Engineering Applications of Artificial Intelligence 145 (2025) 110152

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Combining informed data-driven anomaly detection with knowledge graphs for root cause analysis in predictive maintenance

Patrick Klein [a], Lukas Malburg [a,b,*], Ralph Bergmann [a,b]

[a] Artificial Intelligence and Intelligent Information Systems, University of Trier, Universitätsring 15, Trier, 54296, Germany
[b] German Research Center for Artificial Intelligence (DFKI), Branch University of Trier, Universitätsring 15, Trier, 54296, Germany

ARTICLE INFO

Keywords:
Hybrid artificial intelligence
Knowledge-based diagnosis
Data-driven anomaly detection
Predictive maintenance
Knowledge graph
Semantic web technologies

ABSTRACT

Industry 4.0 has facilitated the access to sensor and actuator data from manufacturing systems, leading to studies on data-driven anomaly detection, but limited attention has been paid to finding root causes and automating this process using formalized expert knowledge. This is crucial due to the scarcity of qualified engineers and the time-consuming nature of diagnosing issues in large production systems. To address this gap, we present a framework that combines data-driven anomaly detection with a knowledge graph that provides domain knowledge by leveraging typical explanations of such models (i.e.,data streams potentially caused the detection) for further diagnosis. The framework's usefulness to infer affected components or data set labels has been evaluated using two deep anomaly detection approaches. For knowledge-based diagnosis, three query strategies that utilize various knowledge graph relationships are implemented through three Artificial Intelligence (AI) techniques. The proposed anomaly detection approach, informed by integrating expert knowledge via the graph structure of the knowledge graph and node embeddings for encoding time series, outperforms baselines and a deep autoencoder in detecting anomalies and in identifying anomalous data streams. In subsequent diagnosis, it achieves the best performance on a complete knowledge graph in combination with a graph pattern matching query by identifying the label or affected component in 60% of detected anomalies by providing 4.1 labels or 2.3 components until the correct one is identified. In case of a corrupted one, Symbolic-Driven Neural Reasoning (SDNR) and Case-Based Reasoning (CBR) with knowledge graph embeddings demonstrate advantages by halving the number of incorrect labels and unaffected components.

1. Introduction

Industry 4.0 has transformed production facilities into Cyber-Physical Production Systems (CPPSs) equipped with numerous sensors and actuators (Nguyen et al., 2019). This transformation enables a change from preventive to Predictive Maintenance (PredM) through automated signal analysis (Selcuk, 2017), ensuring safety and minimizing unexpected disturbances. A core task of PredM is the detection of anomalies and the determination of their causes (Serradilla et al., 2020a). Two key challenges are of particular importance in this regard: (1) when monitoring entire CPPSs (i.e., at system level Tamssaouet et al., 2021), high-dimensional time series are being generated and (2) usually only a few labeled examples of occurred faults and failures[1] are available (Klein et al., 2020). Due to limited labeled data, unsupervised or self-supervised learning approaches have gained high interest for anomaly detection (Darban et al., 2022). Some approaches proposed for that purpose (e.g., Zhao et al., 2020a; Zhang et al., 2019a; Khoshnevisan and Fan, 2019; Bulla and Birje, 2021; Deng and Hooi, 2021; Su et al., 2019a; Li et al., 2021) identify data streams with the most significant deviations from the normal state for samples classified as abnormal (Darban et al., 2022). However, these approaches typically end with identifying abnormal data streams, making it challenging for engineers to deduce affected components due to the high number of monitored data streams and the complexity of the component interrelationships of a CPPS (Monostori, 2014; Diedrich, 2023; Medina-Oliva et al., 2014). Finding the cause is often not considered as part of these models and studies, as the domain knowledge required for inference is difficult to obtain for such a model from unlabeled time series data alone.

For this reason, research explores combining PredM approaches, traditionally divided into knowledge-driven, physics-based, and data-driven, to handle the complexity of modern manufacturing assets

* Corresponding author.
E-mail addresses: patrick.pat.klein@gmail.com (P. Klein), malburgl@uni-trier.de (L. Malburg), bergmann@uni-trier.de (R. Bergmann).

[1] For better readability, we do not distinguish between both in the following anymore and use failures for both.

https://doi.org/10.1016/j.engappai.2025.110152
Received 12 January 2024; Received in revised form 13 November 2024; Accepted 22 January 2025
Available online 8 February 2025

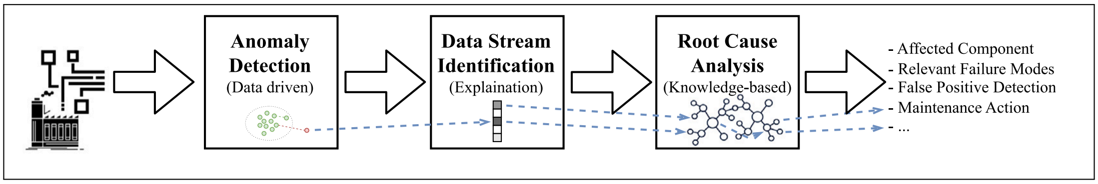

Fig. 1. Sketch of the applied approach that processes incoming data with a data-driven anomaly detection, then identifies anomalous data streams to infer affected components or the data set's label by using a knowledge graph as an RCA.

(Jimenez et al., 2020; Nunes et al., 2023). These approaches, though different, can be complementary when properly deployed (Jimenez et al., 2020; Bouhadra and Forest, 2024). Although data-driven anomaly detection leverages deep learning to process high-dimensional, high-frequency, and high-volume time series data and excels in pattern matching (Rezaeianjouybari and Shang, 2020), a knowledge-based model can incorporate available domain knowledge (von Rueden et al., 2019). Prior knowledge about the system is particularly helpful in finding the root cause in complex and interconnected systems (Niggemann and Lohweg, 2015) and overcoming some challenges (e.g., spurious correlations and number of training examples) in data-driven models alone (Steenwinckel et al., 2021b; Klein et al., 2021; Farbiz et al., 2022; Hagendorfer, 2021). Despite the potential of knowledge-based models to complement data-driven ones for diagnostic purposes (Jimenez et al., 2020; Bouhadra and Forest, 2024), their integration is still in its infancy (De Paepe et al., 2021; Nunes et al., 2023; Breit et al., 2023; Franciosi et al., 2024). To address these aforementioned shortcomings, we propose a framework that extends data-driven anomaly detection with a knowledge-based diagnosis Root Cause Analysis (RCA) module. By this combined method, it is possible to infer the data set's label or affected component of a CPPS, as shown in Fig. 1. The main contributions are as follows:

- Since deep anomaly detection methods for PredM usually end with a case study of identified data streams as an explanation (e.g., Zhao et al., 2020a; Zhang et al., 2019a; Khoshnevisan and Fan, 2019; Bulla and Birje, 2021; Deng and Hooi, 2021; Su et al., 2019a; Li et al., 2021), one main contribution is to continue at this point with a RCA. Therefore, relationships between data streams and components formalized in a knowledge graph are leveraged to infer or reason the data set's label or affected component in a CPPS. We propose several query strategies and AI approaches for logical inference (i.e., SPARQL) or using knowledge graph embeddings with SDNR and CBR.
- We propose an informed self-supervised one-class learning approach that integrates domain knowledge in the form of data stream relationships derived from the knowledge graph. It outperforms several baselines and a deep autoencoder. Combined with a counterfactual approach, it allows identifying data streams for detected anomalies.
- We present a FMEA ontology to model the knowledge about faults and failures, instantiated based on the data set used and aligned with an existing smart factory model ontology.

This paper follows the Design Science Research Methodology (Peffers et al., 2018) and an experimental evaluation (Von Alan et al., 2004, p. 85) to rigorously prove the usefulness, quality, and effectiveness of the developed artifact. A literature review ensures research rigor, and a concept is defined to combine deep anomaly detection models with semantically modeled expert knowledge, implements it prototypically, and evaluates it experimentally using data split into train, validation, and test sets. The anomaly detection is evaluated for its ability to detect faults and failures and identify the causative data streams, as well as for diagnosis. In addition, the usefulness of a knowledge graph for logical inference, SDNR, and CBR is evaluated.

The remainder of the paper is structured as follows: First, some basic definitions, techniques, the used data set and knowledge graph are presented (Section 2). Followed by the proposed framework for combining a data-driven anomaly detection with a knowledge graph for RCA and its specific implementation for evaluation is described (Section 3). Then the results of the evaluation are presented and discussed (Section 4). Next, related work on data-driven anomaly detection with expert knowledge for RCA in the field of PredM is presented (Section 5), before, the contributions and limitations are discussed (Section 6). Finally, a conclusion and possible future research are given (Section 7).

2. Foundations

First, the foundations of anomaly detection are introduced (Section 2.1), focusing on self-supervised approaches used in the proposed method (Section 2.2). Afterwards, counterfactuals for explaining detected anomalies are presented in Section 2.3. Next, graph structure learning (Section 2.4) to integrate expert knowledge into the proposed model is introduced. Thereafter, the used data set and the knowledge graph for finding the data set's label or affected component are described in Sections 2.5 and 2.6. Finally, SDNR for identifying root causes of detected anomalies is explained in Section 2.7.

2.1. Data-driven anomaly detection

The general objective of anomaly detection is to locate patterns in the data that do not align with the expected behavior observed in a data set regarded as describing the usual state (Foorthuis, 2020; Ruff et al., 2020). Formally, an anomaly detection model $f(\cdot)$ assigns examples $x \in \mathcal{X} = \{x^e\}_{e=0}^{N}$ (e.g., windows of time series) to either 0 for indicating that the input is in a healthy state or 1 for an irregularity, expressed as $f : \mathcal{X} \rightarrow \{0, 1\}$. A common approach for unsupervised anomaly detection is the utilization of a one-class classifier for f (Fourure et al., 2021) to obtain an anomaly score $s = f(x)$ in a first step, followed by a final binary classification as anomaly $\hat{y} = 1$ if $s > \tau$ or normal with $\hat{y} = 0$:

$$\hat{y} = \begin{cases} 1, & \text{if } s(x) > \tau \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Depending on the use case, the anomaly score $s(x)$ may be a prediction for the most recent time point of the window x (e.g., Kim et al., 2021) or for the whole window (e.g., Qiu et al., 2021). Typically, f is trained on an anomaly-free data set $\mathcal{X}$, consisting of only examples that represent normal conditions, which are not labeled (Fourure et al., 2021). For empirically determining the threshold τ, only a few failure examples are needed, which is less than what is required to learn a supervised model. A larger anomaly score $s(x)$ implies that the example x is more likely to be anomalous (Fourure et al., 2021) and in PredM, a higher score $s(x)$ may indicate a more severe fault or a condition closer to the end of life (Malhotra et al., 2016).

Common baseline approaches for f include k-Nearest Neighbors (k-NN) and One-Class Support Vector Machine (OC-SVM). k-NN is a distance-based anomaly detection method that relies on the assumption that normal data points are closely grouped and anomalous ones are separated from them. These techniques typically calculate the anomaly score s as the distance (e.g., Euclidean distance) between a sample q and its k-nearest neighbors of the (fault-free) data set, or from one or more mean points. Representation-based methods learn a transformation that projects data into a space that facilitates anomaly recognition (Fourure et al., 2021). A prominent example is OC-SVM (Schölkopf et al., 1999), which uses a hypersphere to encompass all normal instances in the projection space (Fourure et al., 2021). Ruff et al. (2018) proposed a deep one-class classification approach, which learns representations by minimizing the distance of normal examples to a center η as given in (Ruff et al., 2018; Sohn et al., 2021):

$$\min_{f} \mathbb{E}_x \| f(x) - \eta \|^2. \tag{2}$$

The neural network $f(\cdot)$ extracts common features of normal examples to map them close to the center η (Ruff et al., 2018). Another approach of this category by Sohn et al. (2021) obtains useful representations through self-supervised deep learning as input to OC-SVM or for one-class classification.

2.2. Self-supervised representation learning for anomaly detection

Self-supervised learning approaches for computer vision have gained popularity due to their ability to achieve performance comparable to supervised learning with less labeled data (Jaiswal et al., 2020). These approaches embed augmented versions of the same sample image as input and embed them close together, while pushing away embeddings from different samples (Jaiswal et al., 2020). Image representations are often based on a Siamese neural network architecture (Chen and He, 2020) with geometric and color transformations as augmentations. For sequential data such as time series, the task often involves predicting future or missing information (Jaiswal et al., 2020). However, Sohn et al. (2021) argue that self-supervised learning for multi-class classification may not suit one-class anomaly detection due to representational issues: First, most contrastive learning approaches require negative pairs, which can counteract a one-class classification by pushing away the same class instead of centering it (cf. Eq. (2)) (Sohn et al., 2021). Furthermore, the loss function of self-supervised learning aims for a uniform distribution of representations, making outlier isolation difficult (Sohn et al., 2021). This means that the representations learned with those approaches are not suitable for anomaly detection. For using anomaly detection for time series, useful augmentations are less known, and it is challenging to design these transformations manually (Qiu et al., 2021). These shortcomings lead to the proposed modification of the self-supervised approach introduced in the following to be applied for detecting anomalies.

SimSiam siamese network

The self-supervised SIMple SIAMese Network (SimSiam) approach by Chen and He (2020) achieves noteworthy results without requiring negative sample pairs, large batch sizes, or momentum encoders, and serves as the foundation for the proposed one-class anomaly detection method discussed later in this paper. Fig. 2 shows the architecture in conjunction with a traditional Siamese neural network. Two augmented images (x_a, x_b) are created from the same image x and then processed by an encoder network $f(\cdot)$ with the same weights θ, resulting in embedding $e = f(x)$. For visual tasks, f typically consists of an encoder to extract deep features (e.g., ResNet He et al., 2015) plus a predictor Multilayer Perceptron (MLP) $h(\cdot)$ (Grill et al., 2020). The MLP predictor has a bottleneck structure (comparable to an autoencoder), and its role requires further investigation (Chen and He, 2020). The output p of the prediction MLP head $h(\cdot)$ predicts the other encoder's output: $p_a = h(f(x_a))$ and $e_b = f(x_b)$. The total loss for each pair is defined as:

$$L = \frac{1}{2} d(p_a, e_b) + \frac{1}{2} d(p_b, e_a), \text{ with } d(p_a, e_b) = -\frac{p_a}{\|p_a\|_2} \times \frac{e_b}{\|e_b\|_2} \tag{3}$$

where $\|\cdot\|_2$ is the ℓ_2-norm, and $d(\cdot)$ is the negative Cosine similarity between the elements. The stop gradient operation[2] is applied to e_b

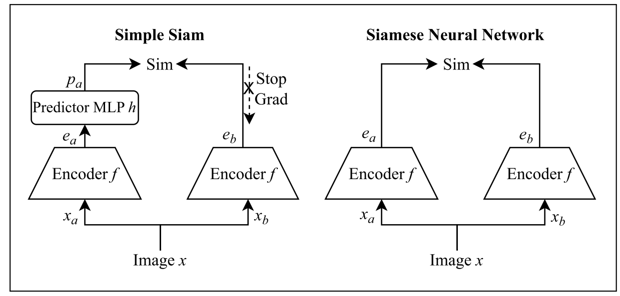

Fig. 2. Sketch of the SimSiam architecture from Chen and He (2020) compared to the common siamese neural network architecture (Bromley et al., 1993).

so that the network is only updated based on the gradients of the other branch, i.e., the error of predictor p_a. The loss is averaged over a batch, with a minimum value of −1. Experiments have shown competitive performance in image classification using representation e from encoder f with a linear classifier or with k-NN (Chen and He, 2020). The authors attribute the prevention of a collapsing solution to the prediction head and gradient stop operation (Chen and He, 2020).

