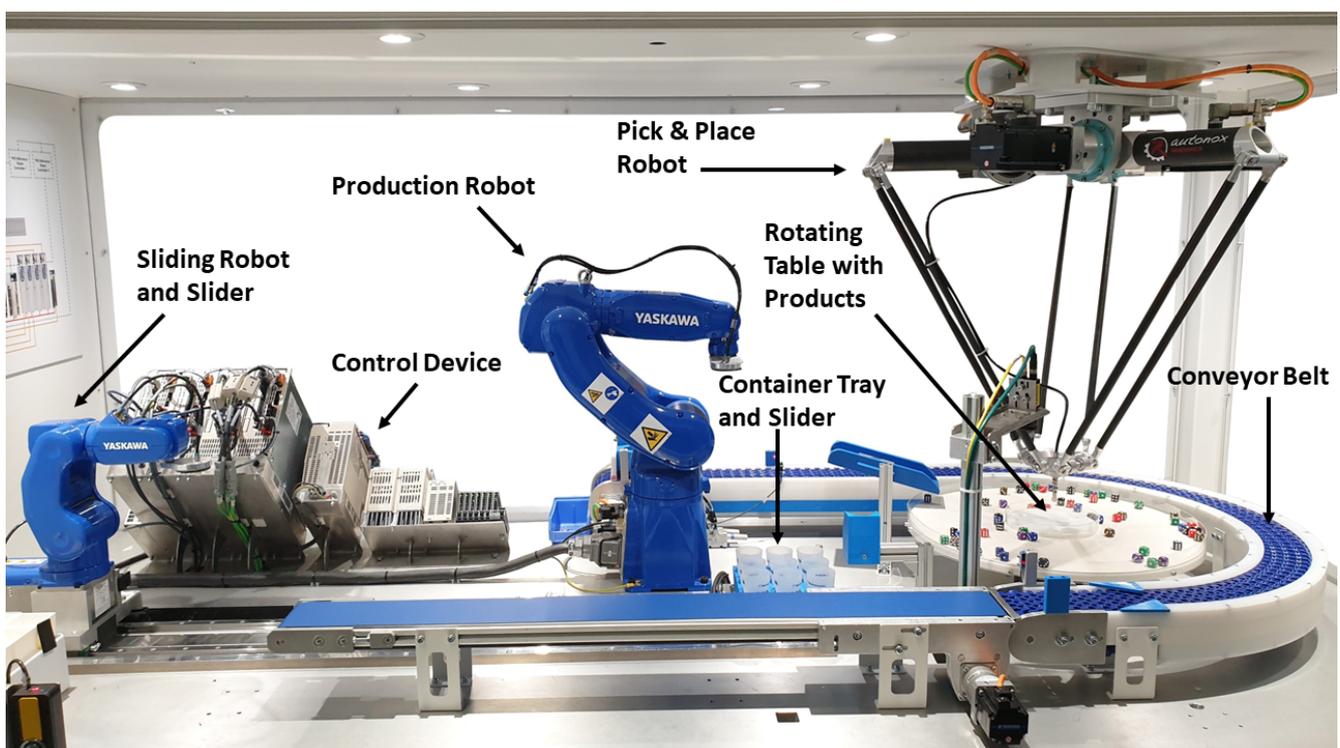


Figure 2. Rotary table dispenser system.

In such a connected structure, even a single failure in one CPS can influence the entire production, resulting in a faulty product, a breakdown of the complete process, or a carryover of the failure through the whole system. Therefore, it is necessary to ensure an error-free operation to realize a secure and modern CPPS [6].

Anomalies can be taken as essential failure indicators, such as a rising vibration at a bearing of the rotating table or an unexpected torque increase on the motor of the conveyor belt. Anomaly detection (AD) in CPPSs refers to the identification of behavior that is not shown under the regular operations of the system. Consequently, by detecting anomalies, there is the possibility to recognize, react early, and in the best case, fix the anomaly to prevent the rise or the carryover of the failure throughout the entire manufacture [7].

Techniques for anomaly detection in CPPS can be distinguished into model-based [8] and data-driven approaches [9]. Model-based methods work based on precise and engineered models of the complete system. Creating such models over the complex structure of CPPS is time-consuming while simultaneously requiring deep expert knowledge. Data-driven approaches establish models only on collected data. Through the high amount of monitored and available data in CPPSs, these approaches are more appropriate for such systems, while additionally, no proper expert knowledge is needed [10]. Recent developments in machine learning and deep learning for anomaly detection have improved the detection performance on complex data sets [11].

By following the scheme to deliver all data from the control and field-level device to one CPS at a higher level to process, analyze, and detect anomalies, current centralized data-driven AD approaches in industrial CPPSs demonstrate better detection abilities than decentralized ones. While this is a significant advantage, it first requires a fully connected and high-performance unit for monitoring all integrated CPSs. Subsequently, adding such a unit increases costs and installation time. Additionally, a centralized concept creates a communication delay between the different stations to exchange the enormous amount of data produced in a CPPS. This can result in a delayed response after detecting an anomaly. Furthermore, it slows down the execution, evaluation, and detection of the anomalies in the individual CPS [12]. The structure of a CPPS is highly dynamic. Often single components, such as motors and sensors, are exchanged, replaced, or modified due to predictive or preventive maintenance. In a centralized approach, this results in a complete recreation of the AD due to the changed characteristics.

In contrast, a decentralized concept addresses these drawbacks by establishing the AD directly in each CPS. While this allows monitoring of the whole system by combining each separate AD, the need for a high-performance unit can be reduced, and the execution and response time can be increased. Furthermore, changes in a single CPS result in only the retraining of the associated AD. By establishing adequate prediction performances in each single CPS, comparable performance to a centralized AD can be reached.

The contribution of this paper is a novel unsupervised, decentralized, and real-time process anomaly detection concept for CPPS under industry constraints. We focus on industrial production processes and common constraints in CPPSs, including real-time requirements, asynchronous signals, prediction quality, configurable design, data-driven limitations, processing limitations, and communication interface constraints.

We employ several 1D convolutional autoencoders (1D-ConvAE) in a sliding window approach to achieve adequate prediction performance and fulfill real-time requirements. Current methods do not consider the limitations and constraints of industrial setups and mainly follow a centralized approach. By executing the installation process on an external, removable device, we increase the flexibility of our concept while considering processing limitations. To meet communication interface and processing constraints in typical CPPSs, we decentralize the execution of the AD into each separate CPS. The installation is fully automated to tackle data-driven limitations. Thereby, no expert knowledge about explicit anomalies is needed. Adjustments to the data collection routine were made to optimize the external sampling procedure and improve the installation process.

This paper is structured as follows. Section 2 summarizes related work about anomaly detection for industrial CPS and CPPS. The problem statement is specified in Section 3. Section 4 presents a concept for fast and decentralized unsupervised anomaly detection in CPPS. Information about a prototypical implementation is provided in Section 5. In Section 6, the evaluation of the approach is presented based on an industrial setup. Finally, a conclusion and an outlook for future work are given in Section 7.

2. Related Work

Surveys on anomaly detection techniques can be found in [13–15]. More industrial-related AD methods are described in [16,17]. Overall, these techniques can be differentiated into model-based and data-driven approaches. Model-based techniques detect anomalies by manually creating precise models about the underlying system. This requires a deep prior knowledge of the individual CPPS. While data-driven approaches are also based on models, those models are generated automatically from data and not manually by domain experts. Furthermore, data-driven approaches can be split into supervised and unsupervised techniques. Anomalous data in CPPS is associated with the undefined behavior of the system. Creating such anomalous data can be hazardous for the CPPS itself, while defining all possible anomalies in advance is nearly impossible. Based on the points mentioned above, we focus on unsupervised data-based methods.

Common approaches in unsupervised data-based AD are one-class classification methods, such as deep one-class networks [18] and one-class support vector machine [19,20]. While multi-class classification techniques typically require labeled datasets, these approaches focus on the normal samples by learning a discriminative hyperplane surrounding them. Other frequently used techniques are unsupervised clustering methods such as Gaussian Mixture Models [21], k-nearest neighbor methods [22], or random isolation forests [23]. These models can identify anomalies by building a detailed representation of the normal data. While the resulting models are generally lightweight and computationally fast, they lack performance when processing high-dimensional data.

Deep learning methods for AD have recently improved the state of the art in detection performance on complex and large datasets [24]. The standard techniques in this field are generative adversarial networks (GAN). GANs consist of a generator combined with a discriminator as the base structure. By teaching the discriminator to distinguish between real and fake samples while the generator tries to generate new data based on the input, GANs can detect anomalies even in large multivariate data streams. Concepts of GANs differ mainly in the models used as the base structure, such as long-short-term-memory (LSTM) recurrent neural networks (RNN) [25], two-dimensional convolutional autoencoder [26], and one-dimensional convolutional autoencoder [27]. While the described approaches achieve good outcomes, they result in highly complex and large models that cannot be applied to a CPS with limited computational resources, which is a common industry constraint.

Reconstruction-based methods in AD combine techniques that rely on the assumption that a model trained only on normal data cannot reconstruct abnormal or unseen data. Typical techniques of these fields are PCA methods [28] or sparse representations [29]. A widely used approach for reconstruction-based anomaly detection in CPS is using autoencoders [30,31] or variants thereof [32,33]. By learning the latent features of the input data, autoencoders can reconstruct their input as output. While these models can be applied to analyze the spatial characteristics of the input data, they miss considering the temporal dependencies, which are necessary indicators for anomalies in the time series data of industrial CPS.

While convolutional neural networks (CNN) were initially developed for solving image classification tasks, they can also be successfully applied for AD in time series data of a CPS through the ability to extract temporal dependencies [34]. Several industrial applications of CNNs in CPS, such as fault detection in motors [35], AD in wheelset bearings [22], and rolling bearings [36], can be found. Additionally, ref. [37] pointed out that CNNs have

lower parameters than other network structures while performing comparably or better, resulting in reduced complexity, needed storage capacity, and computing power [38].

Convolutional autoencoders (ConvAE) combine the ability to detect temporal anomalies with the help of convolutions and spatial anomalies by the autoencoder structure while being also resource-efficient. This results in ideal models for AD in multivariate time series data [39–41]. Using a 2D variational ConvAE, the authors in [42] detect anomalies from unseen abnormal patterns in industrial robots. In [43], a ConvAE based on channel-wise reconstruction in combination with a local outlier factor is used to detect anomalies in automobile sensors.

Several approaches for decentralized AD can be found [44–46]. In [12], different decentralized AD techniques are analyzed and compared in complexity and performance. A decentralized approach for real-time AD in transportation networks is introduced in [47]. The authors of [48] presented spatial anomaly detection in sensor networks using neighborhood information. While these are promising approaches for decentralized AD, no work considers all the different industrial constraints simultaneously, which is important for integration into a CPPS.

Different automated frameworks for anomaly detection can be found. In [49], a framework for automatic time series anomaly detection is introduced. The study focuses on large-scale time series data in a centralized AD approach, which cannot be applied to a CPPS with limited resources. The authors of [50] introduce an unsupervised framework for anomaly detection in CPS. Furthermore, ref. [51] presents a high-performance unsupervised anomaly detection for CPS networks. Both approaches are developed for CPS, but mainly focus on adversarial attacks and not on the process of the CPS and, respectively, of the CPPS.

In our previous work [52], we introduced an unsupervised anomaly detection concept for CPSs under industry constraints while focusing on repetitive tasks with a fixed duration for a single CPS. In this contribution, we improved the concept for CPPS with multiple CPSs, while still considering all industrial constraints. We adapted the technology to a sliding window approach to simultaneously handle processes with variable durations and meet real-time instead of near-time requirements.

In summary, there are several approaches for centralized and decentralized data-driven unsupervised anomaly detection. Only a few are evaluated in real CPSs, and even fewer are applied to real production data of a CPPS. Overall, no work considers all the different industrial limitations of a CPPS while following a decentralized and fast approach to realize anomaly detection in industrial production data.

In this work, we propose a concept that addresses all the requirements that must be considered to realize a usable decentralized, real-time anomaly detection in CPPS under industrial constraints. Our contribution in this paper is summarized as follows. We employ several 1D-ConvAEs for unsupervised anomaly detection in a CPPS to monitor the different processes. We introduce a novel concept to decentralize the different models in each single CPS of the CPPS by splitting the installation and execution of the anomaly detection to meet industrial requirements. While the concept is fully automated, no expert knowledge about explicit known anomalies is needed to meet the defined requirements.

3. Problem Statement

This article aims at a decentralized concept for real-time unsupervised anomaly detection for production processes under industrial constraints. The problem statement can be described by the different industrial requirements that must be considered to implement such a concept. Several conditions are adapted and extended from [52].

1. **Anomaly detection:** An anomaly detection for a CPPS, such as an industrial production system, shall be performed. The CPPS consists of multiple CPSs producing multivariate time series data over variable process lengths, for example, the sliding robot from the CPPS in Figure 2, combining a robot with several axes and a robotic controller to move containers on a conveyor belt.

2. **Real-time:** To cover all different kinds of anomalies and react even in time-critical scenarios, such as detecting collisions in the production system, the result and reaction of the anomaly detection should be available as quickly as possible. Therefore, the execution of the anomaly should be performed during production, and the results must be immediately provided after new data from sensors and actors are available, e.g., a few milliseconds after the data is received.
3. **Prediction quality:** For an AD application in an industrial environment, adequate prediction performance is required. This depends on the different use cases for which the anomaly detection is applied, e.g., an F1 score of 0.95 or better for each CPS in the CPPS.
4. **Configurable:** To apply AD on different CPPSs in different applications, the anomaly detection should be adaptable to various CPSs and use cases. The possibility of using the technique for varied time series data with different variable types and diverse time lengths should be given, for instance, robots or transportation systems with features such as torque, position, and speed.
5. **Data-driven:** As mentioned before, manually creating models is time-consuming and requires deep expert knowledge. Simultaneously recording anomalous data from CPPS can be dangerous for the system itself. Therefore, the AD should only be trained with regular production data and without expert knowledge.
6. **Feasible:** The AD should be compatible with current technological standards in industrial environments to realize a generalist integration for various scenarios. This includes constraints and limitations of commonly used CPPSs in production settings:
 - (a) Process limitations, due to the design of CPSs in industry, that are unable to execute process-intensive tasks in parallel to control and monitor the physical process, e.g., limited available RAM and processing power.
 - (b) Communication interface constraints of commonly available CPSs in industry, e.g., OPC UA Communication, to transfer the high amount of production data at a sample rate of 2 ms during the sampling process to a database.

4. A Concept for a Fast, Decentralized, and Unsupervised Anomaly Detection in CPPSs

4.1. Overview

This section describes a fast and decentralized process anomaly detection concept based on several 1D-ConvAEs, which fulfills the requirements specified in the problem statement. Figure 3 shows the sequence of the different steps that are carried out. The concept consists of one AD Installation Cycle, which triggers the creation of several anomaly detection pipelines (AD pipelines) through the parallel execution of AD Generation Cycles, as shown in Figure 4. A detailed description of the AD Generation Cycle can be found in Section 4.3 and in Figure 5. The number of different AD Pipelines depends on the number of included CPSs in the CPPS. In the AD Production Cycles, located in every CPS in Figure 4, each pipeline is directly implemented and executed as part of the CPS. Explanations about the AD Production Cycle can be found in Section 4.4 and in Figure 6. The processing unit backend, an external device that can be removed after the installation process is finished, performs all heavy processing tasks of the AD Installation Cycle to meet the previously explained industrial constraints of the CPPS. The concept is developed to be executed automatically, enabling AD implementation without deep expert knowledge. In addition, a direct explanation of the individual components of the diagrams can be found in Appendix A.

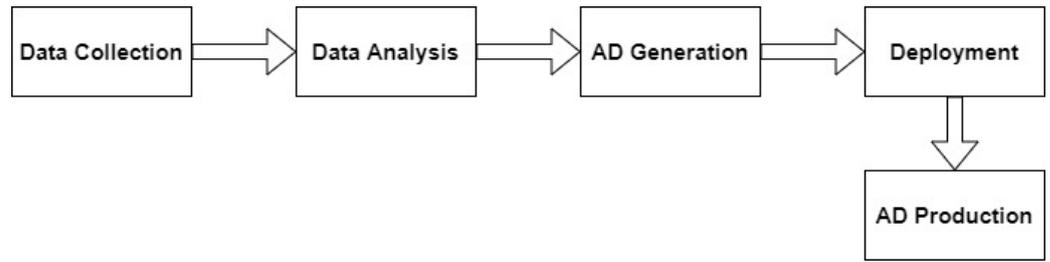


Figure 3. Sequence of steps performed in the concept.