2.3. Counterfactual explanation

Data-driven methods for detecting anomalies in time series data often lack helpful explanations for their predictions (Sulem et al., 2022; Karlsson et al., 2020). The reconstruction error of autoencoders can identify anomalous data streams (Zhang et al., 2019a; Zhao et al., 2020b) (see Fig. 14) or they can be combined with feature-based explanation methods such as SHapley Additive exPlanations (Lundberg and Lee, 2017), as shown by Antwarg et al. (2019) or Bulla and Birje (2021). Another approach is the use of counterfactuals, which highlights what should have been changed in an instance to achieve a different outcome (Guidotti, 2022). For distance-based classification of time series, Delaney et al. (2021, 2020) propose to retrieve the nearest unlike neighbor (NUN) x_{NUN} from another class $y_j \neq y_i$, which is the most similar one to the query q, and use it to create a counterfactual explanation:

$$x_{cf} = x_{NUN} + \delta, \text{ with } x_{NUN} = \min_{x^j} d(q, x^j) \mid x \notin y_i, f(q) \in y_i. \tag{4}$$

The Nearest Unlike Neighbor (NUN) x_{NUN} is modified to be closer to the query q according to a distance measure $d(\cdot,\cdot)$. The perturbation δ ends before the predicted class of $x_{NUN}+\delta$ changes to y_i (Delaney et al., 2020). To explain a supervised deep learning model, Delaney et al. (2021) suggest using a feature weight w (e.g., from class activation mapping (CAM)) (Zhou et al., 2016) to assign weights to the time steps of x_{NUN} based on their contribution to the model's prediction towards y_j. The contiguous sub-sequence of x_{NUN} according to w is then replaced with the values of q until $f(x_{cf}) \in y_j$ (Delaney et al., 2021).

2.4. Graph structure learning

In addressing the challenge of modeling complex relationships between data streams, graph-based approaches offer a promising direction. Graph Structure Learning (GSL) (Zhu et al., 2021) focuses on developing modules to learn an encoding function $f_{GSL}(\cdot)$ that produces the optimal graph structure $A^* \in \mathbb{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ as an adjacency matrix with $\mathcal{N}$ nodes (Zhu et al., 2021). The following variants for a graph structure learning module have been found in the literature:

$$\mathrm{A}^L_W = \mathrm{ReLU}(W) \tag{5}$$

$$\mathrm{A}^L_{E_{11}} = \mathrm{ReLU}(\tanh(\alpha(\mathrm{E}_1 \cdot \mathrm{E}_1^T))) \tag{6}$$

[2] Detailed explanation at https://www.tensorflow.org/api_docs/python/tf/stop_gradient, accessed on 07/24/2022.

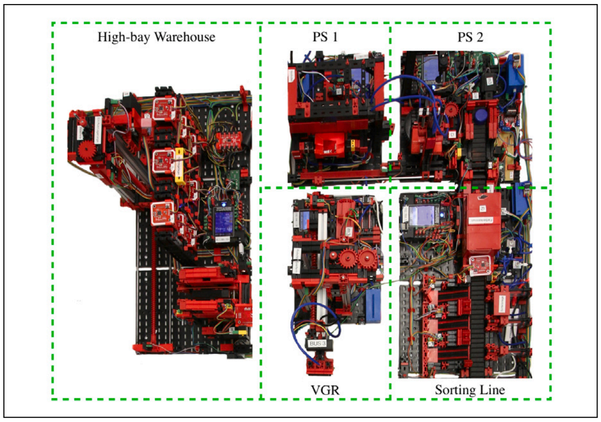

Fig. 3. Smart factory model used for data generation (Klein. and Bergmann., 2019). PS means Processing Station, VGR is Vacuum Gripper Robot.

$$\mathrm{A}^L_{E_{12}} = \mathrm{ReLU}(\tanh(\alpha(\mathrm{E}_1 \cdot \mathrm{E}_2^T))) \tag{7}$$

$$\mathrm{A}^L_{E_{12}-E_{21}} = \mathrm{ReLU}(\tanh(\alpha(\mathrm{E}_1 \cdot \mathrm{E}_2^T - \mathrm{E}_2 \cdot \mathrm{E}_1^T))) \tag{8}$$

$$\mathrm{A}^L_{\|E_{11}\|_2} = \mathrm{ReLU}(\|E_1\|_2 \cdot \|E_1\|_2^T) \tag{9}$$

$$\mathrm{A}^L_{\|E_{12}\|_2} = \mathrm{ReLU}(\|E_1\|_2 \cdot \|E_2\|_2^T) \tag{10}$$

Eq. (5), based on Zhu et al. (2021) and referred to as Global-A in Wu et al. (2020), is a baseline using a learnable weight matrix $W \in \mathrm{R}^{|\mathcal{N}|\times|\mathcal{N}|}$ that is randomly initialized or a predefined adjacency matrix can be used. The Rectified Linear Unit (ReLU) activation function ensures positive values in the adjacency matrix. Fatemi et al. (2021) applied the Exponential Linear Unit (ELU) function (Clevert et al., 2016) instead of ReLU and added 1 to address the gradient flow problem caused by edges in A^L that become zero or less. Eqs. (6)–(10) enable the learning of a node representation E_i for each node n_i, where $E_i = \tanh(\alpha M_i \cdot W_i + b_i)$ with learned parameters $M_i \in \mathrm{R}^{|\mathcal{N}|\times d}$ and $W_i \in \mathrm{R}^{d\times d}$ Here, d is the number of dimensions, and α is a hyperparameter for controlling the saturation rate of $\tanh(\cdot)$. Eqs. (6)–(8) are from Wu et al. (2020), while Eqs. (9) and (10) are from Deng and Hooi (2021). For an undirected graph, the outgoing and ingoing edge representations are identical (Eqs. (6) and (9)), while a directed one uses different representations (e.g., in Eq. (7), (8) and (10)). The learned adjacency matrix A^L is commonly post-processed to achieve A^* with the desired structure and properties, such as sparsity and normalization.

2.5. Data set

The data set[3] used in this work has been generated using a Fischertechnik smart factory model enhanced with various condition monitoring sensors, including accelerometers for measuring vibrations and differential air pressure sensors for pneumatic systems (Klein. and Bergmann., 2019). It comprises five workstations (referred to as txt15 to txt19): a sorting line with color recognition (txt15), two Processing Stations (PSs, txt16 and txt17), a Vacuum Gripper Robot (VGR, txt18), and a high-bay warehouse (txt19), as shown in Fig. 3. These workstations are interconnected to handle incoming workpieces and simulate a production process. During the simulated manufacturing process, faults and failures are randomly injected and labeled with 28 classes, categorized as follows: (i) false signals from sensors used for control-purposes, (ii) artificially induced vibrations on two conveyor belt motors, (iii) leakages in pneumatic systems and (iv) various other malfunctions (e.g., incorrectly executed transport processes). All sensor and actuator data streams are recorded to capture the data that describe the entire state of the CPPS model for monitoring at the system level. For each failure mode, expert knowledge about features (i.e., data streams) is specified that contain directly observable patterns and relevant contextual information to detect the failure (see Table 11). The data is pre-processed, which involves interpolating missing values, re-sampling to 4 ms frequency, and scaling to $[0, 1]$. A sliding window with a step size of one second and a length of four seconds is used to extract individual examples, resulting in a multivariate time series of $X = [0, 1]^{1000\times 61}$ (i.e., 1000 observations with 61 data streams). The examples are divided into training, validation, and test sets, ensuring independence by assigning examples from a failure simulation run to either training/validation or test set. The class and example distribution of the data set are shown in the Appendix in Table 9.

2.6. Knowledge graph

To represent expert knowledge about a manufacturing system such as a CPPS, a common approach is to build a so-called knowledge graph by using Semantic Web Technologys (SWTs) such as the Resource Description Framework (RDF) or the Web Ontology Language (OWL), e.g., Kalayci et al. (2020) and Hubauer et al. (2018). Using OWL, knowledge about things (e.g., components, failure modes, data streams) of a production environment can be represented by forming groups and relationships between them (Hitzler et al., 2012). A knowledge graph provides information on the relationships between its entities (Ji et al., 2021), with relationships between actuators and sensors (e.g., *hosts* of the Semantic Sensor Network ontology Haller et al., 2019) being of particular interest for tracing anomalous signals detected by anomaly detection back to their origin. The built knowledge graph makes expert knowledge explicit and machine-readable regarding component arrangements and relationships, as well as failure modes, which is necessary for effective Root Cause Analysis (RCA).

For the smart factory model used to generate the data set (cf. Section 2.5), the semantic knowledge graph FTOnto (Klein et al., 2019a) provides the foundation as an ontology to model the arrangements and relationships of components based on established domain ontologies. Having this knowledge is crucial for building intelligent applications such as for smart production control (Malburg et al., 2023b,a) with semantic web services (Malburg et al., 2020), or for integrating it in neural networks for fault detection (Klein et al., 2021).

2.7. Symbolic inference and symbolic-driven neural reasoning

For diagnosing detected anomalies, maintenance engineers can consult a knowledge graph (cf. Section 2.6) for RCA information, often through queries formulated in the SPARQL. These queries can infer new information by pattern matching, such as determining potential affected components from anomalous data streams. However, SPARQL queries rely on complete and correct Knowledge Graphs, which is often not the case in practice (Boschin et al., 2022). To address missing relationships, symbolic-driven neural reasoning (SDNR) (Zhang et al., 2021) can predict them using knowledge graph embeddings. These embeddings map entities and relations of a knowledge graph by mapping them into low-dimensional vectors, capturing semantic meanings (Ji et al., 2021). This work uses the multipurpose neural embedding model StarSpace (Wu et al., 2018) to embed entities. Therefore, the knowledge graph is represented in the form of triples (h, r, t) comprising a head concept h, a relation r, and a tail concept t. StarSpace creates two training examples from each triple: inferring the left entity (h) from the relation (r) and the right feature vector (t), so that $a = h$ and $b = r + t$. The similarity $sim(a, b)$ should be higher than when a is replaced

[3] https://drive.google.com/drive/folders/1-1KfT_FjsUSTDgXajhl7-ZCPGsL7rYqw?usp=sharing, last accessed on 05/02/2025.

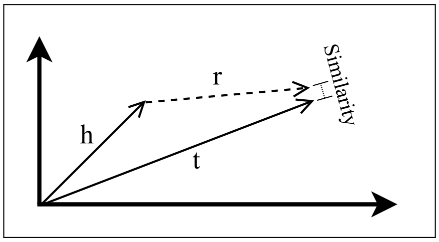

Fig. 4. A translation-based approach (Bordes et al., 2013) such as StarSpace uses the relation vector r as a translation from the head vector h to the entity vector t. Fig. based on Ji et al. (2021).

with a random entity a^-, by a margin of h. During training, StarSpace produces feature vectors yielding $h \approx r + t$ and $h + r \approx t$, as shown in Fig. 4. These embeddings enable SDNR to evaluate logical formulas representing expert knowledge for finding a root cause, and the embeddings can be used to infer unknown (i.e., not modeled) relationships between entities of the knowledge graph. In SDNR, a triple (h, r, t) is interpreted as a binary predicate $r(h, t)$ (Zhang et al., 2021), with its score $s(h, r, t) \in [0, 1]$ from the embedding model like StarSpace, considered as a prediction of its truth value (1 being absolute truth, 0 completely false). More complex logical formulas can be constructed by combining several triples (i.e., binary predicates) with logical operators such as conjunction $\wedge$ (i.e., and), disjunction $\vee$ (i.e., or), or negation $\neg$. By utilizing the (product) t-norm fuzzy logics (Hajek, 1998), a complex formula consisting of atoms f_i (i.e., triples or formulas) can be evaluated using the truth values (i.e., scores $s(\cdot)$) of its constituents, as shown in Eqs. (11a), (11b), and (11c) (Guo et al., 2016). A higher truth value indicates better-satisfied rules (Guo et al., 2016).

$$s\left(f_1 \wedge f_2\right) = s\left(f_1\right) \cdot s\left(f_2\right) \tag{11a}$$

$$s\left(f_1 \vee f_2\right) = s\left(f_1\right) + s\left(f_2\right) - s\left(f_1\right) \cdot s\left(f_2\right) \tag{11b}$$

$$s\left(\neg f_1\right) = 1 - s\left(f_1\right). \tag{11c}$$

3. Combining data-driven anomaly detection with knowledge graphs for diagnosis

After Section 3.1 presented the general framework to combine a data-driven anomaly detection with a knowledge graph, this section describes its application to the previously built knowledge graph (Section 2.6) and generated data set (Section 2.5). First, Section 3.2 describes the data-driven anomaly detection used for module (I), which already integrates expert knowledge. Then, Section 3.3 explains the counterfactual approach used to identify anomalous data streams, which represents the bridge module (II). Next, Section 3.4 presents the RCA module (III). This section compromises how expert knowledge about failures is modeled using semantic web technologies (Section 3.4.1) and how the applied query strategies and constraints are used on the knowledge graph to improve and automate the diagnosis (Section 3.4.2).

3.1. Architectural overview

This section presents a framework for combining data-driven anomaly detection with expert knowledge for diagnosis, represented by a knowledge graph. The technologies selected for anomaly detection and diagnosis reflect the different characteristics of these tasks to optimally fit their respective strengths and weaknesses. Neural networks, for example, possess strong learning capabilities for processing time series data (Rezaeianjouybari and Shang, 2020), while expert knowledge helps to trace the causes of anomalies. This follows a common approach in AI: applying deep learning for low-level perception and logical knowledge for high-level reasoning (Yang et al., 2020). Rather than building a neural network to directly predict root causes, the framework leverages outputs, particularly anomaly scores per data stream, to identify root causes based on prior knowledge and logical constraints. This separation facilitates the learning task of the neural network and allows independent modification of expert knowledge (Yang et al., 2020). The proposed framework, depicted in Fig. 5, consists of three main parts:

Anomaly Detection Module (I): a *data-driven anomaly detection* (left side in Fig. 5),

Bridge Module (II): a *bridge connecting* both modules (middle in Fig. 5), and

RCA Module (III): a *knowledge graph for diagnosis* (right side in Fig. 5).

The framework operates as follows: The data-driven model detects anomalies, as it can learn knowledge from data without any prior knowledge of the system's normal state, typically providing an anomaly score $s \in [0, 1]$ to indicate if an input sample is anomalous (cf. Section 2.1). These models often provide additional information such as affected time intervals and data streams (Darban et al., 2022), which serves as a bridge to access the knowledge graph for diagnosis. The knowledge graph provides information on possible failure modes related to the data stream, using anomalous data streams as entry points. This background knowledge, challenging to learn from time series data alone (Steenwinckel et al., 2021a; Klein et al., 2021), can be used with knowledge graph embedding methods to predict new relationships and uncover unknown knowledge (Garofalo et al., 2018). Logical constraints based on actuator activity and symptoms can refine diagnosis by excluding unreliable predictions. Auxiliary information enriches and verifies predictions based on the knowledge graph's modeled knowledge. Since anomaly diagnosis (i.e., RCA) is knowledge-intensive and time-consuming (Steenwinckel et al., 2018; Diedrich, 2023), quick response is essential to avoid serious failures and unnecessary shutdowns. Thus, accurate anomaly detection and automated RCA are crucial for fast solution identification. This framework can deploy any anomaly detection method ranking data streams by abnormality and any knowledge base providing appropriate entry points. The next section describes the approaches used for each component to assess the framework's usefulness.

3.2. AD module: Anomaly detection with identification of anomalous data streams

This section describes the component associated with the anomaly detection module (I) (cf. Section 3.1). Section 3.2.1 describes the applied Siamese neural network for anomaly detection, and Section 3.2.2 explains the integration of expert knowledge into the neural network via a graph structure learning component and knowledge graph embeddings.

3.2.1. Training a simple siamese network with only normal data

The proposed anomaly detection approach is motivated by the effectiveness of representation learning and distance-based methods (cf. Section 5.1), and their ability to provide (factual) explanations through nearest neighbors. Existing approaches for representation learning are not designed for time series and the specific transformations are unknown (cf. Section 2.2). Approaches that address this (e.g., Qiu et al., 2021) do not provide the affected data streams, which is necessary for diagnosis. To address these shortcomings, the SimSiam architecture (cf. Section 2.2) is chosen for its ability to learn representations without negative examples, its simplicity, and its state-of-the-art performance in classification tasks. It is applied to anomaly detection similarly to its original principle by minimizing the distance between pairs consisting of two differently augmented images x_a, x_b generated from the same underlying image x. As universally useful transformations for generating x_a and x_b are not known for multivariate time series data in anomaly detection (Qiu et al., 2021), the obstacle is to find another way, the model can learn to map both transformations x_a, x_b to the same point in the space, as illustrated in Fig. 6. To accomplish this task, pairs of examples representing the normal condition with different operating states are randomly used as x_a, x_b. The goal is to allow the model to learn a useful transformation for the encoder network $f(\cdot)$ and to map the normal examples near each other (i.e., in high-density regions where i.e., $d(f(x_i), f(x_j)) \approx 0, \exists x \in \mathcal{X}$). Similarly to reconstruction-based anomaly detection approaches, the assumption is that an anomalous instance, which has not been seen before during training, will be mapped apart from any normal example (i.e., in a low-density region). This is because the learned transformation does not map the unseen anomalous data near to the normal data, resulting in an embedding vector that lies further apart than normal examples in the embedding space. A further difference is the removal of the stop gradient operation, which contributes to achieving a standard distribution for each latent feature over a mini batch (Chen and He, 2020). This removal ensures that all normal examples are clustered closely, contrasting with the original architecture's purpose of learning diverse representations for multiple classes (cf. Section 2.2).