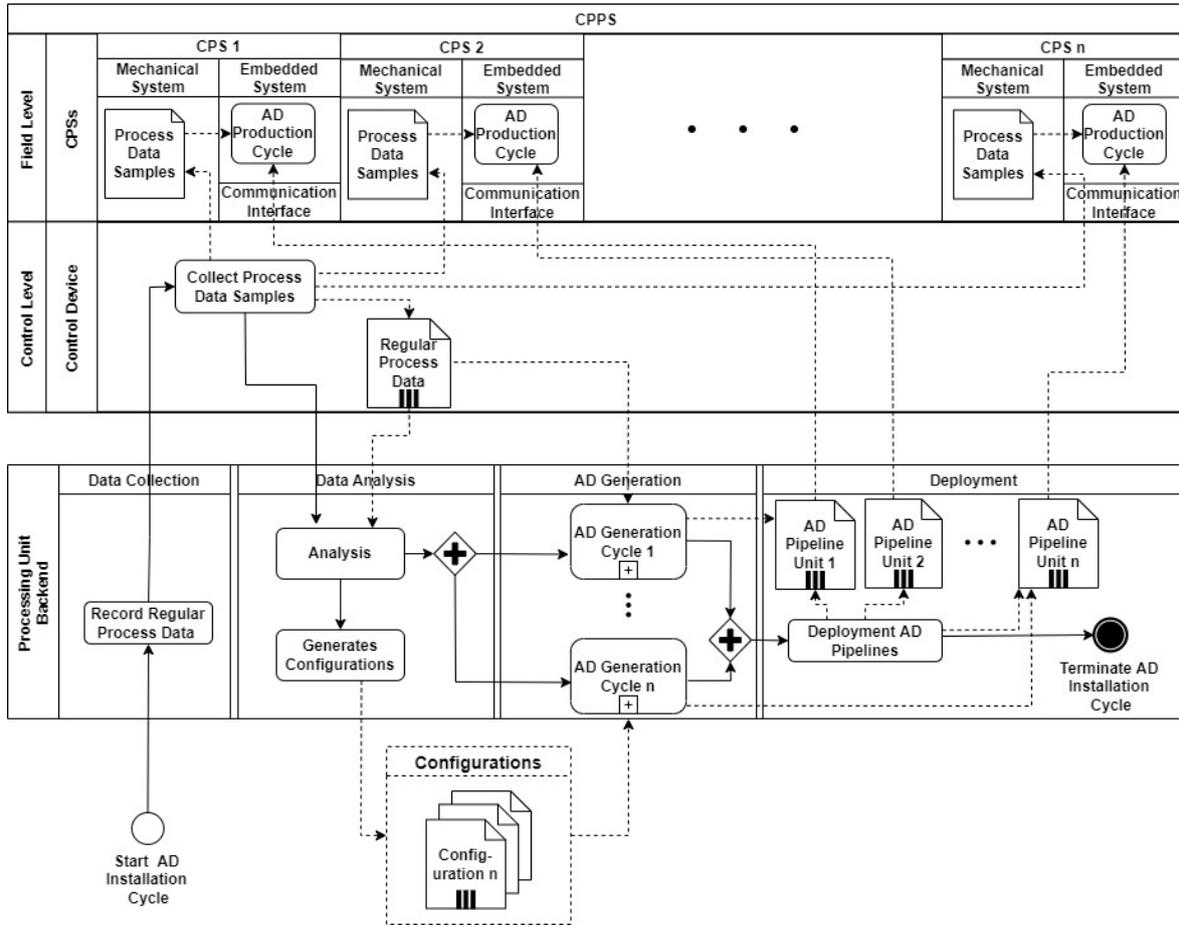


Figure 4. Overview of AD Installation Cycle.

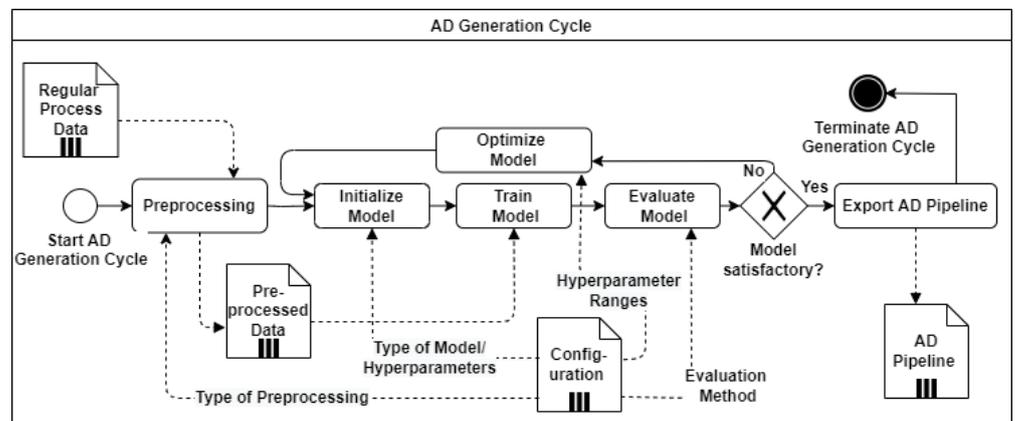


Figure 5. Overview of AD Generation Cycle.

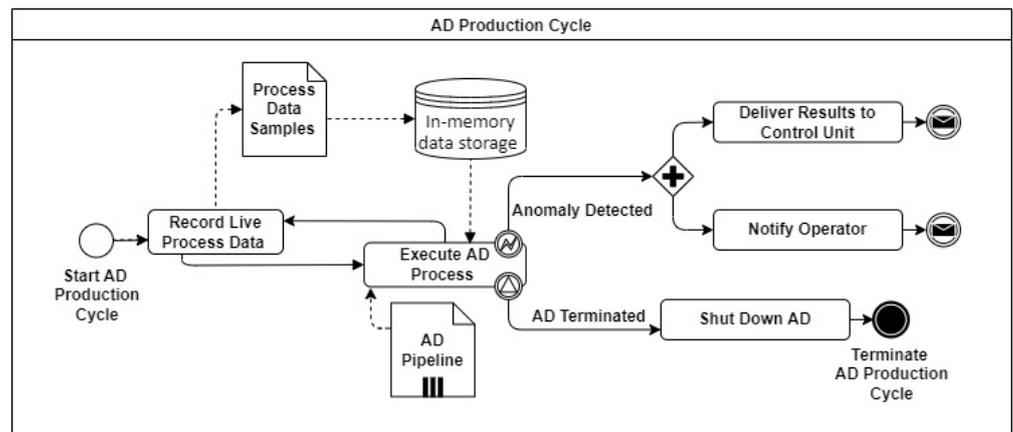


Figure 6. Overview of AD Production Cycle.

4.2. AD Installation

The AD Installation Cycle consists of four parts: data collection, data analysis, AD generation, and deployment (see Figure 4).

Data collection: The operator triggers the data collection at the processing unit backend to record regular process data. Process data samples, single packages of time series data from the individual CPS, are collected at a high sample rate and sent to the control device. Over a defined period of time, the individual data of the various CPSs are recorded and then combined. The resulting package, named regular process data, is then sent to the processing unit backend. This procedure is required to meet the communication interface limitations in the installation process and enable the use of the high sample rates at the AD Production Cycles directly in the CPSs. The data packages are saved inside the processing unit backend until a specified number of records is reached. Regular process data consist of different features like position, torque, and speed sampled in the form of time series data from the various CPSs. This data can be defined as multiple data streams containing the features of the physical process recorded by the different sensors and actors.

Data Analysis: Depending on the diverse CPSs, different features with different ranges are provided. In the analysis step, unnecessary features are automatically removed, and configuration files are accordingly generated. Each configuration file contains the necessary information for the following AD generation cycle, e.g., feature ranges, types, and default hyperparameters. The operator can manually tune this information, or the default values can be used.

AD Generation: In the installation step for each included CPS, an AD Generation Cycle (Figure 5) is triggered. The different AD Generation Cycles can be executed in parallel to speed up the installation process. A detailed description of the AD generation cycle can be found in Section 4.3.

Deployment: After the generation of the AD pipelines, each pipeline is exported and deployed to the separate CPS. This terminates the AD Installation Cycle.

4.3. AD Generation Cycle

The AD Generation Cycle consists of preprocessing, model initialization, training, evaluation, optimization, and export, as shown as a BPMN diagram in Figure 5. First, the provided regular process data, the combined collected data samples of all CPS, are preprocessed with the information received from the configuration files. This transforms the data, which consist of different ranges and units, into an equal numerical range. The type of the desired preprocessor is defined in the configuration file. This enhances a configurable setup, which can handle various variables with different units and ranges. Next, the model is initialized, trained, evaluated, and optimized. Additional hyperparameters set in the configuration file are, e.g., the number of layers, filters per layer, used loss function, and type of optimizer. After initialization, the model is trained on the preprocessed

data. The method specified in the configuration file is used to evaluate the model. In the optimization step, the hyperparameters are changed, influenced by the defined ranges and tuning parameters. The search algorithm declared in the configuration file searches over a generated search space for the best possible parameters. These steps are executed iteratively until the specified reconstruction performance (e.g., the desired MAE Value) is reached. After the tuning is finished, the AD pipeline, a combination of preprocessor and model, is exported to the deployment step. This terminates the AD generation cycle.

4.4. AD Production Cycle

After the AD Generation Cycle is finished and the AD pipeline is deployed in the CPS, the AD Production Cycle, shown as a BPMN diagram in Figure 6, starts. Process data samples, single packages of time series data from the CPS, are collected at a high sample rate and stored in an in-memory data storage. When the required amount of data packages to execute the AD process step is reached, the data are preprocessed and evaluated by the AD pipeline. After the execution, the previously collected data in the in-memory data storage will be released to limit the needed memory capacity. The AD process step will be executed again immediately after enough data is available. In case of an anomaly, the detection can be delivered to the control unit, or the operator can be directly notified. Additionally, the AD process can be terminated, and therefore the AD Production Cycle.

4.5. Sliding Window Convolutional Autoencoder

To achieve adequate prediction performance and meet real-time requirements, we choose a sliding-window-based 1D-ConvAE as the model type (see Figure 7). Autoencoders are reconstruction-based neural networks that reconstruct their input as output. By only learning the reconstruction of the regular pattern, every datum consisting of unseen, abnormal patterns cannot be correctly reconstructed, which will result in a higher reconstruction error. To gain adequate prediction performance and meet the processing limitations, 1D convolutional layers are used. Adding these layers to the autoencoder allows the model to learn spatially invariant features and capture spatially local correlations from the data. This means it can recognize patterns of high-dimensional data without requiring feature engineering. At the same time, the required parameters and the computational complexity of a 1D convolutional layer are significantly lower than the comparable 2D convolutional layers. The 1D-ConvAE can be trained without expert knowledge or explicitly known anomalies, only with regular process data. This fulfills the requirement 5, *data driven*. A detailed comparison with other methods can be found in Appendix B.

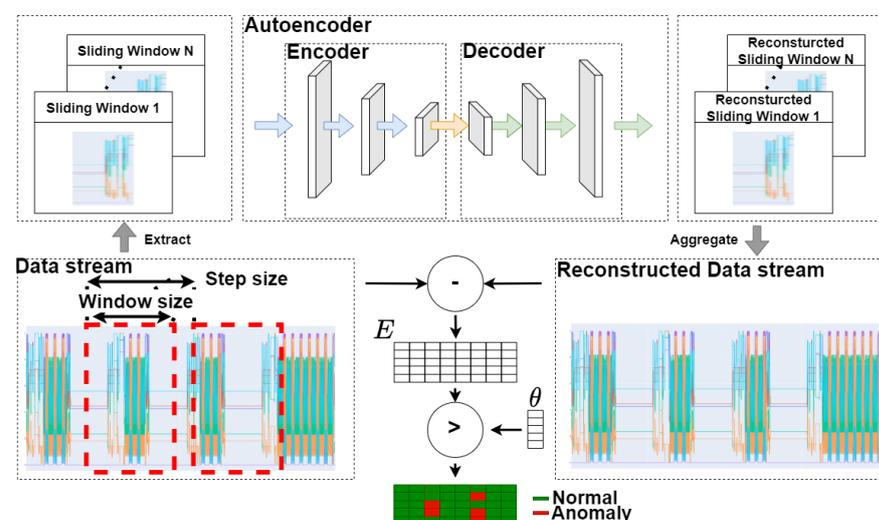


Figure 7. Convolutional autoencoder.

4.6. Anomaly Detection

Regular process data can be defined as a data stream containing several time series of sampled features $F = (f_1, f_2, \dots, f_n)$, where n defines the number of different features such as position, speed, and torque from the various sensors of the mechanical system. The data stream is split into several windows depending on the chosen window size m and step size s . Each window consists of several time series equal to n different features in the data stream over a time period corresponding to the window size m . These generated sliding windows act as the input to the model. The output of the model is each separated reconstructed sliding window. With the help of an aggregation function (e.g., arithmetic mean), the reconstructed sliding windows can be merged into a reconstructed data stream. To calculate the reconstruction error matrix E , the reconstructed error e_{f_i} of each feature f_i at every time step t can be calculated as the Absolute-Error (AE) $e_{f_i} = |f_{i,t} - \hat{f}_{i,t}|$ ($1 \leq i \leq n$) between the input and output. This results in a matrix E representing each feature at each time step as a value of the differentiation between the input and reconstructed data stream. Threshold values must be defined to evaluate which value a reconstruction error indicates if an anomaly is detected. We employ the following method for automatically computing and tuning the threshold values. After training the model, the described method re-evaluates all training data. This results in an error matrix over the whole training data stream. The maximum reconstruction error of each feature is taken from this matrix to construct a threshold vector θ . This vector can be adapted when the model is integrated directly into the CPS by automatically tuning the values in the live testing stage. In the AD production cycle, after enough high sample data are collected in the in-memory storage, each column of the reconstruction matrix E is evaluated with the threshold vector θ . The number of collected data samples can be flexibly chosen but must be at least twice as large as the window size to allow the reconstruction concept to be applied. Suppose a value of e_{f_i} , where i is the considered feature of the total features and n exceeds the associated threshold value of this feature θ_{f_i} ($1 \leq i \leq n$). In that case, the data point in the input data stream is declared anomalous. Therefore, anomalies in the input data stream can be detected by applying the threshold vector θ to each timestep t of the reconstruction matrix E .

5. Prototype Implementation

The concept has been implemented prototypically. As programming language, Python 3.9 is used. A MongoDB <https://www.mongodb.com/> (accessed on 12 February 2023) is established on the processing unit backend to save and export the regular process data. As a preprocessor, a MinMaxScaler was generated. The model is implemented using the Keras library <https://keras.io/> (accessed on 12 February 2023), running on top of Tensorflow <https://www.tensorflow.org/> (accessed on 12 February 2023) [53]. For hyperparameter tuning, the python library Ray Tune <https://ray.io/> (accessed on 12 February 2023) [54] is used. Finally, a tracking server based on the library Mlflow <https://mlflow.org/> (accessed on 12 February 2023) [55] was established to track the training results. The communication between the motion controller and the processing unit backend was realized through an OPC UA server–client model based on publish–subscribe routines. Several function blocks for buffering the high sample process data from the CPS at the motion controller were developed to establish this concept. This enables an intelligent communication pattern, where only minor changes on the motion controller must be performed to allow the described data exchange. The configuration files are written in YAML and can be accessed and changed by the operator. For each CPS, a separate configuration file is created. These files are also tracked to enable a traceable process at a later stage. The preprocessor and model integrations are developed as interfaces to satisfy the configurable requirement. Therefore, various considered models and preprocessors can be implemented as long as they follow the abstract class structure, making it easy to exchange, adapt, or evolve the described technique. To visualize the detection results and allow the user to interact with the system, a dashboard for bi-directional communication between the CPPS and the operator was implemented.