In summary, the proposed data-driven anomaly detection approach assumes that the model can learn a representation that eliminates intra-class variations between normal instances of the normal class (e.g., due to different operating states or production processes). This principle is similar to using the reconstruction error in autoencoders. Each time series window x is represented by a latent vector e extracted from an unsupervised deep network $f(\cdot)$, with the anomaly score based on the distance $d(\cdot)$ to known normal examples. Larger distances indicate greater deviations from the normal state (Malhotra et al., 2016) and examples beyond a certain threshold are classified as anomalous.

3.2.2. Integrating expert knowledge about data streams

The knowledge graph from Section 2.6 describes the manufacturing system and contains valuable domain knowledge on the relationships between components. Integrating this kind of prior knowledge aims to improve the anomaly detection model's performance and prevent relearning with the risk of learning it wrong. This leads to a so-called informed machine learning model (von Rueden et al., 2019). Integration of expert knowledge is achieved through an adjacency matrix derived from the knowledge graph, providing a foundation for learning explicit dependencies between data streams using graph convolutions. The derived adjacency matrix contains binary values and is bidirectional due to challenges in determining exact information flow direction and weight of the relationship. To enhance this initial representation, a graph structure learning module (cf. Section 2.4) is employed to refine this adjacency matrix, learning meaningful and causal dependencies between the data streams while considering the existing relationships from the derived one. The learned adjacency matrix A^L is post-processed by applying an element-wise multiplication ($\odot$) with the derived (binary) adjacency matrix A^E, to achieve the optimal one:

$$A^{\star} = A^E \odot A^L. \tag{12}$$

This operation masks out irrelevant relationships that are not modeled in the knowledge graph. The adjacency matrix A^E derived from the knowledge graph (cf. B) ensure that only meaningful/causal dependencies are learned by the graph structure learning module and used to process the data by the anomaly detection model. Utilizing A^E should enable detection of anomalous events that disrupt learned correlations between data streams within the knowledge graph's boundaries. Related work has demonstrated the importance of this approach for processing multivariate time series, particularly for anomaly detection in cyber–physical systems (e.g., Deng and Hooi, 2021).

Furthermore, the embeddings learned from the knowledge graph applying Owl2Vec (Chen et al., 2020) are used as an additional input to provide latent features to describe data streams, as proposed in Klein et al. (2021). These embeddings enable the anomaly detection model to incorporate expert knowledge from the knowledge graph not present in the time series data. Technically, in a deeper layer of the model, the characteristics of the encoded data stream features are concatenated with the latent features of the node that represents the data stream in the knowledge graph, as described in Klein et al. (2021) and in H.2. This integration allows the model to make use of the knowledge in the form of the latent features to encode each data stream's time series along with its knowledge graph features. For example, two data streams with binary values from a light barrier or position switch may have different meanings. By combining data streams with knowledge graph embeddings during encoding, latent features add semantics to the data stream values, aiming to enhance the model's representation.

3.3. Bridge module: Finding the subset of anomalous data streams via a counterfactual approach

An instance-based counterfactual approach is proposed to identify the subset of anomalous data streams by using the example of the nearest normal condition according to the anomaly detection model. Combined with the trained Siamese neural network (Section 3.2), the approach works as follows:

1. For an example $Q^i \in \mathbb{R}^{n \times m}$ detected as anomalous, retrieve the nearest neighbor $X^i_{cf} \in \mathbb{R}^{n \times m}$ from normal examples based on the distance-based anomaly detection (cf. Section 3.2).

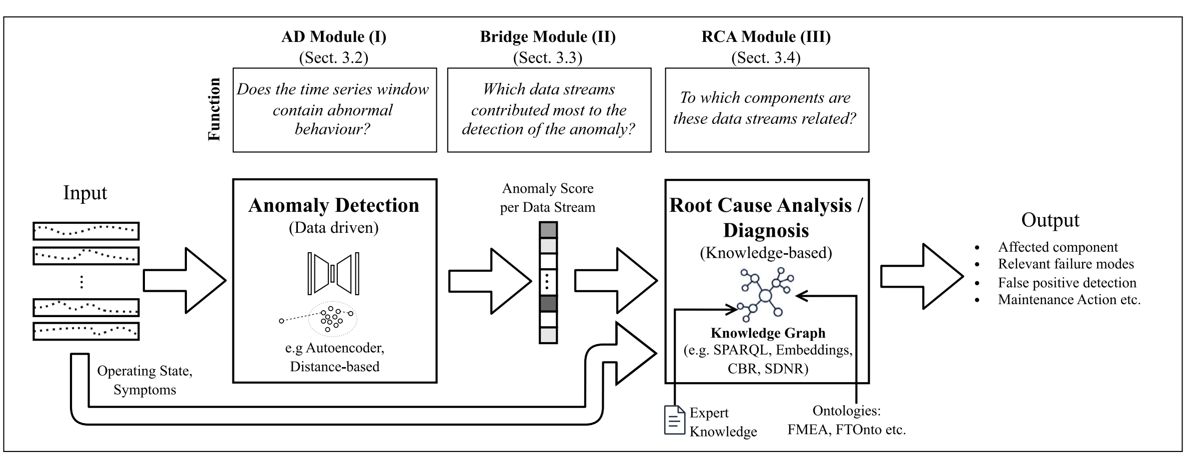

Fig. 5. Sketch of the proposed combination between a data-driven Anomaly Detection and a knowledge-based semantic Knowledge Graph for root-cause analysis.

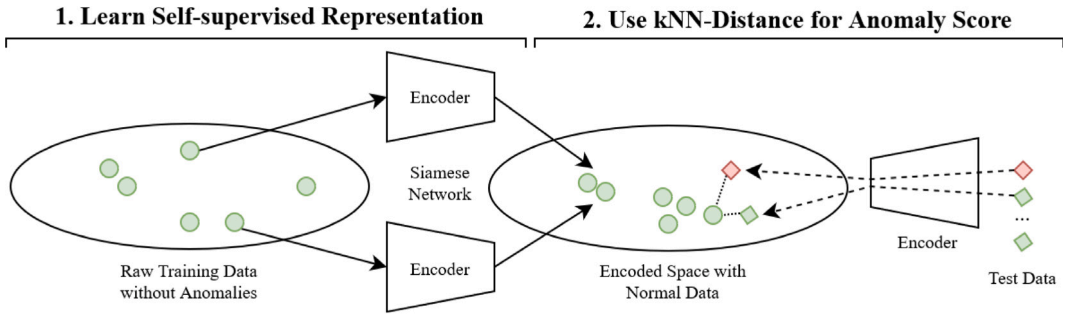

Fig. 6. The SimSiam architecture (cf. Section 2.2) learns a representation using pairs of randomly chosen normal data points (illustrated as green dots in the left ellipse). The encoder is assumed to transform normal data into a Euclidean vector space, mapping them into high-density clusters. For anomaly detection, the Cosine distance between the $k = 1$ nearest neighbors (k-NN) of normal data determines whether an unseen data point is normal or anomalous.

2. Replace each data stream of Q^i with those from X^i_{cf}, creating m versions ($Q^i_{\delta_j}$) where only one data stream j is replaced.
3. For each modification, compare the similarity $s_j = sim\left(f(Q^i_{\delta_j}), f(X^i_{cf})\right)$ in the deep embedding space to the original similarity s (i.e., without the perturbation applied to the input). An improvement ($s_j > s$) indicates that replacing the data stream makes the example more normal, thus j is considered anomalous.
4. Generate a new (artificial) query example by replacing all suspicious data streams with those of the normal example. Retrieve a new nearest neighbor and repeat steps 2–3. The subset of data streams C now marked as suspected is considered causative of the detected anomaly, with order defined by similarity $sim(\cdot)$.

Theoretically, replacing multiple data streams in step two of generating Q^i_δ is possible but increases the computational effort. The number of combinations is calculated as $\binom{m}{k} = \frac{m!}{(m-k)!\cdot k!}$, where m is the total number of data streams and k the number selected in a combination. Input perturbation (as in the second step) allows assessment of the model's reaction to local changes, serving as an explanation (Utkin et al., 2019). The higher output variation indicates a higher dependence on the input feature (i.e., data stream).

3.4. RCA module: Knowledge graph for diagnosis

This section describes the content of the RCA module which consists of the developed FMEA ontology and modeled fault and failure knowledge (Section 3.4.1) and how the knowledge graph is used for RCA (Section 3.4.2).

3.4.1. Developed FMEA ontology and modeled fault and failure knowledge

To facilitate subsequent diagnosis after an anomaly is detected, the knowledge graph from Section 2.6 is extended with knowledge about faults and failures that can occur in the used physical smart factory and are contained in the data set. Before an actual occurrence of a failure, possible ones are already identified and documented as part of a FMEA in spreadsheets, and this knowledge can be used for anomaly diagnosis. Although various ontologies model FMEA (cf. Section 5.2), they lack failure mode to function relations and the failure mechanism concept. Due to different understandings (e.g., some work modeled causes as a subclass of a failure mode), this work follows Burge's FMEA concept diagram (Burge, 2011). The resulting *FMEA ontology*[4] is shown in Fig. 7. It models *Components* (e.g., a workstation, machine, actuator, or sensor) with *Functions* and possible *FailureModes*, as well as *FailureEffects*, *FailureCauses* and *FailureMechanisms*. A *Symptom* class is introduced, which is used to describe symptoms that occur after a fault and can be used as indicators for an impending failure mode. To ensure a shared meaning of the concepts, each class has the corresponding

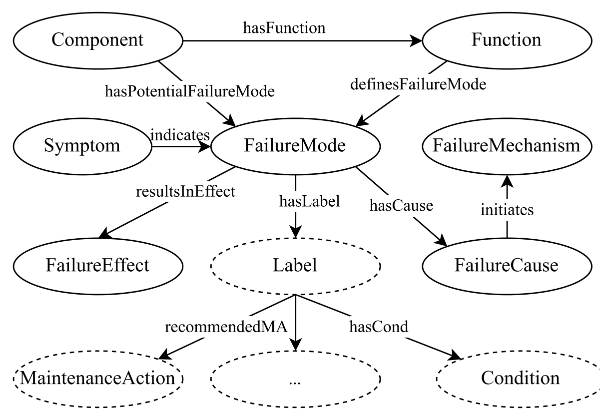

Fig. 7. Excerpt of FMEA core concepts and their relationships Extensions are illustrated with dashed lines.

International Electrotechnical Commission (IEC) definition in natural language as an annotation. Additionally, logical class constructors have been used to express further knowledge in a more formal and explicit way.

The instantiated knowledge graph used in this work[5] comprises nearly 5600 axioms and contains in total 466 classes, as well as 442 individuals. Fig. 8 shows an excerpt of a modeled failure mode, its relation to a sensor, and its effects. It depicts an acceleration sensor (*AccSen_ADXL_1*) mounted on a motor (*SM_Motor_1*) for condition monitoring purposes. The figure illustrates how vibration data insights connect to an actuator's condition in the model via a chain over related entities. By analyzing the vibration data, a symptom can be discovered that could be *Higher_Vibration* (i.e., higher amplitudes in the vibration data) which is indicative of the failure mode *Insufficient_Power_to_Drive_Conveyor_Belt* which is related to the function *Drive_Conveyor_Belt* of the motor *SM_Motor_1*. This chain enables inferences from sensor signals to components, functions, and implications for the production process. The model allows concluding that higher vibrations in the acceleration sensor (*AccSen_ADXL_1*) may indicate a bearing defect in the motor *Vibration_SM_Motor_1*, which drives the conveyor belt *SM_Conveyor_Belt*. For diagnosis, related information such as possible cause (*Defect_Bearing*), mechanism (*Wear*), and effect (*Slower/No_Transport_To_Target_Pos*) are modeled. This example demonstrates how FMEA knowledge combined with the structure of the factory model is used to create a knowledge graph, helping maintenance engineers rapidly infer causes and appropriate actions.

[4] https://gitlab.rlp.net/kleinp/public/-/blob/main/FMECA.owl, last accessed on 11/06/23.

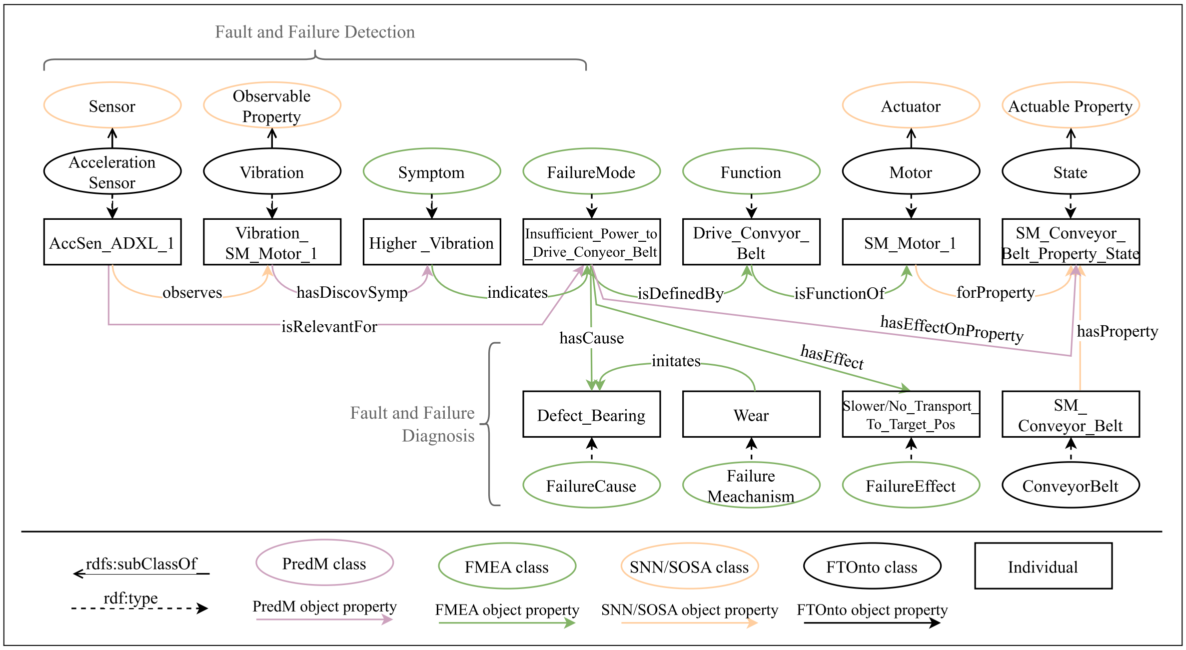

Fig. 8. Excerpt of a modeled failure mode with its relation to a sensor and its modeled effects.

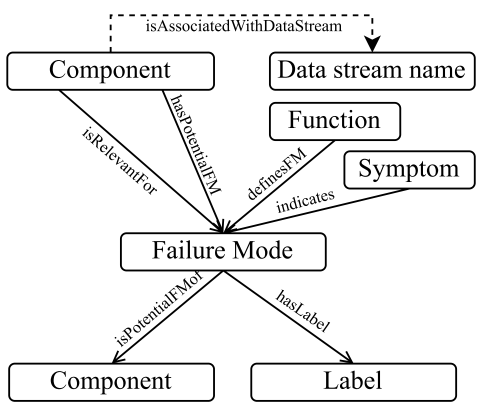

Fig. 9. The semantic failure mode description proposed in Section 3.4.1 is illustrated in the excerpt of the Knowledge Graph shown in the figure. Rectangles with rounded corners represent entities, while edges indicate object properties and dotted edges represent data properties. This sub-graph is used to define the logical constraint of Eq. (13).