6. Evaluation

6.1. Experimental Setup

The rotary table dispenser system shown in Figure 2 was used to evaluate the decentralized concept. The CPPS consists of different CPSs and a control unit working together to realize several processes, e.g., transport and pick-and-place operations. The overall process involves picking small items from a rotating table and putting them into several containers which are moving on conveyor belts around the machine. During the process, the different time series data of each CPS is collected in the motion controller. Several buffers are written in the motion controller to adapt the high sample rate of 2 ms of each CPS to the minimal data exchange cycle time of 50ms at the OPC UA server. After one buffer is filled, the data package is sent to the OPC UA Server running on the processing unit backend, a pc type NUC8i5BEK. The collected data are saved in the established database after each import cycle. In total, 33 different time series over a period of 6 min were recorded. Based on the high sample rate, each time series consists of around 176,000 samples, resulting in approximately 5,808,000 data points as training data.

6.2. Data Recording

Realistic fault data were generated by forcing different anomalies into the normal process to evaluate the performance of the used models. The resulting deviations from the normal process were manually classified as anomalous areas in the resulting data stream to rate the performance. Five error cases were defined, and at least one error case was generated for each CPS. Additionally, long-term tests, including several complete processes without anomalies, were carried out to control the resulting models in the normal industrial setup.

1. **Friction:** To simulate friction, which can result from abrasion of used mechanical components, delayed maintenance, or broken parts, external forces were applied to the mechanical systems of the different CPSs, e.g., against the rotation direction of the conveyor belt or the movement of the linear sliders. This results in increased torque values at the applied CPS.
2. **Vibration:** Undefined vibration, which can be caused by broken bearings or loose attachments, was applied to the mechanical system of the CPS. The simulation was done by manually applying shocks to the rotating table.
3. **Defect components:** Another industry-related anomaly can be caused by defect components in the production process, such as a broken container. To examine this type of anomaly, different containers were manipulated in such a way that they could not be picked by the robots anymore, resulting in an undefined status of the whole production line.
4. **Incorrect process:** In addition, external manipulations can influence industrial production lines. These injections in the normal process can result in some undefined behavior of the system, which can cause damage to the products or the system itself. To simulate this kind of anomaly, the placement of the containers on the belt was changed in the running process. Therefore, the real positions differ from the fixed pre-defined positions in the machine scope.
5. **Collision:** Due to external influences or process errors, even in modern industrial systems, collisions may occur. The system typically detects heavy collisions, whereas smaller collisions resulting in damaged products or fragile components are mostly not recognized by the internal system. This can be, for example, a collision with an obstacle in the moving path of the linear sliders or a displaced product on the conveyor.

6.3. Model Configuration

The sampled data from the regular process was used to train the model. A MinMax Scaler was chosen to preprocess the data by scaling the time series between zero and one. An Adam optimizer was used, and the loss function was set to MAE. The default

hyperparameter tuning results in 80 different decoder and encoder structures for each CPS. The best model was automatically picked by evaluating the number of parameters and the resulting loss value. Detailed information about the different considered parameters can be found in Table 1. The focus was on realizing small and efficient model architectures to meet the computational limitations (Section 3, point 6). Therefore, shallow structures with a limited amount of parameters were preferred. By comparing all achieved loss values and the resulting model structures, the smallest structure that achieved a low loss value and, thus, a good reconstruction capability was automatically selected. Due to the unsupervised setup, no anomalous data are available in the training process. Therefore, an immediate evaluation of the detection performance is not possible; consequently, only the reconstruction capability can be taken as an additional selection criterion in this process. Additional discussion on the selection process can be found in future work. As activation function, the rectified linear unit was chosen. Dropout layers were applied as regularization between the convolutional layers, and max pooling layers were used to reduce the dimensionality. Different step sizes in the training process were tested. The best results were reached with a step size of one.

Table 1. Summary table of all parameters taken in the process of automatically selecting the models.

Model Parameter	Range	Definition
Number of Layers	[4, 8]	Total number of layers used in the model.
Number of Filters in the first Layer	[32, 128]	The number of filters used in the first layer of the model. To realize the dimensionality reduction, the inner layers have fewer filters. (In the automated concept, half of the previous layer).
Window size	[32, 128]	Number of time steps of the sliding window.
Step size	[1, 64]	The length of the sequence shifted between the individual windows.
Patience	[1, 10]	Number of epochs with no improvement after which training will be stopped.
Total number of parameters	[12,642, 208,614]	Total number of parameters of the resulting model.
Mean absolute error	[0.002, 0.3]	Achieved mean absolute error between input and output at the end of training.

6.4. Experimental Results

This section validates the described concept applied in the experimental setup against the requirements defined in the problem statement.

1. **Anomaly detection:** Figure 8 shows some of the forced anomalies in the experimental setup, illustrating the detection performance of the generated models. In the pictures, the detected anomalies are marked with red points, while the pre-defined anomalous areas are indicated by the red background color of the figure. Combined with the results in Table 2, this confirms that the different models can be successfully applied to detect anomalies in the CPSs.
2. **Real-time:** The evaluated sliding window sizes from the hyperparameter tuning were between 32–64, resulting in comparably small windows. To ensure a fast detection in the real process, each generated sliding window was treated as a data stream and evaluated immediately. With a sample rate of 2 ms, the overall time to collect one window as input data for the model is between 64 and 128 ms. The average execution time per reconstruction and verification for anomalies was around 34 ms, with a maximum of 49 ms and a minimum of 22 ms. Therefore, anomaly detection can be carried out with a maximum delay of 177 ms at our setup, which allows an immediate reaction of the system on detected anomalies.

3. **Prediction quality:** The F1 Score is used to evaluate the model performance. The detailed performance for each CPS is shown in Table 2. To calculate the F1 Score, the manually forced anomalies were classified as anomalous areas. If an anomaly in a window was detected, the used window was assigned as anomalous and evaluated against the area. By reaching high F1 Scores above 0.95, adequate prediction performances for every single CPS are realized. This confirms that the automatically created models for each CPS can reliably detect anomalies in the given CPPS.
4. **Configurable:** The described concept and resulting anomaly detection can be configured for various applications. Only minor changes must be made to the motion controller to enable the sampling process. The automatically generated configuration files can be manually changed, or the default values can be used.
5. **Data-driven:** The models are trained only with the regular process data. Therefore, no anomalous data or feature engineering is needed. No values are added or changed. All removed features are automatically declared. Only the data from the sensors and actors of the CPSs are used. The model is created in an automated way by the configuration file without the need for expert knowledge.
6. **Feasible:** The method utilized standard communication technologies of common industrial setups. By outsourcing the process-intensive tasks to the processing unit backend, the concept enables the application of anomaly detection for the CPPS, even with the processing limitation and constraints of each CPS. In our experimental setup, the simulated process reaches a maximum consumption of 350MB while not exceeding a maximum of 12% CPU load.

Based on the experimental results, the introduced novel concept fulfills all the defined industrial requirements of the problem statement in Section 3.

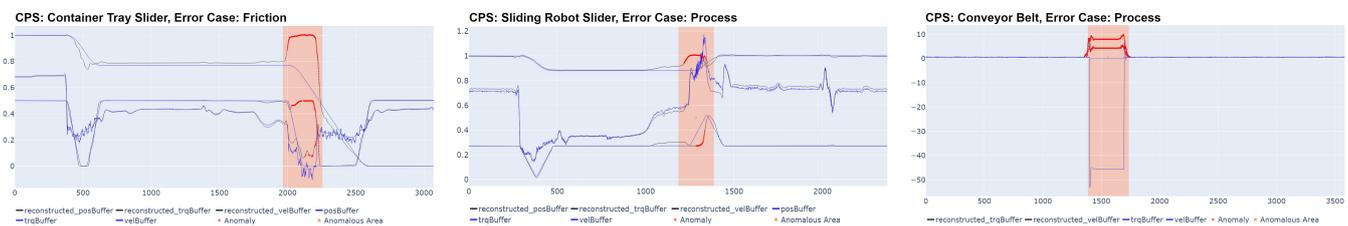


Figure 8. Anomalous Samples.

Table 2. Performance Evaluation.

Unit	TP	TN	FP	FN	Precision	Recall	F1-Score
CB	232	3938	7	10	0.9707	0.958	0.964
RT	193	3199	9	8	0.955	0.960	0.957
SR	22	3002	2	0	0.916	1	0.956
PR	22	3002	1	1	0.956	0.956	0.956
P&P S	484	2965	21	20	0.958	0.960	0.959
P&P U	484	2967	29	15	0.943	0.969	0.956
P&P L	484	2964	27	22	0.947	0.956	0.951
CTS	75	3199	3	3	0.961	0.961	0.961
SRS	231	3374	8	10	0.966	0.958	0.962

CB = Conveyor Belt; RT = Rotating Table; SR = Sliding Robot; PR = Production Robot; P&P S = Pick & Place Robot S Axis; P&P U = Pick & Place Robot U Axis; P&P L = Pick & Place Robot L Axis; CTS = Container Tray Slider; SRS = Sliding Robot Slider.

7. Conclusions and Future Work

This paper presents a fast and decentralized anomaly detection concept for CPPS under industry constraints. The concept is configurable and feasible to apply anomaly detection in different use cases under the limitations of commonly used CPPSs in industrial environments. Due to the decentralization, no additional computational units must be

integrated. The generated models allow a fast and performant integration. The anomaly detection is executed, and evaluations are carried out immediately during production. The model is generated and tuned in a fully automated fashion. No expert knowledge about anomalous data is needed. Overall, the experiments show that each model achieves stable and accurate results. This presents a promising approach for decentralized and fast anomaly detection in CPPSs under industry constraints.

However, despite the apparent success of the concept, there are several directions for future research. In this work, the concept was only tested in a single CPPS with a limited amount of CPSs. Therefore, more studies with different models, more CPSs, and under different scenarios will be performed in future work. Secondly, the models are only evaluated against the defined simulated computational resources and data storage limitations of the used CPSs. This is mainly caused by the integration limitations of the available CPSs. To integrate the models, adaptations to the hardware and software of the CPSs must be carried out in the future. Additionally, several anomalies which can emerge in a CPPS cannot be detected, e.g., process anomalies such as changing the overall process to fewer containers as in the learning process. This forces the CPPS not into an undefined state, although the actual process differs from the learned process. Therefore, another research direction in the future is to adapt the concept even to detect this kind of anomaly. Furthermore, the selection of the model is only based on two parameters, the achieved loss value and the resulting model structure. Despite the good results obtained in the tests with the defined anomalies, this method cannot guarantee the selection of the best model. Further approaches and concepts for a better evaluation of the models and a guaranteed choice of the best model must be found.

Finally, up to now, the output of the anomaly detection is the identification of the anomaly, defined by the time and feature, in the data stream. Adding more information may be helpful to increase the accuracy of the AD for the operator. Ways to gather and provide this additional context information will be evaluated and investigated.

Author Contributions: Conceptualization, C.G.; methodology, C.G.; software, C.G.; validation, C.G.; formal analysis, C.G.; investigation, C.G.; resources, C.G.; data curation, C.G.; writing—original draft preparation, C.G.; writing—review and editing, C.G. and B.H.; visualization, C.G.; supervision, C.G. and B.H.; project administration, C.G.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Limited accessibility to the dataset can be given in single cases.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A

Tables A1–A3 lists each individual item of Figure 4–6 with a short description.

Table A1. Definition Table BPMN Diagram AD Installation Cycle Figure 4.

Item	Definition
Processing Unit Backend	The processing unit backend, an external device that can be removed after the installation process is finished, performs all heavy processing tasks in the AD Installation cycle to meet the previously explained industrial constraints of the CPPS.
Control Device	Unit which typically controls the industrial process.
Communication Interface	The interface of the embedded system to exchange data with the control device or the processing unit backend.

Table A1. *Cont.*

Item	Definition
Embedded System	Part of the CPS which interacts with sensors and actors to monitor and control the mechanical system.
Mechanical System	Summarizes all mechanical components of the system.
Process data samples	Single packages of time series data from the individual CPS. Process data samples consist of features like position, torque, and speed sampled as time series data from the CPS.
Record Regular Process Data	Combined process data samples of all CPS collected from the normal process sampled over a defined time.
Collect Process Data Samples	Process data samples at a high sample rate are collected from the different CPS, combined, and sent to the control device as a data package.
Analysis	In the analysis, unnecessary features are automatically removed from the data, and important information like feature range and data types are collected.
Generate Configurations	Based on the analysis, configuration files are generated. The operator can manually tune this information, or the default values can be used.
AD Generation Cycle	Main cycle to create the preprocessor and train the model.
Deployment AD Pipelines	Each generated AD pipeline is exported and deployed to a separate CPS.
AD Production Cycle	Live integration and execution of the AD pipeline in the individual CPS.

Table A2. Definition Table BPMN Diagram AD Production Cycle Figure 6.

Item	Definition
In-memory data storage	A fast and effective data store that caches live data until it is passed to the AD pipeline for processing.
Record Live Process Data	Live process data is sampled at a high sample rate to an in-memory data storage to collect the needed data to execute the AD pipeline.
Execute AD Process Step	The collected live data is preprocessed and evaluated by the AD pipeline.
Deliver Results to Control Unit	The AD output can be delivered from the CPS to the control unit.
Notify Operator	Depending on the CPS, the Operator can be immediately notified by the separate CPS.
Shut Down AD	In this step, the whole AD production cycle can be switched off to free resources and stop the anomaly detection.

Table A3. Definition Table BPMN Diagram AD Generation Cycle Figure 5.

Item	Definition
Regular Process Data	Data collected from the normal process of the CPS over a defined time.
Preprocessed Data	Transformed and scaled regular process data by the chosen Preprocessor.
AD Pipeline	A combination of initialised Preprocessor and trained model.
Configuration	Contains necessary parameters for the separate steps of the generation cycle, e.g., the number of layers, filters per layer, loss function, and type of optimizer. Default parameters are automatically provided but can also be manually changed and tuned.
Preprocessing	In the preprocessing step, the regular process data is transformed by the chosen preprocessor. This scales the data provided, which normally consists of different ranges and units, to an equal numerical range.

Table A3. *Cont.*

Item	Definition
Initialize Model	Here, the model is built based on the configuration. Therefore, the number of layers, filter, and type of each layer and the optimizer and loss function are set.
Train Model	In this step, the initialized model is trained with the preprocessed regular process data.
Evaluate Model	Depending on the evaluation method defined in the configuration step, the model is tested, the results are tracked, and the complete experiment is saved.
Optimize Model	In the optimization step, the hyperparameters are changed, influenced by the defined ranges and tuning parameters. The search algorithm declared in the configuration file searches over a generated search space for the best possible parameters.
Export AD Pipeline	Normally, after the tuning is finished, the AD pipeline is exported to the deployment step.

Appendix B

To reach adequate prediction performance and fulfil the requirements defined in Section 3, several models were investigated. All models were tested and evaluated against a reduced test data set, consisting of the time series data of one CPS and a limited number of anomalies. No optimisations were made to the models. The test results can be seen in Table A4. Decisive characteristics for the choice of the model are the number of required parameters, the recognition rate and the time required for the training and the evaluation of the test data. The table clearly shows that shallow methods, despite their fast evaluation, have significant weaknesses in the recognition rate for dynamic and complex time series, as specified in [11]. The LSTMAE has a higher number of parameters compared to the other models. Due to the focused universal application and the limited process and memory resources, this is a major disadvantage. The CAE shows a slightly improved recognition in our test dataset compared to the AE architecture without convolutional layers. Based on the findings of [37–41,43], which identified the good performance of the ConvAE on industrial time series data, we have chosen the 1D-ConvAE as the model.

Table A4. Model Evaluation.

Model	Performance							Size	Avg. Time [ms]	
	TP	FP	TN	FN	Recall	Precision	F1	Complexity	Training	Evaluation
OCSVM	400	25,005	19,633	138	0.7434	0.0157	0.0308	Low	37,041.6	36,811.1
iForest	467	13,709	30,865	71	0.8680	0.0329	0.0634	Low	1874.9	574.1
LSTMAE	8	1	693	1	0.8888	0.8888	0.8888	High	372,029.5	23,491.3
AE	8	2	692	1	0.8888	0.8	0.8421	Medium	13,151.2	18,720
1D-ConvAE	8	0	694	2	0.8	1	0.8888	Medium	113,227.6	20,045.2

OCSVM = One-Class Support Vector Machine [56]; iForest = Isolation Forest [57]; LSTMAE = Long Short-term Memory Autoencoder [58]; AE = Autoencoder [59]; 1D-ConvAE = One dimensional convolutional Autoencoders.