3.4.2. Using a knowledge graph for root cause analysis

Of particular interest in our work is the diagnosis of anomalies using knowledge-based approaches, an area often overlooked in data-driven methods (cf. Section 5). In our case, diagnosis involves identifying the data set's label or affected component for a detected anomaly. We utilize knowledge illustrated in Figs. 7 and 9, expressed as logical formula:

$$\big[\ \text{PredM:isRelevantFor}(c, f) \tag{13a}$$

$$\lor\ \text{FMEA:hasPotentialFM}(c, f)\ \big] \tag{13b}$$

$$\land\ \text{FMEA:definesFM}(a, f) \tag{13c}$$

$$\land\ \text{FMEA:indicates}(s, f) \tag{13d}$$

$$\land\ \text{PredM:hasLabel}(f, l) \tag{13e}$$

[5] https://gitlab.rlp.net/kleinp/public/-/tree/main/, last accessed on 11/06/23.

where $\land$ is the conjunction operator (i.e., and), $\lor$ is the disjunction operator (i.e., or), c, a, f, s, c, and l are variables, and *FMECA:definesFM*, *FMECA:indicates*, *PredM:isRele-vantFor*, and *PredM:hasLabel* are predicates. We can view RCA as assigning reasonable values to these variables based on the output of data-driven anomaly detection and prior expert knowledge. If the formula evaluates to true, we have found a possible root cause candidate. We distinguish between query strategies (how we assign variables) and approaches for finding the root cause.

Query strategies. As a baseline for finding concrete labels, the following strategies are proposed for a given set C of data streams, ordered by severity of contribution to the detected anomaly:

Query 1-L (Q1-L): Iterate over each anomalous data stream in C and provide associated *labels*; start with the highest contributor to the detection.

Query 2-L (Q2-L): Iterate over a decreasing amount of combinations of multiple data streams from C and provide *labels* from the intersection of anomalous data streams; start with the highest contributor.

Query 3-L (Q3-L): Iterate over each anomalous data stream of C and provide all *labels* that are associated with components in the same workstation; start with the highest contributor.

The term *label* denotes a specific node/entity related to a failure mode, condition, etc. (cf. Fig. 7). It is expected to be the final result of a RCA and corresponds to the data set's label. From a label node, other relevant things can be retrieved, e.g., the affected component, (operational) functions, causes, recommended maintenance actions, etc. These query strategies can also be applied to identify the affected components by replacing Eq. (13e) with the predicate *PredM:isPotentialFMof* which is of great interest in the case of previously unknown or not modeled failure modes.

Query 1-C (Q1-C): Iterate over each data stream of C and provide all associated affected *components*; start with the highest contributor to the detection.

In general, integrating logically specified background knowledge as constraints on the output of a data-driven anomaly detection aims to improve performance and ensure compliance with the background knowledge, which is crucial in safety-critical environments (Giunchiglia et al., 2022; Farbiz et al., 2022; Hagendorfer, 2021). For each of query strategy, expert knowledge on failure mode detection plausibility can be integrated as constraints, further restricting found labels or affected components based on: (i) operational state of components (active/inactive) and (ii) observed symptoms. The first constraint, expressed in Eq. (13c), is based on anomalies being detectable only when the components are operational. The second, expressed in Eq. (13d), requires confirmation through observed symptoms modeled in the knowledge graph (Section 3.4.1), allowing for finer classification between failure modes for the same component. Although applicable to all queries, these constraints are evaluated only with the first query strategy, leading to the following strategies:

Q1-L + Constr. and Q1-C + Constr. To the previous queries Q1-L and Q1-C, constraints (Constr.) are added regarding (i) components' operational state and (ii) observed symptoms, further restricting found labels or affected components. The first constraint addresses the fact that anomalies are often detectable only when components are operational. The second requires symptom confirmation for specific failure modes, enabling finer classification between different failure modes for the same component.

After this subsection has outlined the general query strategies, the following subsections illustrate how these are implemented with different (AI) approaches (SPARQL, SDNR, CBR).

Approaches used for implementing the query strategies. The most intuitive method is the implementation in SPARQL (Harris and Seaborne, 2013) – the common querying language to infer knowledge from knowledge graphs. The implementation of the query strategy Q1-L, referred to as SPARQL Q1-L, is shown in Listing 1 and consists of the union of two graph patterns. This query corresponds to Eq. (13) (excluding constraint specific Eqs. (13c) and (13d)). Its execution is according Q1-L which means replacing the data streams according to the anomalous list provided by the data-driven anomaly detection model and the returned labels (i.e., results of the query) are evaluated. Note that SPARQL Q1-L + Constr. extends SPARQL Q1-L with constraints from Q1-L + Constr., formalized in Eqs. (13c) and (13d). For the implementations of the other SPARQL queries (SPARQL Q2-L, SPARQL Q3-L, SPARQL Q1-C, SPARQL Q1-L + Constr., SPARQL Q1-C + Constr.), the reader is referred to I.

Listing 1: SPARQL Q1-L is used to obtain possible labels based on a given anomalous data stream (e.g., a_15_1_y, the x-axis of an acceleration sensor).

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX ftonto: <http://iot.uni-trier.de/FTOnto#>
PREFIX fmea: <http://iot.uni-trier.de/FMEA#>
PREFIX predm: <http://iot.uni-trier.de/PredM#>