Appendix C

Figure A1 shows additional showcases of anomalous samples. In Figure A1a the collision case is shown. As described in Section 6.2, the system detects typically heavy collisions. The production of light collisions is a difficult task that requires precise interference in the process. Therefore, a pole was prepared with a predetermined breaking point and applied against the rotation direction of the rotary table during regular operation. Figure A1b shows the case defect component. A container was manipulated to force pressure on the moving conveyor belt at a certain point. To create the error case seen in Figure A1c, minor

shocks were manually applied to the plate of the rotation table. A specialist carried out all the different tests, considering safety aspects for the person and the system.

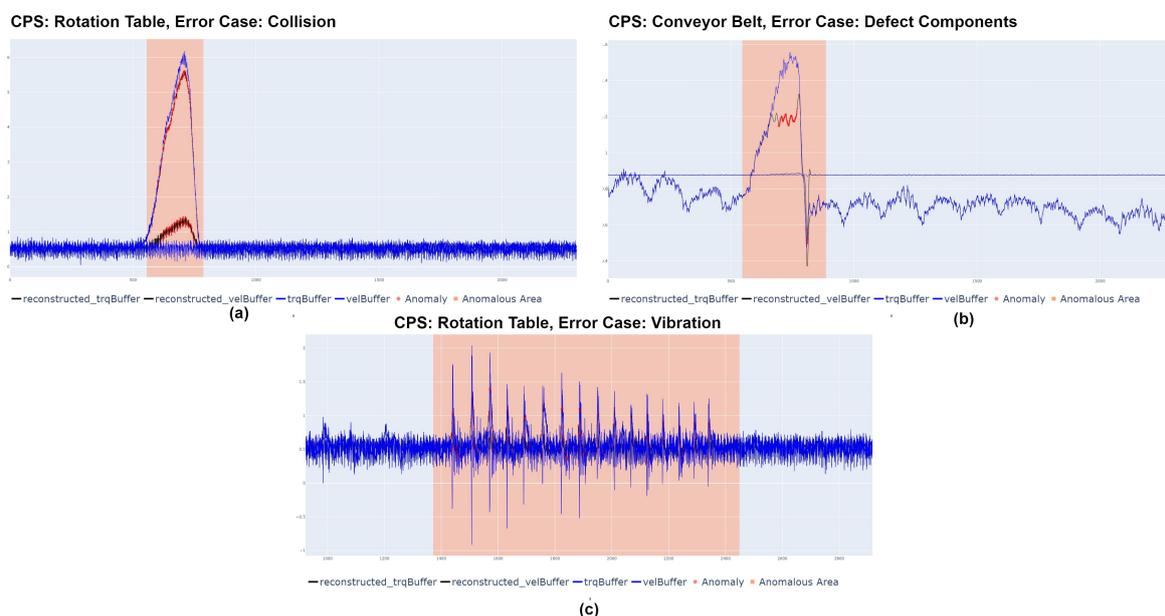


Figure A1. Additional Anomalous Samples.

References

1. Marwedel, P. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*; Springer: Cham, Switzerland, 2021; pp. 1–15.
2. Jazdi, N. Cyber physical systems in the context of Industry 4.0. In Proceedings of the 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 22–24 May 2014; pp. 14–16. [\[CrossRef\]](#)
3. Rajkumar, R.; Lee, I.; Sha, L.; Stankovic, J. Cyber-Physical Systems: The Next Computing Revolution. In Proceedings of the Design Automation Conference, Anaheim, CA, USA, 13–18 June 2010; pp. 731–736. [\[CrossRef\]](#)
4. Müller, T.; Jazdi, N.; Schmidt, J.; Weyrich, M. Cyber-physical production systems: Enhancement with a self-organized reconfiguration management. *Procedia CIRP* **2021**, *9*, 549–554. [\[CrossRef\]](#)
5. Monostori, L. Cyber-physical Production Systems: Roots, Expectations and R & D Challenges. *Procedia CIRP* **2014**, *17*, 9–13. [\[CrossRef\]](#)
6. Ali, N.; Hussain, M.; Hong, J.-E. SafeSoCPS: A Composite Safety Analysis Approach for System of Cyber-Physical Systems. *Sensors* **2022**, *22*, 4474. [\[CrossRef\]](#)
7. Eiteneuer, B.; Hranisavljevic, N.; Niggemann, O. Dimensionality Reduction and Anomaly Detection for CPPS Data using Autoencoder. In Proceedings of the 2019 IEEE International Conference on Industrial Technology (ICIT), Melbourne, VIC, Australia, 13–15 February 2019; pp. 1286–1292. [\[CrossRef\]](#)
8. Adepu, S.; Mathur, A. Distributed Attack Detection in a Water Treatment Plant: Method and Case Study. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 86–99. [\[CrossRef\]](#)
9. Chandola, V.; Banerjee, A.; Kumar, V. Anomaly detection: A survey. *ACM Comput. Surv.* **2009**, *41*, 58. [\[CrossRef\]](#)
10. Stojanovic, L.; Dinic, M.; Stojanovic, N.; Stojadinovic, A. Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 1647–1652. [\[CrossRef\]](#)
11. Ruff, L.; Kauffmann, J.R.; Vandermeulen, R.A.; Montavon, G.; Samek, W.; Kloft, M.; Dietterich, T.G.; Müller, K.-R. A Unifying Review of Deep and Shallow Anomaly Detection. *Proc. IEEE* **2021**, *109*, 756–795. [\[CrossRef\]](#)
12. Gerz, F.; Bastürk, T.R.; Kirchhoff, J.; Denker, J.; Al-Shrouf, L.; Jelali, M. Comparative Study and a New Industrial Platform for Decentralized Anomaly Detection Using Machine Learning Algorithms. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–8. [\[CrossRef\]](#)
13. Bulusu, S.; Kailkhura, B.; Li, B.; Varshney, P.K.; Song, D. Anomalous Example Detection in Deep Learning: A Survey. *IEEE Access* **2020**, *8*, 132330–132347. [\[CrossRef\]](#)
14. Mishra, P.; Varadharajan, V.; Tupakula, U.; Pilli, E.S. A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 686–728. [\[CrossRef\]](#)
15. Thudumu, S.; Branch, P.; Jin, J.; Singh, J. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Big Data* **2020**, *7*, 1–30. [\[CrossRef\]](#)

16. Cook, A.A.; Misirlı, G.; Fan, Z. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet Things J.* **2020**, *7*, 6481–6494. [[CrossRef](#)]
17. Goldstein, M.; Uchida, S. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE* **2016**, *11*, e0152173. [[CrossRef](#)] [[PubMed](#)]
18. Oza, P.; Patel, V.M. One-Class Convolutional Neural Network. *IEEE Signal Process. Lett.* **2019**, *26*, 277–281. [[CrossRef](#)]
19. Erfani, S.M.; Rajasegarar, S.; Karunasekera, S.; Leckie, C. High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning. *Pattern Recognit.* **2016**, *58*, 121–134. [[CrossRef](#)]
20. Smets, K.; Verdonk, B.; Jordaan, E.M. Discovering novelty in spatio/temporal data using one-class support vector machines. In Proceedings of the 2009 International Joint Conference on Neural Networks, Atlanta, GA, USA, 14–19 June 2009; pp. 2956–2963. [[CrossRef](#)]
21. Zong, B.; Song, Q.; Min, M.R.; Cheng, W.; Lumezanu, C.; Cho, D.; Chen, H. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
22. Xiaoyi, G.; Akoglu, L.; Rinaldo, A. Statistical analysis of nearest neighbor methods for anomaly detection. *arXiv* **2019**, arXiv:1907.03813.
23. Elnour, M.; Meskin, N.; Khan, K.; Jain, R. A dual-isolation-forests-based attack detection framework for industrial control systems. *IEEE Access* **2020**, *8*, 36639–36651. [[CrossRef](#)]
24. Pang, G.; Shen, C.; Cao, L.; Van Den Hengel, A. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* **2022**, *54*, 1–38. [[CrossRef](#)]
25. Li, D.; Chen, D.; Jin, B.; Shi, L.; Goh, J.; Ng, S.K. MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks. In *Artificial Neural Networks and Machine Learning—ICANN 2019: Text and Time Series ICANN 2019 Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2019; Volume 11730, pp. 703–716. [[CrossRef](#)]
26. Choi, Y.; Lim, H.; Choi, H.; Kim, I.-J. GAN-Based Anomaly Detection and Localization of Multivariate Time Series Data for Power Plant. In Proceedings of the 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Republic of Korea, 19–22 February 2020; pp. 71–74. [[CrossRef](#)]
27. Jiang, W.; Hong, Y.; Zhou, B.; He, X.; Cheng, C. A GAN-Based Anomaly Detection Approach for Imbalanced Industrial Time Series. *IEEE Access* **2019**, *7*, 143608–143619. [[CrossRef](#)]
28. Heiko, H. Kernel PCA for novelty detection. *Pattern Recognit.* **2007**, *40*, 863–874.
29. Zhao, Y.; Deng, B.; Shen, C.; Liu, Y.; Lu, H.; Hua, X.S. Spatio-temporal autoencoder for video anomaly detection. In Proceedings of the 25th ACM International Conference on Multimedia, Mountain View, CA, USA, 23–27 October 2017.
30. Gong, D.; Liu, L.; Le V.; Saha, B.; Mansour, M.R.; Venkatesh, S.; Hengel, A.V.D. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019.
31. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot-network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [[CrossRef](#)]
32. Park, S.; Adosoglou, G.; Pardalos, P.M. Interpreting rate-distortion of variational autoencoder and using model uncertainty for anomaly detection. *Ann. Math. Artif. Intell.* **2022**, *90*, 735–752. [[CrossRef](#)]
33. Jinwon, A.; Cho, S. Variational autoencoder based anomaly detection using reconstruction probability. *Spec. Lect. IE* **2015**, *2*, 1–18.
34. Munir, M.; Siddiqui, S.A.; Dengel, A.; Ahmed, S. DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series. *IEEE Access* **2019**, *7*, 1991–2005. [[CrossRef](#)]
35. Gong, W.; Chen, H.; Zhang, Z.; Zhang, M.; Gao, H. A Data-Driven-Based Fault Diagnosis Approach for Electrical Power DC-DC Inverter by Using Modified Convolutional Neural Network With Global Average Pooling and 2-D Feature Image. *IEEE Access* **2020**, *8*, 73677–73697. [[CrossRef](#)]
36. Gong, W.; Wang, Y.; Zhang, M.; Mihankhah, E.; Chen, H.; Wang, D. A Fast Anomaly Diagnosis Approach Based on Modified CNN and Multisensor Data Fusion. *IEEE Trans. Ind. Electron.* **2022**, *69*, 13636–13646. [[CrossRef](#)]
37. Qu, C.; Zhou, Z.; Liu, Z.; Jia, S. Predictive anomaly detection for marine diesel engine based on echo state network and autoencoder. *Energy Rep.* **2022**, *8* (Suppl. 4), 998–1003. [[CrossRef](#)]
38. Malviya, V.; Mukherjee, I.; Tallur, S. Edge-Compatible Convolutional Autoencoder Implemented on FPGA for Anomaly Detection in Vibration Condition-Based Monitoring. *IEEE Sens. Lett.* **2022**, *6*, 1–4. [[CrossRef](#)]
39. Guo, X.; Liu, X.; Zhu, E.; Yin, J. Deep Clustering with Convolutional Autoencoders. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, 14–18 November 2017; Proceedings, Part II 24*; Springer: Cham, Switzerland, 2017; p. 10635. [[CrossRef](#)]
40. Lee, G.; Jung, M.; Song, M.; Choo, J. Unsupervised anomaly detection of the gas turbine operation via convolutional auto-encoder. In Proceedings of the 2020 IEEE International Conference on Prognostics and Health Management (ICPHM), Detroit, MI, USA, 8–10 June 2020; pp. 1–6. [[CrossRef](#)]
41. Yu, J.; Zhou, X. One-Dimensional Residual Convolutional Autoencoder Based Feature Learning for Gearbox Fault Diagnosis. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6347–6358. [[CrossRef](#)]
42. Chen, T.; Liu, X.; Xia, B.; Wang, W.; Lai, Y. Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder. *IEEE Access* **2020**, *8*, 47072–47081. [[CrossRef](#)]

43. Kwak, M.; Kim, S.B. Unsupervised Abnormal Sensor Signal Detection With Channelwise Reconstruction Errors. *IEEE Access* **2021**, *9*, 39995–40007. [[CrossRef](#)]
44. Lai, Y.; Liu, Z.; Song, Z.; Wang, Y.; Gao, Y. Anomaly detection in Industrial Autonomous Decentralized System based on time series. *Simul. Model. Pract. Theory* **2016**, *65*, 57–71. [[CrossRef](#)]
45. Sanjith, S.L.; Prakash Raj, E.G.D. Decentralized Time-Window Based Real-Time Anomaly Detection Mechanism (DTRAD) in Iot. *Int. J. Recent Technol. Eng.* **2019**, *8*, 1619–1625. [[CrossRef](#)]
46. Gupta, K.; Sahoo, S.; Mohanty, R.; Panigrahi B.K.; Blaabjerg, F. Decentralized Anomaly Identification in Cyber-Physical DC Microgrids. In Proceedings of the 2022 IEEE Energy Conversion Congress and Exposition (ECCE), Detroit, MI, USA, 9–13 October 2022; pp. 1–6. [[CrossRef](#)]
47. Wilbur, M.; Dubey, A.; Leão B.; Bhattacharjee, S. A Decentralized Approach for Real Time Anomaly Detection in Transportation Networks. In Proceedings of the 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 12–15 June 2019; pp. 274–282. [[CrossRef](#)]
48. Bosman, H.; Iacca, G.; Tejada, A.; Wörtche H.J.; Liotta, A. Spatial anomaly detection in sensor networks using neighborhood information. *Inf. Fusion* **2017**, *33*, 41–56. [[CrossRef](#)]
49. Nikolay, L.; Amizadeh, S.; Flint, I. Generic and scalable framework for automated time-series anomaly detection. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015.
50. Mayra, M.; Wu, C. An unsupervised framework for anomaly detection in a water treatment system. In Proceedings of the 2019 18th IEEE International Conference on Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019.
51. Schneider, P.; Böttinger, K. High-Performance Unsupervised Anomaly Detection for Cyber-Physical System Networks. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy, Toronto, ON, Canada, 15–19 October 2018.
52. Goetz, C.; Humm, G.B. Unsupervised Process Anomaly Detection under Industry Constraints in Cyber-Physical Systems using Convolutional Autoencoder. In *Computational Intelligence for Engineering and Management Applications, Select Proceedings of CIEMA 2022*; Springer: Singapore, 2023, to be published.
53. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. Available online: [tensorflow.org](https://www.tensorflow.org) (accessed on 13 February 2023).
54. Moritz, P.; Nishihara, R.; Wang, S.; Tumanov, A.; Liaw, R.; Liang, E.; Elibol M.; Yang, Z.; Paul, W.; Jordan M.; et al. Ray: A distributed framework for emerging AI applications. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018; pp. 561–577.
55. Zaharia, M.A.; Chen, A.; Davidson, A.; Ghodsi, A.; Hong, S.A.; Konwinski, A.; Murching, S.; Nykodym, T.; Ogilvie, P.; Parkhe, M.; et al. Accelerating the Machine Learning Lifecycle with MLflow. *IEEE Data Eng. Bull.* **2018**, *41*, 39–45.
56. Schölkopf, B.; Williamson, R.; Smola, A.; Shawe-Taylor, J.; Platt, J. Support Vector Method for Novelty Detection. *NIPS* **1999**, *12*, 582–588.
57. Liu, F.; Ting, K.; Zhou, Z. Isolation-Based Anomaly Detection. *ACM Trans. Knowl. Discov. Data* **2012**, *6*, 3. [[CrossRef](#)]
58. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)]
59. Baldi, P. Autoencoders, unsupervised learning and deep architectures. In Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop, Bellevue, DC, USA, 2 July 2011; Volume 27, pp. 37–50.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.