SELECT ?labels
WHERE {
 {
   ?component ftonto:is_associated_with_data_stream
        "a_15_1_y"^^xsd:string.
   ?component fmea:hasPotentialFailureMode ?failureModes.
   ?failureModes predm:hasLabel ?labels
 }UNION{
   ?component ftonto:is_associated_with_data_stream
        "a_15_1_y"^^xsd:string.
   ?failureModes predm:isDetectableInDataStreamOf_Direct
        ?component.
   ?failureModes predm:hasLabel ?labels
 }
}
```

The second approach, SDNR (cf. Section 2.7), uses knowledge graph embeddings and allows a domain expert to express logical formulas like Eq. (13). Unlike SPARQL queries, it does not guarantee valid knowledge but can predict relationships, potentially helpful for incomplete knowledge bases (Boschin et al., 2022). Such approaches are considered by itself as a combination between expert knowledge and machine learning. SDNR is applied in two variants:

SDNR Q1-L + Constr.: Evaluates a logical expression (Eq. (13)) similar to SPARQL Q1-L + Constr., assigning the anomalous data stream as fixed and varying other variables to find the highest score. Returns the label with the highest assertion.

SDNR Q1-C + Constr.: Similar to SDNR Q1-L + Constr. but replicates SPARQL Q1-C + Constr., returning affected components instead of labels.

Note that SDNR differs from simple k-Nearest Neighbors (k-NN) by allowing experts to consider specific relationships, formulate logical expressions, and integrate background knowledge as logical constraints.

A knowledge-based approach to finding the root cause is provided by CBR (Bergmann, 2002). CBR is a problem-solving paradigm that utilizes the specific knowledge of previously experienced concrete problem situations (cases) to address new problems. In our context, CBR allows us to take advantage of past experiences in identifying root causes of anomalies. As our scenario is anomaly detection where no examples (of sensor data) from previous failures are available, we have to apply analogical reasoning on between a query and instances of the knowledge graph. Unlike the two previous approaches that rely on symbolically encoded knowledge, the CBR approach applies analogical reasoning between sets of similar data streams. Whereas the two previous approaches are relying on symbolic encoded knowledge to find a root cause, the CBR approach applies analogical reasoning between sets of similar data streams represented by embedding to measure the similarity between the anomaly detection set and the sets stored in the case base. Similarly to SDNR Q1-L + Constr., knowledge graph embeddings are used to represent domain knowledge instances, and the approach is similar to the retrieval approach presented in Klein et al. (2019b). Case descriptions are represented as the sums of data stream embeddings that are related to the label (see Table 11). The label (with its knowledge graph node) corresponds to the case's knowledge item (i.e., solution). In this way, all labels and components of the data set are represented as cases in a case base, and the query strategies introduced previously can be applied to implement CBR Q1-L, CBR Q2-L, CBR Q1-C. For CBR Q1-L + Constr. and CBR Q1-C + Constr., the similarity is measured using the cosine function between the sum of the query and case embeddings. For implementing the constraint of CBR Q1-L + Constr. and CBR Q1-C + Constr., the minimum similarity is computed as:

$$\mathrm{sim}^G(a,b) = \min\{\mathrm{sim}_D(a_D,b_D), \mathrm{sim}_F(a_F,b_F), \mathrm{sim}_S(a_S,b_S)\} \tag{14}$$

where sim_D is the local similarity between the embedding vectors a_D and b_D representing the data streams of the query a and a case b. Likewise, the local similarity measure sim_F is used for the actuated function and sim_S for observed symptoms. In the case of multiple embeddings used for a or b such as in the case of the case representation b_D, the sum of the embeddings is used. The minimum function as global similarity ensures that cases that do not fit all aspects are ranked lower, considering only the lowest measured similarity as global similarity sim^G. It is important to note that our current implementation of CBR focuses primarily on the retrieval phase of the traditional 4R cycle (Retrieve, Reuse, Revise, Retain) (Aamodt and Plaza, 1994). Although this approach allows us to effectively identify similar cases for root cause analysis, the full development of the reuse, revise, and retain phases remains an area for future work.

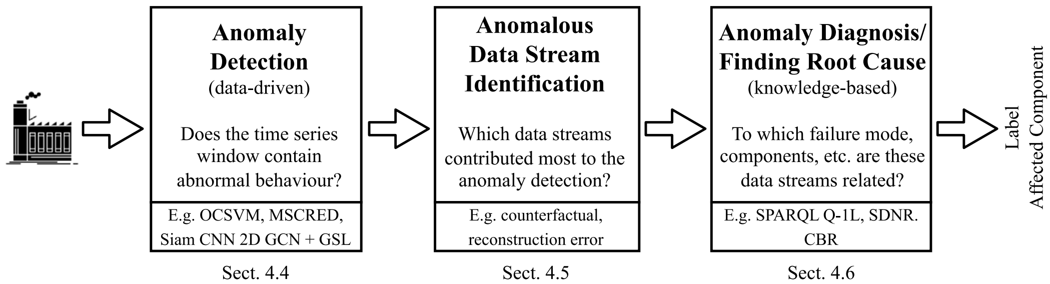

Fig. 10. Order of the steps required to evaluate the performance of finding a root cause.

Applied procedure for finding a root cause. After the query strategies and approaches are presented, the root cause finding procedure (used in the evaluation in Section 4) for a predicted anomaly is shown in Algorithm 1. Upon correct anomaly detection, the Jenks Natural Break (JNS) algorithm reduces the abnormal data stream list C to the subset C, containing only streams with the highest anomaly scores. Each stream c in C is processed using the chosen query strategy q (e.g., Q1-L, Q2-L) to find reasonable labels or affected components $\hat{l}$ until the ground truth l matches. Ground truth y and the correct root cause l are used only for evaluation, not for RCA or anomaly detection.

Algorithm 1: Implemented Procedure for Finding the Root Cause (As Used for Evaluation)

Input: Test data set $\mathcal{D}_{Test}$, know. graph $\mathcal{K}$, query strat. q, ano. pred. model $f(\cdot)$, ano. pred. ground truth y, ano. pred. by model $\hat{y}$, found label or aff. comp. $\hat{l}$, label or aff. comp. ground truth l

Result: Root cause l

for x *in* $\mathcal{D}_{Test}$ **do** `// Iterate over all test examples`
  $\hat{y} = f(x)$ `// Anomaly prediction`
  if $\hat{y} == y$ *and* $y == 1$ **then** `// Only true positives used for eval.`
    $C = f(x)$ `// Ordered list w. anomalous data streams`
    $C^* = reduce(C)$ `// Reduce to most important ones`
    for c *in* C^* **do** `// c is a abnormal data stream`
      $f, s = extract(x, c)$ `// Get functions f and symptoms s`
      $\hat{\mathcal{L}} = retrieve(\mathcal{K}, q, c, f, s)$ `// Query knowledge graph`
      for $\hat{l}$ *in* $\hat{\mathcal{L}}$ **do** `// Verify found labels`
        if $\hat{l} == l$ **then**
          return $\hat{l}$ and break `// Correct root cause found!`

4. Evaluation

The evaluation aims to validate the usefulness of combining data-driven anomaly detection with expert knowledge represented by semantic web technologies for RCA. For this purpose, the following tasks shown in Fig. 10 are evaluated:

(i) Anomaly Detection: Detecting abnormal behavior in time series windows (Section 4.4).
(ii) Anomalous Data Stream Identification: Identifying causative data streams for the detection (Section 4.5).
(iii) Anomaly Diagnosis: Finding root causes (labels or affected components) using expert knowledge from the knowledge graph (Sections 2.6 and 3.4.1) based on identified causative data streams.

While measuring the performance in these tasks, the evaluation seeks to assess whether the combination of expert knowledge and Machine Learning (ML) proposed in the framework is useful, by examining the following hypotheses:

H1 Integrating expert knowledge from the knowledge graph (Section 2.6, 3.4.1) improves anomaly detection performance.
H2 Anomaly detection models incorporating expert knowledge are more accurate in identifying relevant data streams than models without.
H3 Query strategies combining multiple causative data streams (e.g., Q2-L) find root causes more accurately.
H4 With complete and correct knowledge modeling, SPARQL queries are more precise than embedding-based approaches (SDNR, CBR). However, for incomplete knowledge bases, embedding approaches are preferable due to their relation prediction capabilities.

H1 postulates that the integration of expert knowledge into a data-driven model should lead to comparable or better performance than the same or similar model without such knowledge. Furthermore, any performance improvement suggests that expert knowledge is appropriate and valid for anomaly detection. The second **H2** focuses on improved causative stream identification. The assumption is that the integration of expert knowledge also helps the model in determining the causative streams. **H3** proposes that the combination of multiple causative data streams reduces the number of incorrect findings. Finally, **H4** postulates that if domain knowledge is perfectly modeled, working directly at the symbolic level is advantageous. However, if there are gaps in the modeled knowledge, the approaches using embeddings such as CBR and SDNR are superior due to its link prediction capabilities. In summary, hypotheses **H1** and **H2** focus on performance improvements by combining expert knowledge and machine learning, while hypotheses **H3** and **H4** are more focused on applying the developed ontology and knowledge graph for automating RCA.

4.1. Evaluated anomaly detection approaches

To evaluate the performance of the proposed anomaly detection approach and validate its compatibility with other deep learning methods, it is compared to classical baselines and a deep autoencoder approach. These are briefly introduced in the following, with further details in Appendix H.

4.1.1. Baselines

As a baseline, OC-SVM is used with different input representations and default hyperparameters from sklearn (Pedregosa et al., 2011). In addition, k-NN with $k = 1$ is employed, using various input representations and distance measures (ℓ_1, ℓ_2, and Cosine). The best performer on the validation set, consistently the Cosine distance, is chosen for the test set. Dynamic Time Warping (DTW) as measure is excluded due to prohibitive computational costs for our large dataset (more than 20,000 examples, each with 61,000 entries) (Klein et al., 2020). Furthermore, DTW's time-axis adaptation could mask anomalous differences, leading to suboptimal results. Instead, we use time series representations from ROCKET (Dempster et al., 2020) and signature matrices (as in MSCRED Zhang et al., 2019a), alongside raw data.

4.1.2. Multi-scale convolutional recurrent encoder-decoder (MSCRED) and MSCRED-AM

The Multi-scale Convolutional Recurrent Encoder-Decoder (MSCRED) framework (Zhang et al., 2019b) is used as an example of a state-of-the-art deep reconstruction-based anomaly detection approach. In Shen et al. (2020), MSCRED ranked second among 13 approaches evaluated on four datasets. How MSCRED is applied is described in detail in H.4. The MSCRED variant integrating expert knowledge through masking irrelevant correlations in the signature matrix is called MSCRED AM. AM denotes restriction by an adjacency matrix derived from the knowledge graph (cf. Appendix B). This masking, applied to input and output, aims to exclude irrelevant correlations from signature matrices based on expert knowledge. The intention is that reconstruction errors of correlations calculated from unrelated data streams are not considered, potentially reducing noise in anomaly detection. Other components and hyperparameters remain similar to MSCRED.

4.1.3. Siam 1D CNN FC and siam 2D CNN GCN+GSL

As described in Section 3.2, we use a special Siamese neural network approach (cf. Section 2.2) trained on normal data, with two encoder structures:

- Siam CNN-1D FC: 1D convolutions followed by fully connected layers.
- Siam CNN-2D GCN + GSL: 2D convolutions with graph convolutional layers, using knowledge graph embeddings for each data stream and an adjacency matrix derived from the knowledge graph. A graph structure learning module (cf. Section 2.4) learns connection strengths.

The integration of knowledge graph embeddings for encoding time series and the adjacency matrix derived from the knowledge graph by the second approach are used to infuse prior expert knowledge, resulting in an informed anomaly detection. Detailed architectures for both approaches are in H.2.

4.2. Performance measures

For each evaluation task, we present measures to compare the performance of the approach and evaluate the hypotheses. The anomaly detection performance is assessed using Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves (Goix, 2016), reporting mean ROC AUC, PR AUC, and Average Precision of Precision-Recall Curve (AvgPR) over five runs with Standard Deviation (SD). For anomaly diagnosis, we evaluate the ranked data streams using HitRate@p% and Hits@k[6] (Darban et al., 2022). To evaluate query strategies (Section 3.4.2) to identify labels or affected components, we introduce three measures:

$$\text{Query Ratio} = \frac{\#\ \text{executed queries}}{\#\ \text{correctly found labels}}. \tag{15}$$

$$\text{Label Ratio} = \frac{\#\ \text{provided labels}}{\#\ \text{correctly found labels}}. \tag{16}$$

$$\text{Coverage} = \frac{\#\ \text{correctly found labels}}{\#\ \text{true positives}}. \tag{17}$$

The Query Ratio (QR) quantifies computational effort vs. positive outcomes. Label Ratio (LR) compares the provided with correctly identified labels, preferring fewer labels for human inspection. For affected component searches, components replace labels. Coverage (Cov) measures found labels relative to true positive anomalies. The *# executed queries* corresponds to the amount of iterations made in the second for-loop of Alg. 1 and the third loop corresponds to *# provided labels* in Eq. (16).

[6] The detailed calculation is explained in Appendix J.

Table 1
Comparison of anomaly detection performance.

Inp.	Approach	Test		
		Roc Auc	Avg PR	PR Auc
Raw	kNN w. cosine	80.76	42.63	42.55
SigMat	kNN w. cosine	86.82	<u>57.51</u>	<u>57.46</u>
Rocket	kNN w. cosine	<u>87.81</u>	54.01	54.15
Raw	OCSVM	86.34	45.86	45.73
SigMat	OCSVM	77.00	33.30	33.20
Rocket	OCSVM	79.78	40.72	40.66
SigMat	MSCRED	63.46 ± 1.13	18.99 ± 1.13	18.94 ± 0.70
SigMat	MSCRED + AM	67.04 ± 1.99	23.68 ± 2.09	23.96 ± 2.23
Raw	Siam CNN1D FC	75.93 ± 9.70	39.63 ± 7.15	39.54 ± 9.70
Raw	Siam CNN2D GCN+GSL	**92.78 ± 1.23**	**72.89 ± 2.93**	**72.98 ± 1.23**

4.3. General experiment setting and evaluation protocol

The experiments are conducted on the dataset from Section 2.5, with expert knowledge provided by the knowledge graph (Sections 2.6 and 3.4.1). To evaluate the anomaly detection, each example x_i is a four-second time series window that is labeled as a whole instance either as normal (i.e., $y_i = 0$) or as a fault or failure (i.e., anomaly, $y_i = 1$) (e.g., as in Qiu et al., 2021). In the case of a labeled anomaly, the anomalous behavior can be observed either over the entire sequence (e.g., in the case of a motor Fault and Failure (FaF)), at multiple time points or sub-sequences during this period (i.e., flickering of a light barrier sensor) or only in a single sub-sequence of the whole window (e.g., delayed opening of a valve). This labeling approach is similar to Deng and Hooi (2021) method of down-sampling to 10-s medians with the most common label. This approach can be justified by the fact that time series in an industrial environment are commonly collected at a relatively high frequency (i.e., several hundreds of measurement values per second) and their interpretation is often not based on individual measurements, but on features calculated over several measurements instead. Moreover, conducting an anomaly detection for each time point of a time series sampled with a frequency of several hundred per second would theoretically lead to several hundreds of windows (corresponding to the sampling frequency of the time series) for which an anomaly detection needs to be executed, resulting in uneconomical high computational costs.

4.4. Results of the anomaly detection

Table 1 presents the usefulness of anomaly scores from baselines and deep learning approaches. The baselines (k-NN with Cosine distance and OC-SVM) outperform the reconstruction-based MSCRED on the test and validation sets. The k-NN using the signature matrix (SigMat) consistently achieves second-best results, indicating the usefulness of this representation. MSCRED's poorer performance likely stems from its reconstruction-based approach rather than its representation. Since MSCRED learns to reconstruct the input data well (cf. training loss in Fig. 18), it can be suggested that it generalizes too well, which leads to anomalies being reconstructed accurately (Gong et al., 2019). This is supported by high false negative rates (FNR) for MSCRED and MSCRED

Table 2
Comparison of deep anomaly detection approaches for identifying anomalous data streams.

Approach	True Positives	Hits@k			HitRate@p%
		k = 1	k = 3	k = 5	p=100
Random	62.2 ± 8.63	3.69 ± 2.08	12.80 ± 2.53	21.35 ± 4.45	4.24 ± 0.84
MSCRED	264.6 ± 46.06	5.79 ± 2.43	16.18 ± 4.71	24.24 ± 5.01	5.59 ± 1.17
MSCRED + AM	227.4 ± 61.68	11.32 ± 11.04	23.92 ± 14.52	32.16 ± 11.28	10.45 ± 9.59
Siam CNN1D FC	176 ± 93.15	14.56 ± 4.24 16.68 ± 3.19	30.61 ± 10.46 32.12 ± 7.05	41.35 ± 11.93 44.40 ± 7.20	17.05 ± 5.47 18.97 ± 3.90
Siam CNN2D GCN+GSL	**321.6** ± 64.23	27.04 ± 3.62 **27.07** ± 3.89	42.37 ± 3.86 **43.16** ± 4.17	49.44 ± 3.64 **50.26** ± 4.09	25.95 ± 2.90 **26.40** ± 3.71

Table 3
Results obtained by SPARQL queries and knowledge graph embeddings for finding the label for detected anomalies.

<table>
<tr><th>Retrieval Approach</th><th>Query Stragey</th><th>MSCRED</th><th>MSCRED AM</th><th>Siam CNN1D FC</th><th>Siam CNN2D GCN+GSL</th><th>Oracle (Masking)</th></tr>
<tr><td>SPARQL</td><td>Q1-L</td><td>18.9 | 11.4 | 0.7</td><td>14.8 | 10 | 0.7</td><td>12.2 | 14.2 | 0.6</td><td>7.0 | 8.9 | 0.6</td><td>1.0 | 2.4 | 1.0</td></tr>
<tr><td>SPARQL</td><td>Q2-L</td><td>320 | 19.9 | 0.5</td><td>245 | 15.7 | 0.6</td><td>112 | 17.6 | 0.6</td><td>58.1 | 12.5 | 0.6</td><td>1.0 | 2.4 | 1.0</td></tr>
<tr><td>SPARQL</td><td>Q3-L</td><td>7.6 | 20.2 | 0.9</td><td>6.6 | 18.6 | 0.9</td><td>3.3 | 17.3 | 0.9</td><td>2.6 | 16.2 | 1.0</td><td>1.1 | 5.5 | 1.0</td></tr>
<tr><td>CBR</td><td>Q1-L</td><td>16.3 | 15.0 | 0.8</td><td>11.0 | 13.5 | 0.8</td><td>8.0 | 12.2 | 0.7</td><td>5.7 | 9.4 | 0.6</td><td>1.0 | 2.4 | 1.0</td></tr>
<tr><td>CBR</td><td>Q2-L</td><td>309 | 31.2 | 0.5</td><td>189 | 15.9 | 0.6</td><td>80 | 16.4 | 0.7</td><td>38.6 | 11.4 | 0.7</td><td>1.0 | 2.3 | 1.0</td></tr>
</table>

The values of each cell represent the following metrics: (QR) | (LR) | (Cov). The best (LR) value beside the Oracle is marked in bold. A more detailed overview is provided in the appendix with Table 13.

AM. For instance, MSCRED's FNR on the test set is around 0.90, implying many undetected anomalies. The distance-based Siam CNN-2D GCN + GSL proposed achieves the best performance. Regarding the integration of expert knowledge (**H1**), MSCRED AM outperforms MSCRED, and Siam CNN-2D GCN + GSL is the best performing approach. Siam CNN-2D GCN + GSL's 128-dimensional output vector allows more efficient computations compared to larger representations of k-NN baselines (nearly 60,000 for SigMats or 20,000 for ROCKET), indicating that its learned features are sufficiently expressive. In summary, the results of anomaly detection indicate that the proposed integration of prior expert knowledge and its implementation have a positive effect and therefore satisfy **H1**. This is supported by the better performance of MSCRED AM and Siam CNN-2D GCN + GSL compared to their prior knowledge-free counterpart. Furthermore, it can be concluded that the knowledge modeled in Sections 2.6 and 3.4.1 is valid, sufficient, and useful, as well as the proposed ways to inject it (i.e., restricting via an adjacency matrix and as data stream embeddings).

4.5. Results of the identification of anomalous data streams

The bridge module (cf. Section 3.1) flags and sorts data streams based on their anomaly score contribution for diagnosis and RCA. Performance is evaluated by comparing identified streams with expert assessments (cf. Table 11, Appendix). Table 2 presents results for identifying relevant anomaly-causing data streams, focusing on deep learning approaches due to Siam CNN-2D GCN + GSL's superior performance. However, their black-box character is a frequent source of criticism and their explainability is difficult to achieve (Roelofs et al., 2021; Steenwinckel et al., 2021a). The Siamese neural networks with the counterfactual approach outperform MSCRED models in all metrics. The results from the nearest and second-nearest normal neighbors indicate potential for improvement by combining multiple counterfactuals. Comparing MSCRED and MSCRED AM reveals that considering only meaningful correlations (MSCRED AM) doubles Hits@1 and Hitrate@100% values. Siam CNN-2D GCN + GSL outperforms Siam CNN-1D FC, also supporting **H2** that expert knowledge improves performance in this task. For 25% of the correctly predicted anomalies, Siam CNN-2D GCN + GSL's counterfactual approach identified one of 61 possible causative data streams in the first position, demonstrating its usefulness.

4.6. Results of finding the data set's label and affected component

In this section, we present the results of our investigation into the effectiveness of various query strategies to identify the correct labels and affected components using well-modeled and corrupted knowledge graphs as outlined in Fig. 11, which is a common challenge for real-world applications (Boschin et al., 2022).

4.6.1. Well-modeled knowledge graph

Table 3 shows the results for finding the correct labels without constraints. Query strategy Q1-L, iterating over each anomalous data stream ordered by scores, achieves the lowest (LR) with CBR and SPARQL, with SPARQL Q1-L achieving the overall lowest (LR) at 8.9. Based on these findings, there is no support for **H3**, which expected improvement by combining multiple anomalous data streams (which is done by SPARQL Q2-L). An explanation for this is the imperfect identification of anomalous data streams (cf. Section 4.5) leading to numerous queries with inaccurate data stream combinations, resulting in a high (QR) for SPARQL Q2-L. Table 2 indicates decreasing marginal utility as k increases in Hits@k, implying more misidentified streams and unsuccessful combinations. The Oracle, representing perfect anomaly detection, finds the correct labels with its first query ((QR) = 1), leveraging Q2-L's restrictiveness to slightly improve its (LR). This improvement is not observed for SPARQL and CBR (cf. Table 13, Appendix). Furthermore, query strategy 3 (Q3-L), using the compositional structure, achieves low (QR) but higher (Cov) (> 0.9 vs. 0.5–0.7 for Q1-L and Q2-L). However, its imprecise labels lead to high (LR), which requires more manual verification, which is not desirable (see Fig. 11).

Since Q1-L achieves the lowest (LR), it is used to integrate expert knowledge through constraints considering whether the failure mode function is actuated and symptoms are present. The results reported in Table 4 show that (LR) is halved in most cases, with stable (QR) and (Cov). SPARQL Q1-L + Constr. achieves the lowest (LR) (around 4) with Siam CNN-2D GCN + GSL, which is less than double the Oracle's performance. SDNR Q1-L + Constr. delivers 1.3 or 2.4 more false labels than SPARQL Q1-L + Constr., except for Siam CNN-1D FC. Differences between CBR Q1-L + Constr., SDNR Q1-L + Constr., and SPARQL Q1-L + Constr. are attributable to knowledge graph

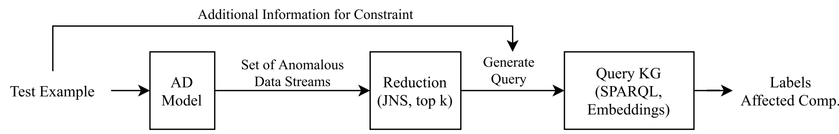

Fig. 11. Procedure of the evaluation process for finding a label or affected component for an anomaly event.

Table 4
Results obtained by SPARQL queries and knowledge graph embeddings for finding the label for detected anomalies under consideration of the logical constraint.

Retrieval Approach	Query Stragey	MSCRED	MSCRED AM	Siam CNN1D FC	Siam CNN2D GCN+GSL	Oracle (Masking)
SPARQL	Q1-L + Con.	19.1 | 6.2 | 0.7	15.2 | 5.6 | 0.7	9.9 | 7.8 | 0.6	7.0 | **4.1** | 0.6	1.0 | 2.4 | 1.0
SDNR	Q1-L + Con.	14.0 | 8.6 | 0.8	11.2 | 7.8 | 0.8	11 | 7.4 | 0.7	6.3 | 5.4 | 0.6	1.0 | 2.6 | 1.0
CBR	Q1-L + Con.	15.4 | 11.7 | 0.8	11.3 | 9.4 | 0.8	8.6 | 7.2 | 0.7	5.9 | 6.1 | 0.6	1.0 | 2.4 | 1.0

The values of each cell represent the following metrics: (QR) | (LR) | (Cov). The best (LR) value beside the Oracle is marked in bold. A more detailed overview is provided in the appendix with Table 13.

Table 5
Results obtained by SPARQL queries and knowledge graph embeddings for finding the affected component for detected anomalies.

Retrieval Approach	Query Stragey	MSCRED	MSCRED AM	Siam CNN1D FC	Siam CNN2D GCN+GSL	Oracle (Masking)
SPARQL	Q1-C	17.2 | 8 | 0.6	15 | 8.7 | 0.7	10.7 | 7.6 | 0.7	6.3 | 4.5 | 0.6	1 | 1.4 | 1
SPARQL	Q1-C + Con.	21.9 | 4 | 0.6	17.8 | 3.8 | 0.7	11.9 | 3 | 0.6	7.1 | **2.3** | 0.6	1 | 1.4 | 1
SDNR	Q1-C + Con.	16.8 | 3.8 | 0.7	9.7 | 3.9 | 0.9	8.9 | 3.6 | 0.7	6.3 | 2.8 | 0.6	1 | 1.1 | 1
CBR	Q1-C	15 | 20 | 0.8	11 | 14.9 | 0.8	8.1 | 11 | 0.7	5.7 | 6.5 | 0.6	1 | 1.2 | 1
CBR	Q1-C + Con.	15.4 | 11.2 | 0.8	11.3 | 8.5 | 0.8	8.6 | 5.7 | 0.7	5.9 | 4.2 | 0.6	1 | 1.2 | 1

The values of each cell represent the following metrics: (QR) | (LR) | (Cov). The best (LR) value beside the Oracle are marked in bold. A more detailed overview is provided in the appendix with Table 14.

embedding computations' fuzziness. Table 5 shows results for detecting affected components, a less demanding task than predicting the correct labels. Comparing Q1-C with Q1-C + Constr. reveals that the constraint introduction halves (LR). SPARQL Q1-C + Constr. with Siam CNN-2D GCN + GSL performs best (excluding Oracle), identifying affected components within 2.3 suggestions versus Oracle's 1.4. Oracle's imperfect (LR) is due to some sensors being relevant for detecting faults and failures occurring at multiple components. A further observation is that SDNR Q1-C + Constr. and CBR Q1-C + Constr. outperformed its symbolic counterpart (i.e., SPARQL Q1-C + Constr.) in (LR) with the Oracle, which is caused by a better ranking of the affected component in the returned set of candidates.

4.6.2. Corrupted knowledge graph

To simulate more likely incomplete knowledge graphs as common in the real world (Boschin et al., 2022; Li et al., 2023), the semantic knowledge graph is corrupted by deleting relevant relationships used in SPARQL Q1-L and required for finding labels or components, as well as triples necessary for constraint effectiveness (cf. Listing 2 and 3, Appendix). The corruption effects are evident in SPARQL Q1-L and SPARQL Q1-C + Constr. results with Oracle, which no longer finds all labels or components ((Cov) wo. adjustment < 1, cf. Table 15). For comparison, when SPARQL queries fail due to corruption, an effort equal to a random search on 28 failure labels (average 14 trials) or 15 components (7.5 trials) is assumed. For (QR), the number of average data streams (61/2) is used for missing labels. The results in Table 6 show that SDNR and CBR outperforming SPARQL significantly on both levels of corruption for labels and components. While SDNR and CBR performance deteriorates compared to the uncorrupted knowledge graph, their (LR) remains below (sometimes 50% below) SPARQL

Table 6
Results for the identification of anomalous data streams with a corrupted knowledge graph.

	Retrieval Approach	Query Stragey	KG Corrupted I: Siam CNN2D GCN+GSL	KG Corrupted I: Oracle (Masking)	KG Corrupted II: Siam CNN2D GCN+GSL	KG Corrupted II: Oracle (Masking)
Label	SPARQL	Q1-L	22 | 13 | 0.7	10 | 6.4 | 1	29 | 15 | 0.6	24 | 11 | 1
Label	SPARQL	Q1-L + Con.	28 | 12 | 0.7	13 | 7.4 | 1	36 | 14 | 0.6	26 | 12 | 1
Label	SDNR	Q1-L + Con.	5.2 | 7.3 | 0.7	1 | 4.3 | 1	7.9 | 9.8 | 0.6	1.2 | 5.2 | 1
Label	CBR	Q1-L + Con.	7.5 | **6.8** | 0.7	1 | **2.4** | 1	5.4 | **6.9** | 0.6	1 | **2.4** | 1
Component	SPARQL	Q1-C	25 | 5.9 | 0.6	20 | 5 | 1	27 | 5.9 | 0.6	26 | 5.8 | 1
Component	SPARQL	Q1-C + Con.	33 | 6.0 | 0.6	23 | 5.5 | 1	36 | 6.3 | 0.6	28 | 6.4 | 1
Component	SDNR	Q1-C + Con.	6.3 | **2.3** | 0.6	1 | **1.2** | 1	7.2 | **3.0** | 0.6	1 | **1.1** | 1
Component	CBR	Q1-C + Con.	5.8 | 4.2 | 0.6	1 | **1.2** | 1	5.4 | 4.8 | 0.6	1 | 1.2 | 1

The values of each cell represent the following metrics: (QR) | (LR)| (Cov) whereas for all approaches the reported values are adjusted for not found labels and components to approach that returned the highest amount. The best (QR) and (LR) value for each anomaly detection approach are marked in bold. A more detailed overview is provided in the appendix with Table 15.

with random search. These experimental results confirm H4: symbolic queries (SPARQL) perform better with perfectly modeled knowledge graphs, while embedding-based approaches (SDNR and CBR) achieve stronger performance in imperfect and incomplete real-world scenarios.

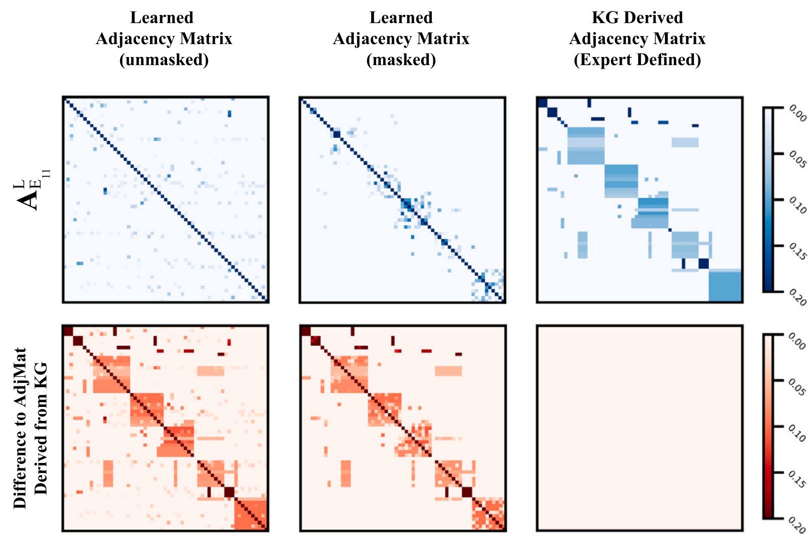

Fig. 12. Comparison of the learned adjacency matrix with the masked version based on the one that is defined based on expert knowledge (i.e., derived from knowledge graph).

4.7. Results of the ablation study on graph structure learning for expert knowledge integration

An ablation study regarding approaches for learning the graph structure with respect to expert knowledge integration in the form of an adjacency matrix derived from the knowledge graph is analyzed in the following section. Various methods for learning relationships between data streams are proposed in the literature (cf. Section 2.4). We apply different graph structure learning modules to the proposed Siamese neural network encoder (Siam CNN-2D GCN + GSL) with k-NN reduction, adjacency matrix masking, and preprocessing for graph convolutional neural networks. The results in Table 7 indicate that all graph structure learning approaches improve or match performance compared to the model without such a module. Embedding-based variants for directed and unidirected graphs ($A^L_{\|E_{11}\|_2}$ (cf. Eq. (9)), $A^L_{\|E_{12}\|_2}$ (cf. Eq. (10)), $A^L_{E_{11}}$, cf. Eq. (6)) yield comparable performance and obtain the best results. Learning the parameters of the adjacency matrix directly (A^L_W) outperforms replacing the graph structure learning module with a graph attention network layer. Wu et al.'s uni-directed variant ($A^L_{E_{12}-E_{21}}$) performs worst, suggesting that relationships are best represented by symmetric or directed graphs.

We investigate the integration of expert knowledge through the adjacency matrix A^E derived from the knowledge graph. Masking irrelevant relationships (Eq. (12)) yields better results than merging learned and predefined adjacency matrices via $A^\star = \beta A^E + (1-\beta)A^L$ of Zhu et al. (2021) (Table 8). However, learning the model without masking the adjacency matrix with the predefined one yielded a comparable performance (second last row of Table 8). For this reason, Fig. 12 compares two learned adjacency matrices: left without and middle with masking. On the right is the adjacency matrix derived from the knowledge graph. The adjacency matrix derived from expert knowledge A^E exhibits strong local relationships between components, visible as connections near the diagonal line in the plot. This preference is absent in the unmasked learned variant, where the relationships are evenly distributed throughout the matrix. The difference graph reveals many learned relationships that do not match expert-defined ones and lack obvious causal dependency. When masking is applied, the graph structure learning module appears to focus on expert-identified relationships, learning their weights, as shown in the middle top plot. Although a different graph structure is learned, an equivalent performance is achieved. This might suggest graph structure irrelevance; however, improved results with graph structure learning and weaker performance when altering structure (e.g., merging with pre-defined adjacency matrix) indicate that coincidental correlations are being learned. Such correlations are undesirable for the robustness and explainability of the model, which can be mitigated through the proposed masking.

Table 7
Results for different graph structure learning approaches.

GSL Approach	Test		
	Roc Auc	Avg PR	PR Auc
Siam CNN2D GCN+GSL with A^L_W	89.82 ± 1.72	64.60 ± 3.10	64.62 ± 3.19
Siam CNN2D GCN+GSL with $A^L_{E_{11}}$	**92.78 ± 1.23**	**72.89 ± 2.93**	**72.98 ± 1.23**
Siam CNN2D GCN+GSL with $A^L_{E_{12}}$	90.55 ± 2.93	68.54 ± 5.19	68.59 ± 5.17
Siam CNN2D GCN+GSL with $A^L_{E_{12}-E_{21}}$	87.20 ± 3.40	64.76 ± 4.77	64.93 ± 4.82
Siam CNN2D GCN+GSL with $A^L_{\|E_{11}\|_2}$	92.37 ± 0.98	70.82 ± 3.52	70.82 ± 3.55
Siam CNN2D GCN+GSL with $A^L_{\|E_{12}\|_2}$	92.79 ± 0.91	73.16 ± 2.86	73.26 ± 2.87
Siam CNN2D GAT (5 heads, concat, DO=0.3)	71.65 ± 6.88	36.39 ± 7.61	33.23 ± 8.10

Table 8
Results for different parameters/designs of the graph structure learning module.

Changes in GSL Design for Siam CNN2D GCN+GSL	Test		
	Roc Auc	Avg PR	PR Auc
Allow 5 Highest Masked Edges	93.03 ± 0.83	72.99 ± 2.30	73.06 ± 2.31
Allow 10 Highest Masked Edges	93.30 ± 0.10	73.67 ± 2.81	73.78 ± 2.82
Allow 20 Highest Masked Edges	92.83 ± 0.96	72.56 ± 2.79	72.51 ± 2.79
w/o kNN Reduction	88.91 ± 2.42	64.37 ± 5.81	64.54 ± 5.81
GSL initialised w. pre-trained Embeddings	92.52 ± 1.80	72.93 ± 5.21	73.00 ± 5.22
Merged with Predefined A^E (beta=0.3)	88.30 ± 6.46	64.38 ± 9.41	64.29 ± 9.39
w/o Adj Masking	93.03 ± 0.61	72.35 ± 2.29	72.44 ± 2.33
Merged with Predefined A^E (beta=0.3)	86.14 ± 4.23	56.90 ± 7.49	56.95 ± 7.51

4.8. Comparison of run times

Runtime is crucial in assessing the efficiency of anomaly detection and diagnosis approaches. The experiments have been conducted in a shared GPU environment,[7] which may affect computation times. The used implementation prioritizes experimental ease over speed optimization, but assumes equal impact across variants. In our experiments, OC-SVM requires the longest training time: 12 h using raw data, 10-11 h using signature matrices, and 3.5 h using ROCKET representation.

[7] Each experiment typically uses a dedicated NVIDIA V100 GPU and shares 40 Intel Xeon Gold 6138 CPU @ 2.00 GHz with two cores and 750 GB RAM.

In contrast, Siam CNN-1D FC and Siam CNN-2D GCN + GSL require only 5 and 11 min, respectively. Processing times for single examples are 0.01 and 0.03 s, respectively, for the Siamese networks. The cosine similarity computation for the entire test set takes less than 10 s, significantly faster than k-NN on raw data (ca. 10 min). The time complexity of k-NN is $O(n \times m)$, where n is the number of examples and m is the number of features (Raschka, 2020). Generating counterfactuals takes 0.48 s for Siam CNN-1D FC and 0.81 s for Siam CNN-2D GCN + GSL. Complete anomaly detection with Siam CNN-2D GCN + GSL takes less than 2 s per example (0.1 s for pre-processing, 0.03 s for encoding, 0.1 s for executing k-NN on an encoded case base plus less than 1 s to generate a counterfactual).

SPARQL queries for anomaly diagnosis using OWLReady2 (Lamy, 2017) take 0.000052 s for Q1-L and 0.0002 s for SPARQL Q1-C + Constr.. SDNR Q1-L + Constr. and SDNR Q1-C + Constr. take nearly 2 s per query, primarily due to implementation prioritizing comprehensibility over efficiency. In conclusion, detecting and diagnosing an anomaly requires approximately 2 s per example using SPARQL queries and 4 s using SDNR.

5. Related work

First, related work w.r.t. the applied data-driven anomaly detection approaches is presented in Section 5.1. Then Section 5.2 discusses how other work modeled expert knowledge by employing semantic web technologies for knowledge-based PredM approaches. Finally, Section 5.3 presents approaches that combine data-driven anomaly detection with knowledge-based approaches.

5.1. Data-driven anomaly detection

Anomaly detection is relevant in domains such as cyber security (Li et al., 2019; Huong et al., 2021), robotics (Park et al., 2017), manufacturing (Malburg et al., 2023b), and failure detection (Theusch et al., 2021; Luo et al., 2021). Due to the diversity, this discussion focuses on distance-based approaches, representations obtained by self-supervised models, and Siamese neural networks. Skvára et al. (2018) found that k-NN outperforms generative deep learning models on non-image data sets, when there are fewer anomalous examples (which are required for finding optimal hyperparameters). Alimohammadi and Nancy Chen (2022) evaluated eight algorithms for anomaly detection on time series from oil and gas production, with k-NN performing best. Combining deep features with k-NN has achieved state-of-the-art performance (Bergman et al., 2020; Reiss et al., 2021; Roth et al., 2022). Bergman et al. (2020) demonstrated superior performance using k-NN with a 2056-dimensional feature vector from a ResNet encoder pre-trained on ImageNet (He et al., 2015). Reiss et al. (2021) further improved this approach using the loss specified in Eq. (2). Roth et al. (2022) achieved near-perfect performance in industrial image anomaly detection by concatenating multiple ResNet feature blocks. However, the use of pre-trained models is debatable and those are scarce for time series data (Liznerski et al., 2022), which has led to the exploration of self-supervised approaches (cf. Section 2.2) that have shown strong improvements in image anomaly detection (Golan and El-Yaniv, 2018; Sohn et al., 2021). Deng and Hooi's graph deviation network (Deng and Hooi, 2021) targets anomaly detection in multivariate time series for CPSs. This method learns the structure of the graph using embeddings and uses a graph neural network to represent each data stream. Siamese Neural Networks are rarely used for anomaly detection due to the lack of negative examples (i.e., anomalies) during training. However, Alaverdyan et al. (2020) integrated an autoencoder architecture into a Siamese neural network to detect subtle brain lesions in MRI scans. Castellani et al. (2020) proposed weakly supervised anomaly detection using synthetically generated data from a digital twin. Hashemi and Mäntylä (2021) trained a Siamese neural network with negative pairs (i.e., anomalies) and used the learned transformation with k-NN or a classifier. Masana et al. (2018) utilized images from other datasets to generate negative pairs, measuring Euclidean distance on encoded embeddings as an anomaly detector.

5.2. Semantically modeled expert knowledge for knowledge-based predictive maintenance approaches

This section presents related work using semantic web technologies to describe fault and failure knowledge in knowledge-based PredM approaches. Although some works developed their own concepts (Günel et al., 2013; Mazzola et al., 2016; Cao et al., 2019), many are based on the FMEA methodology (Müller et al., 2020; Burge, 2011; Pecht and Gu, 2009; Guo et al., 2019). Zhou et al. (2015) and Xu et al. (2018a) modeled relationships between failure modes and components, effects, and causes through class subsumption. Steenwinckel et al. (2018) expanded on Zhou's concepts to build the Folio ontology, which is aligned with the SSN ontology by defining Folio's *LocalEffect* class as a subclass of SSN's *Observation* class, and a property called *happenedAt*. Ali and Hong (2018) used specific object properties to relate causes, effects, and components to failures (and not via subClassOf). Nuñez and Borsato's OntoProg (Lira and Borsato, 2018) introduced the concept of symptoms related to potential failure causes to monitor the condition of a pump with means of vibration analysis. The recent work by Ali and Hong (2018) used object properties instead of subsumption relationships. Failure modes are directly linked to a *ManufacturingItem*, with an item's function modeled as a literal. In conclusion, while Cao et al.'s model (Cao et al., 2019) initially considers a manufacturing environment, its implementation and evaluation are presented using an acceleration sensor of a bearing failure, as well as other works focusing on wind turbines, loaders, trains, pumps, or video cameras for surveillance. In summary, most approaches have adopted FMEA concepts from Zhou et al. (2015), modeling effects and causes as subclasses of failure modes (instead of disjoint entities) and previous works lack the aspect that failure modes in FMEA always refer to a component's function (Burge, 2011). Including this could provide valuable information for supervisory control and maintenance planning, ensuring the reliable functioning of important functions. Knowing which function is affected could provide valuable information for supervisory control to determine available manufacturing capabilities, as well as for maintenance planning to ensure reliable functioning of important capabilities.

5.3. Data-driven anomaly detection combined with knowledge-based approaches

Research combining data-driven and knowledge-based techniques for anomaly detection, particularly for time series data with semantic web technologies, is scarce (De Paepe et al., 2021). Dalzochio et al. (2020) found no and Franciosi et al. (2024) mentioned five studies combining machine learning and ontology reasoning for PredM. The most similar work is by Steenwinckel et al. (2021a) who proposed FLAGS, a methodology that combines data-driven and knowledge-based methods for anomaly detection, fault recognition, and root cause analysis. It semantically enriches time series, applies rule matching, and uses the Matrix Profile algorithm (Yeh et al., 2016) for data-driven detection. FLAGS has been demonstrated on railway accelerometer data, while the geographical location and previously stored anomalies are primarily considered for RCA. It is demonstrated using data from a 1-axis shock pulse accelerometer from a railway company, where the RCA component can distinguish between anomalies caused by railway track or bogie faults. Cao et al. (2019) combine a semantic model with rules learned from a decision tree to reason if instances are in normal condition or failures occur. Another work by Cao et al. (2022) built a knowledge-based PredM system by mining a sequential pattern from event data to obtain rules that are integrated in a semantic model on which ontological reasoning is applied. Nyulászi et al. (2018) combined data-driven models with a knowledge-based system using three if-then rules: if one anomaly is detected, the fault type is a sensor fault; if three or more are detected, it is an engine fault. Radtke and Bock (2022) developed a supervised fault classification method for bearings using logic tensor networks that combines logical rules with statistical features (e.g., kurtosis) and constants as thresholds.

In conclusion, there is a lack of research on the combination of knowledge-based and data-driven methods to detect the cause of an anomaly in the context of the Internet of Things (IoT) (De Paepe et al., 2021) to improve PredM (Franciosi et al., 2024) as well as reported in the application domain of manufacturing to combine machine learning with semantic web technologies (Breit et al., 2023). Specifically, there is a need for further exploration of combinations with deep learning techniques for processing time series data, such as the commonly used autoencoder architecture for anomaly detection (Ruff et al., 2020). Some of the mentioned anomaly detection approaches (e.g., Zhao et al., 2020b,a; Zhang et al., 2019a; Khoshnevisan and Fan, 2019; Bulla and Birje, 2021; Deng and Hooi, 2021) provide commonly used types of explanations or interpretations (Darban et al., 2022), but those approaches end at this point and do not consider a knowledge-based RCA. Our paper addresses this research gap by proposing a framework focusing on integrating deep learning-based anomaly detection with semantic web-based knowledge graph for RCA, to advance the field of PredM in IoT and manufacturing contexts.

6. Summary of contributions and discussion of limitations

This paper presents a framework that combines deep learning-based anomaly detection with knowledge-based RCA for PredM in CPPSs. Building on data-driven approaches (Section 5.1), we propose an informed self-supervised one-class learning method that incorporates domain knowledge from data stream relationships and knowledge graph embeddings, outperforming several baselines and a deep autoencoder. Our work extends related research by integrating deep anomaly detection with knowledge-based RCA, addressing gaps identified in Section 5.3, where previous combinations lacked deep learning for anomaly detection or on the other side, RCA investigation. For RCA, we utilize a knowledge graph based on semantic web technologies, proposing query strategies and AI approaches for logical inference using SPARQL, knowledge graph embeddings with SDNR, and CBR. This extends previous work which focused mainly on logical inference and reasoning (cf. Section 5.3) by investigating embedding techniques. We contribute a FMEA ontology for modeling fault and failure knowledge, addressing the need for domain-specific knowledge representation (Section 5.2). This ontology aligns with common FMEA concepts, a domain ontology, and a data set generated by a smart factory model. In summary, this comprehensive framework aims to advance PredM capabilities in manufacturing contexts by seamlessly integrating deep learning-based anomaly detection with semantic web-based knowledge graphs for robust RCA, thereby filling a significant research gap in IoT and manufacturing contexts (cf. Section 5.3).

Our work's primary limitation is the evaluation on a single instance (time series dataset and corresponding knowledge graph), but this is consistent with related work (cf. Section 5.3). This constraint is due to the scarcity of comprehensive publicly accessible datasets with fault and failure time series and corresponding expert knowledge (Klein. and Bergmann., 2019; De Paepe et al., 2021), required to evaluate integrated data-driven and knowledge-based approaches in PredM. Studies integrating expert knowledge typically focus on physical knowledge at the component level (Nunes et al., 2023). However, our data set offers advantages over those in comparable studies, featuring diverse sensors and more failure modes. Physical models with small-scale industrial components have similar characteristics regarding runtime properties and cyber–physical interactions to real-world production environments while being easier to operate (Abele et al., 2017). Our knowledge representation employs established upper ontologies constructed for real industrial settings, potentially facilitating transfer to other settings. While using a single dataset limits generalizability, our framework's design, incorporating standard semantic web technologies, and adaptable deep learning architectures, suggests potential applicability to other industrial contexts. We used two deep learning approaches and baselines to demonstrate generalizability and compare performance. As is common for neural networks, applying the proposed approaches to other data would require hyperparameter tuning (Serradilla et al., 2020b). However, the general methods for integrating and combining expert knowledge (e.g., reasoning from anomalous data streams to affected components, applying logical constraints on active operations and symptoms) appear logically reasonable and potentially applicable to other datasets or domains with similar characteristics.

7. Conclusion and future work

This work presents a framework that combines deep data-driven anomaly detection models with expert knowledge for subsequent RCA using a knowledge graph. The framework utilizes the commonly used explainable outputs of deep anomaly detection models (causative data streams) as entry points to a semantic web-based knowledge graph for identifying data set labels or affected CPPS components. To ensure that the data streams that caused the model to make its prediction correspond to those an expert would expect, an informed distance-based deep one-class approach using a specific Siamese neural network with a counterfactual approach is proposed. The evaluated models integrate background knowledge through an adjacency matrix derived from a knowledge graph, and using active functions and symptoms as diagnostic constraints. The evaluation shows that integrating expert knowledge about data stream relationships via masking irrelevant correlations or using graph convolutional networks and knowledge graph embeddings can improve anomaly detection performance. For diagnosis, combining the knowledge graph with deep anomaly detection outputs effectively identifies labels or affected components using SPARQL queries, SDNR, or CBR with graph embeddings. The latter two methods demonstrate their ability to infer relations in incomplete knowledge representations typical of real-world scenarios (Boschin et al., 2022). Although assessing integration approaches on a single instance is a limitation, the scarcity of suitable research data (Klein. and Bergmann., 2019; De Paepe et al., 2021) required this approach. In general, the results support the usability of the framework for combining data-driven anomaly detection with knowledge-based models for diagnostics, contributing to research on the integration of expert knowledge to improve PredM with limited fault and failure data.

Future work could improve the performance of anomaly detection and identification of affected data streams, which should enhance subsequent RCA. Investigating more expressive knowledge graph embedding approaches for SDNR and CBR could be beneficial. Extending our CBR approach to incorporate the reuse, revise, and retain phases of the 4R cycle could further improve the performance and lead to a self-learning system. Successful identification of anomalous data streams would increase the confidence in prediction in terms of eXplainable Artificial Intelligence (XAI) (Vollert et al., 2021), with user interaction for generating counterfactuals and expert knowledge helping this process (Beckh et al., 2021). Integrating semantic constraints as an additional loss function in deep anomaly detection (e.g., Xu et al., 2018b; von Hahn and Mechefske, 2022) could improve normal state representations. Interoperability is crucial for integrating the results into other systems. Sharing results with higher-level systems, such as manufacturing execution system and production planning, could enable more efficient scheduling of production and maintenance tasks. An interface could map affected FMEA function instances to the corresponding manufacturing tasks (expressed as semantic web services Malburg et al., 2020), guiding process execution planning (Malburg et al., 2023a,b).

CRediT authorship contribution statement

Patrick Klein: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Lukas Malburg:** Writing – review & editing, Validation, Project administration,

Table 9
Class distribution of the used data set.

Label	Train (after split)	Valid (after split)	Test
no_failure	22763	326	2907
txt15_conveyor_failure_mode_driveshaft_slippage...	9	2	0
txt15_i1_lightbarrier_failure_mode_1	4	0	0
txt15_i1_lightbarrier_failure_mode_2	3	0	18
txt15_i3_lightbarrier_failure_mode_2	7	1	5
txt15_m1_t1_high_wear	71	3	88
txt15_m1_t1_low_wear	103	3	79
txt15_m1_t2_wear	33	3	58
txt15_pneumatic_leakage_failure_mode_1	9	2	0
txt15_pneumatic_leakage_failure_mode_2	10	2	0
txt15_pneumatic_leakage_failure_mode_3	2	2	0
txt16_conveyor_failure_mode_driveshaft_slippage...	9	2	12
txt16_conveyorbelt_big_gear_tooth_broken_failure	11	2	11
txt16_conveyorbelt_small_gear_tooth_broken_failure	3	0	0
txt16_i3_switch_failure_mode_2	7	2	0
txt16_i4_lightbarrier_failure_mode_1	28	3	42
txt16_m3_t1_high_wear	39	3	26
txt16_m3_t1_low_wear	7	2	20
txt16_m3_t2_wear	59	3	43
txt17_i1_switch_failure_mode_1	15	2	15
txt17_i1_switch_failure_mode_2	21	3	11
txt17_pneumatic_leakage_failure_mode_1	16	2	9
txt17_workingstation_transport_failure_mode_wou...	13	2	18
txt18_pneumatic_leakage_failure_mode_1	9	2	9
txt18_pneumatic_leakage_failure_mode_2	24	3	10
txt18_pneumatic_leakage_failure_mode_2_faulty	9	2	8
txt18_transport_failure_mode_wout_workpiece	6	0	0
txt19_i4_lightbarrier_failure_mode_1	7	2	0
txt19_i4_lightbarrier_failure_mode_2	6	2	0
Sum	23303	381	3389

Examples are part of different run-to-failure recordings in the test set that are not contained in the train or validation set.

Methodology, Funding acquisition, Data curation. **Ralph Bergmann:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Data set

See Tables 9–11.

Appendix B. Algorithmically deriving an adjacency matrix from a knowledge graph

The used graph convolutional layer typically requires the graph structure in the form of an adjacency matrix as well as it was used to mask irrelevant correlations in MSCRED referred to as MSCRED AM. As manually defining them for numerous data streams such as sensors is a complex and time-consuming task, an adjacency matrix can be generated by using existing semantic knowledge graphs and previously defined International Resource Identifiers (IRIs). This provides the advantage over manual definition that algorithmic determination can be made based on a multitude of relationships modeled in detail, and each entry of the adjacency matrix can be explained by the relationship between entries provided by the knowledge graph. The resulting adjacency matrix $A \in \{0,1\}^{m \times m}$ for m data streams has binary entries A_{ij} which indicate whether there is a dependency between data streams d_i and d_j or not. All query patterns used to derive the adjacency matrix are presented in Table 12 (see Fig. 13).

Table 10
Overview of labels and their affected component.

Affected Component	Label
FTOnto#VGR_Compressor_7	PredM#Label_txt18_pneumatic_leakage_failure_mode_1
FTOnto#VGR_Compressor_7	PredM#Label_txt18_pneumatic_leakage_failure_mode_2_faulty
FTOnto#VGR_Compressor_7	PredM#Label_txt18_pneumatic_leakage_failure_mode_2
FTOnto#MPS_Position_Switch_3	PredM#Label_txt16_i3_switch_failure_mode_2
FTOnto#BF_Position_Switch_1	PredM#Label_txt17_i1_switch_failure_mode_2
FTOnto#BF_Position_Switch_1	PredM#Label_txt17_i1_switch_failure_mode_1
FTOnto#MPS_Motor_3	PredM#Label_txt16_m3_t2_wear
FTOnto#MPS_Motor_3	PredM#Label_txt16_m3_t1_high_wear
FTOnto#MPS_Motor_3	PredM#Label_txt16_m3_t1_low_wear
FTOnto#SM_Motor_1	PredM#Label_txt15_m1_t2_wear
FTOnto#SM_Motor_1	PredM#Label_txt15_m1_t1_high_wear
FTOnto#SM_Motor_1	PredM#Label_txt15_m1_t1_low_wear
FTOnto#MPS_WorkstationTransport	PredM#Label_txt17_pneumatic_leakage_failure_mode_1
FTOnto#MPS_Conveyor_Belt	PredM#Label_txt16_conveyor_small_gear_tooth_broken_failure
FTOnto#SM_Light_Barrier_3	PredM#Label_txt15_i3_lightbarrier_failure_mode_2
FTOnto#MPS_Conveyor_Belt	PredM#Label_txt16_conveyor_big_gear_tooth_broken_failure
FTOnto#SM_Light_Barrier_1	PredM#Label_txt15_i1_lightbarrier_failure_mode_2
FTOnto#SM_Light_Barrier_1	PredM#Label_txt15_i1_lightbarrier_failure_mode_1
FTOnto#HRS_Light_Barrier_4	PredM#Label_txt19_i4_lightbarrier_failure_mode_2
FTOnto#SM_Compressor_8	PredM#Label_txt15_pneumatic_leakage_failure_mode_1
FTOnto#HRS_Light_Barrier_4	PredM#Label_txt19_i4_lightbarrier_failure_mode_1
FTOnto#SM_Compressor_8	PredM#Label_txt15_pneumatic_leakage_failure_mode_2
FTOnto#SM_Compressor_8	PredM#Label_txt15_pneumatic_leakage_failure_mode_3
FTOnto#MPS_Conveyor_Belt	PredM#Label_txt16_conveyor_failure_mode_driveshaft_slippage
FTOnto#SM_Conveyor_Belt	PredM#Label_txt15_conveyor_failure_mode_driveshaft_slippage
FTOnto#MPS_Compressor_8	PredM#Label_txt17_pneumatic_leakage_failure_mode_1
FTOnto#MPS_Light_Barrier_4	PredM#Label_txt16_i4_lightbarrier_failure_mode_1
FTOnto#FT_VacuumSuctionGripper	PredM#Label_txt18_transport_failure_mode_wout_workpiece

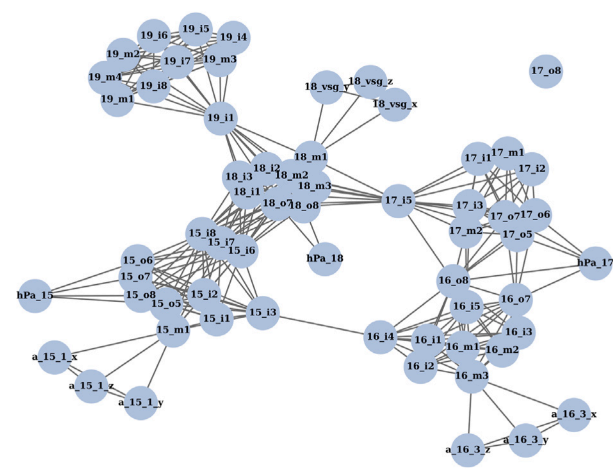

Fig. 13. Derived adjacency matrix depicted as a graph.

Appendix C. MSCRED reconstruction error-based explanation

See Fig. 14.

Appendix D. Counterfactual explanation of a detected anomaly

Two visualizations of the counterfactual approach[8] used to identify anomalous data streams in case of a false positive and a true positive are shown in Figs. 15 and 16. The example shows the data streams that are identified as anomalous according the procedure of Section 3.3.

Appendix E. Full results of anomaly diagnosis

See Tables 13–15.

[8] A collection with counterfactual explanations for all FaFs of the whole data set can be found on https://drive.google.com/drive/folders/1-D-55W1UngpGFu8zpGkQGuZ9DdjE07Fx?usp=sharing (last accessed on 08/09/2022).

Table 11
Relevant features per failure Mode/Label.

Label	Direct Features	Contextual Features
txt15_conveyor_failure_mode_driveshaft_slippage_failure	a_15_1_x, a_15_1_, ya_15_1_z	txt15_m1.finished, a_15_1_x, a_15_1_y, a_15_1_z
txt15_i1_lightbarrier_failure_mode_1	txt15_i1	txt15_m1.finished, txt15_i1
txt15_i1_lightbarrier_failure_mode_2	txt15_i1	txt15_m1.finished, txt15_i1
txt15_i3_lightbarrier_failure_mode_2	txt15_i3	txt15_m1.finished, txt15_i3
txt15_m1_t1_high_wear	a_15_1_x, a_15_1_y, a_15_1_z	txt15_m1.finished, a_15_1_x, a_15_1_y, a_15_1_z
txt15_m1_t1_low_wear	a_15_1_x, a_15_1_y, a_15_1_z	txt15_m1.finished, a_15_1_x, a_15_1_y, a_15_1_z
txt15_m1_t2_wear	a_15_1_x, a_15_1_y, a_15_1_z	txt15_m1.finished, a_15_1_x, a_15_1_y, a_15_1_z
txt15_pneumatic_leakage_failure_mode_1	hPa_15	hPa_15 hPa_15, txt15_o5, txt15_o6, txt15_o7, txt15_o8
txt15_pneumatic_leakage_failure_mode_2	hPa_15	hPa_15 hPa_15, txt15_o5, txt15_o6, txt15_o7, txt15_o8
txt15_pneumatic_leakage_failure_mode_3	hPa_15	hPa_15 hPa_15, txt15_o5, txt15_o6, txt15_o7, txt15_o8
txt16_conveyor_failure_mode_driveshaft_slippage_failure	a_16_3_x, a_16_3_y, a_16_3_z	a_16_3_x, a_16_3_y, a_16_3_z, txt16_m3.finished
txt16_conveyorbelt_big_gear_tooth_broken_failure	a_16_3_x, a_16_3_y, a_16_3_z	a_16_3_x, a_16_3_y, a_16_3_z, txt16_m3.finished
txt16_conveyorbelt_small_gear_tooth_broken_failure	a_16_3_x, a_16_3_y, a_16_3_z	a_16_3_x, a_16_3_y, a_16_3_z, txt16_m3.finished
txt16_i3_switch_failure_mode_2	txt16_i3	txt16_i3, txt16_m1.finished
txt16_i4_lightbarrier_failure_mode_1	txt16_i4	txt16_i4, txt16_m3.finished
txt16_m3_t1_high_wear	a_16_3_x, a_16_3_y, a_16_3_z	a_16_3_x, a_16_3_y, a_16_3_z, txt16_m3.finished
txt16_m3_t1_low_wear	a_16_3_x, a_16_3_y, a_16_3_z	a_16_3_x, a_16_3_y, a_16_3_z, txt16_m3.finished
txt16_m3_t2_wear	a_16_3_x, a_16_3_y, a_16_3_z	a_16_3_x, a_16_3_y, a_16_3_z, txt16_m3.finished
txt17_i1_switch_failure_mode_1	txt17_i1	txt17_i2, txt17_i1, txt17_o7, txt17_m1.finished, txt17_i5, txt16_o8
txt17_i1_switch_failure_mode_2	txt17_i1	txt17_i2, txt17_i1, txt17_o7, txt17_m1.finished, txt17_i5, txt16_o8
txt17_pneumatic_leakage_failure_mode_1	hPa_17	hPa_17, txt17_o7, txt17_i5, txt17_o5, txt17_o6
txt17_workingstation_transport_failure_mode_wout_workpiece	hPa_17	hPa_17, txt17_i5, txt17_o7, txt17_m1.finished, txt17_o5, txt17_m2.finished, txt16_o8, txt17_o6, txt16_o7
txt18_pneumatic_leakage_failure_mode_1	hPa_18	hPa_18, txt18_o7, txt18_o8
txt18_pneumatic_leakage_failure_mode_2	hPa_18	hPa_18, txt18_o7, txt18_o8
txt18_pneumatic_leakage_failure_mode_2_faulty	hPa_18	hPa_18, txt18_o7, txt18_o8
txt18_transport_failure_mode_wout_workpiece	hPa_18	hPa_18, txt18_o7, txt18_o8, txt18_m2.finished, txt18_m3.finished, txt18_m1.finished
txt19_i4_lightbarrier_failure_mode_1	txt19_i4	txt19_m1.finished, txt19_m2.finished, txt19_m3.finished, txt19_m4.finished, txt19_i4
no_failure	txt15_i1, txt15_i3, txt16_i3, txt16_i4, txt17_i1, txt19_i4, a_15_1_x, a_15_1_y, a_15_1_z, a_16_3_x, a_16_3_y, a_16_3_z, hPa_15, hPa_17 hPa_18	txt15_i1, txt15_i3, txt16_i3, txt16_i4, txt17_i1, txt17_i2, txt17_i5, txt19_i4, txt15_m1.finished, txt16_m1.finished, txt16_m3.finished, txt17_m1.finished, txt17_m2.finished, txt18_m1.finished, txt18_m2.finished, txt18_m3.finished, txt19_m1.finished, txt19_m2.finished, txt19_m3.finished, txt19_m4.finished, txt15_o5, txt15_o6, txt15_o7, txt15_o8, txt16_o7, txt16_o8, txt17_o5, txt17_o6, txt17_o7, txt17_o8, txt18_o7, txt18_o8, a_15_1_x, a_15_1_y, a_15_1_z, a_16_3_x, a_16_3_y, a_16_3_z, hPa_15, hPa_17, hPa_18

For each label, relevant features (i.e., data streams) are selected by a domain expert under consideration of the semantic knowledge graph and a visual inspection of the data streams. In the end, the relevant features are divided into two categories. The features referred to as *direct* contain directly observable patterns, while the *contextual* features are required to understand in which context the directly observable pattern appears. Features starting with *a* are accelerating sensors, containing an *i* stands for light barriers and position switches, a *m* stands for a motor, *o* for valves and compressors, and *hPa* indicates a pressure sensor.

Table 12
Overview of SPARQL query patterns used to find pairs (n_q, n_y) between data streams for building an adjacency matrix.

No	SPARQL Query Pattern	Query	Rel. Type	Pairs
1	?q sosa:hosts ?y .	direct	Compositional	24
2	?q FTOnto:isConnectedViaPipeTo ?y .	direct	Associational	16
3	?x sosa:hosts ?q . ?x sosa:hosts ?q .	indirect	Compositional	516
4	?x FTOnto:hasInput/Output ?q . ?x FTOnto:hasInput/Output ?y .	indirect	Associational	190
5	?q ssn:forProperty ?x . ?x ssn:isPropertyOf ?y .	indirect	Associational	120
6	?q ssn:forProperty ?x . ?x ssn:isPropertyOf ?z . ?z sosa:hosts ?y .	indirect	Associational	18
7	?q ssn:forProperty ?x . ?y ssn:forProperty ?x .	indirect	Associational	130
8	?q ssn:forProperty ?x . ?x FTOnto:usedToControl ?y .	indirect	Associational	44
9	?q ssn:forProperty ?x . ?x FTOnto:usedToControl ?z . ?z sosa:hosts ?y .	indirect	Associational	80
10	?x PredM:not_in_same_state_at_the_same_time ?z . ?q ?r ?x . ?y ?r2 ?z .	indirect	Associational	20
11	?x PredM:DetectableInDS_Cont ?q . ?x PredM:DetectableInDS_Cont ?y	indirect	Associational	192

The previous notation n_x and $?nx$ means node x and is shortened by removing the n.

Table 13
Detailed overview of results for finding the root cause (i.e., Label) of an anomaly with the semantic knowledge graph.

		MSCRED	MSCRED AM	Siam CNN1D FC	Siam CNN2D GCN+GSL	Oracle (Masking)
SPARQL Q1-L	Queries executed	3,396 ± 640	2,493 ± 474	1,361 ± 545	1,285 ± 230	482 ± 0.0
	Labels provided	2,058 ± 494	1,673 ± 455	1,585 ± 728	1,631 ± 345	1,170 ± 23
	Correct Labels	180 ± 43	168 ± 66	112 ± 74	184 ± 35	482 ± 0.0
	QR / LR / Cov.	**18.9 | 11.4 | 0.7**	**14.8 | 10 | 0.7**	**12.2 | 14.2 | 0.6**	**7 | 8.9 | 0.6**	**1 | 2.4 | 1**
SPARQL Q1-L + Con.	Queries executed	3,410 ± 619	2,536 ± 422	1,104 ± 481	1,286 ± 230	484 ± 0.0
	Labels provided	1,113 ± 261	936 ± 285	868 ± 559	744 ± 191	1151 ± 22
	Correct Labels	179 ± 44	167 ± 66	112 ± 74	183 ± 36	480 ± 0.0
	QR / LR / Cov.	**19.1 | 6.2 | 0.7**	**15.2 | 5.6 | 0.7**	**9.9 | 7.8 | 0.6**	**7 | 4.1 | 0.6**	**1 | 2.4 | 1**
SDNR Q1-L + Con.	Queries executed	2,876 ± 532	2,056 ± 430	1,297 ± 470	1,233 ± 198	484 ± 0.0
	Labels provided	1779 ± 402	1426 ± 444	870 ± 509	1,057 ± 283	1270 ± 0.0
	Correct Labels	206 ± 46	184 ± 59	118 ± 81	197 ± 47	480 ± 0.0
	QR / LR / Cov.	**14 | 8.6 | 0.8**	**11.2 | 7.8 | 0.8**	**11 | 7.4 | 0.7**	**6.3 | 5.4 | 0.6**	**1 | 2.6 | 1**
SPARQL Q2-L	Queries executed	40,635 ± 6,542	30,888 ± 6,383	11,710 ± 3,655	11,846 ± 1,513	482 ± 0.0
	Labels provided	2,533 ± 492	1,972 ± 540	1,843 ± 771	2,543 ± 430	1,138 ± 33
	Correct Labels	127 ± 37	126 ± 59	105 ± 78	204 ± 56	482 ± 0.0
	QR / LR / Cov.	**320 | 19.9 | 0.5**	**245.1 | 15.7 | 0.6**	**111.5 | 17.6 | 0.6**	**58.1 | 12.5 | 0.6**	**1 | 2.4 | 1**
SPARQL Q3-L	Queries executed	1,893 ± 600	1,413 ± 356	543 ± 233	834 ± 131	536 ± 0.0
	Labels provided	4,998 ± 793	3,979 ± 1031	2,832 ± 1459	5,171 ± 663	2,631 ± 44.0
	Correct Labels	248 ± 51	214 ± 77	164 ± 92	319 ± 39	482 ± 0.0
	QR / LR / Cov.	**7.6 | 20.2 | 0.9**	**6.6 | 18.6 | 0.9**	**3.3 | 17.3 | 0.9**	**2.6 | 16.2 | 1**	**1.1 | 5.5 | 1**
CBR Q1-L	Queries executed	3,020 ± 573	2,074 ± 323	1,043 ± 475	1,144 ± 239	482 ± 0.0
	Labels provided	3,271 ± 702	2,531 ± 641	1578 ± 722	1,877 ± 409	1,153 ± 19
	Correct Labels	201 ± 46	188 ± 69	129 ± 78	199 ± 36	482 ± 0.0
	QR / LR / Cov.	**16.3 | 15.0 | 0.8**	**11.0 | 13.5 | 0.8**	**8.0 | 12.2 | 0.7**	**5.7 | 9.4 | 0.6**	**1 | 2.4 | 1**
CBR Q1-L + Con.	Queries executed	3,063 ± 585	2,112 ± 329	1,080 ± 489	1,160 ± 233	484 ± 0.0
	Labels provided	2,326 ± 468	1,766 ± 444	901 ± 559	1,199 ± 278	1,150 ± 17
	Correct Labels	199 ± 45	187 ± 69	125 ± 77	196 ± 37	480 ± 0.0
	QR / LR / Cov.	**15.4 | 11.7 | 0.8**	**11.3 | 9.4 | 0.8**	**8.6 | 7.2 | 0.7**	**5.9 | 6.1 | 0.6**	**1 | 2.4 | 1**
CBR Q2-L	Queries executed	38,669 ± 7006	25,869 ± 5972	9,169 ± 3,191	8,520 ± 1681	482 ± 0.0
	Labels provided	2,798 ± 638	2,173 ± 639	1,886 ± 800	2,524 ± 436	1,135 ± 19
	Correct Labels	125 ± 40	137 ± 65	115 ± 80	221 ± 57	482 ± 0.0
	QR / LR / Cov.	**309 | 31.2 | 0.5**	**189 | 15.9 | 0.6**	**79.7 | 16.4 | 0.7**	**38.6 | 11.4 | 0.7**	**1 | 2.3 | 1**

Appendix F. Ablation study for siam AD model

An ablation study is conducted on architectural/design changes in the proposed Siam models for anomaly detection of Section 3.2, as shown in Table 16. The reported scores are the mean of five consecutive runs on the test set. These results indicate that the prediction head (cf. Section 3.2.1) only improves the performance of Siam CNN-1D FC in terms of AvgPR and PR AUC metrics. Otherwise, a normal Siamese neural network that maximizes the negative Cosine similarity achieves comparable results in this setting. Removing the graph structure learning module from Siam CNN-2D GCN + GSL leads to a significant performance drop and highlighting its importance. For this reason, the following section investigates different aspects of it in detail.

Appendix G. Corrupted semantic models

See Listings 2 and 3.

Appendix H. Details of applied anomaly detection approaches

In the following, details how the anomaly detection approaches are applied are given so that results are reproducible.

H.1. One class support vector machine

The input of the OC-SVM is flattened to a 1-dimensional vector and the features are standardized by subtracting the mean and scaling to unit variance to obtain normally distributed data (i.e., Gaussian with zero mean and unit variance). Unless otherwise specified, OC-SVM is used with its default hyperparameters, as implemented in sklearn (Pedregosa et al., 2011).

H.2. Unsupervised Siamese networks

As outlined in Section 3.2, a Siamese network with two distinct encoder structures is trained on normal data as part of an approach

Table 14
Detailed overview of results for finding the affected component of an anomaly with the semantic knowledge graph.

	MSCRED	MSCRED AM	Siam CNN1D FC	Siam CNN2D GCN+GSL	Oracle (Masking)
SPARQL Q1-C	2,704 ± 546	2,513 ± 609	1,257 ± 477	1,201 ± 253	482 ± 0
	1,250 ± 246	1,459 ± 239	884 ± 335	849 ± 229	662 ± 11
	157 ± 69	167 ± 46	117 ± 78	190 ± 36	482 ± 0
	17.2 | 8 | 0.6	**15 | 8.7 | 0.7**	**10.7 | 7.6 | 0.7**	**6.3 | 4.5 | 0.6**	**1 | 1.4 | 1**
SPARQL Q1-C + Con.	3,554 ± 614	2,725 ± 561	1,350 ± 532	1,294 ± 230	484 ± 0
	644 ± 151	578 ± 189	342 ± 205	415 ± 94	652 ± 9
	162 ± 46	153 ± 70	113 ± 75	181 ± 38	480 ± 0
	21.9 | 4 | 0.6	**17.8 | 3.8 | 0.7**	**11.9 | 3 | 0.6**	**7.1 | 2.3 | 0.6**	**1 | 1.4 | 1**
SDNR Q1-C + Con.	3,230 ± 566	1,902 ± 451	1,126 ± 456	1,201 ± 236	484 ± 0
	735 ± 189	775 ± 190	457 ± 261	536 ± 121	513 ± 2
	192 ± 45	197 ± 64	126 ± 78	191 ± 37	480 ± 0
	16.8 | 3.8 | 0.7	**9.7 | 3.9 | 0.9**	**8.9 | 3.6 | 0.7**	**6.3 | 2.8 | 0.6**	**1 | 1.1 | 1**
CBR Q1-C	3,015 ± 571	2,073 ± 322	1,040 ± 473	1,143 ± 239	482 ± 0
	4,019 ± 934	2,797 ± 317	1,414 ± 633	1295 ± 292	595 ± 8
	201 ± 46	188 ± 69	129 ± 78	199 ± 35	482 ± 0
	15 | 20 | 0.8	**11 | 14.9 | 0.8**	**8.1 | 11 | 0.7**	**5.7 | 6.5 | 0.6**	**1 | 1.2 | 1**
CBR Q1-C + Con.	3,060 ± 582	2,111 ± 328	1,073 ± 482	1,159 ± 234	484 ± 0
	2,232 ± 398	1,595 ± 292	709 ± 167	826 ± 159	596 ± 9
	199 ± 45	187 ± 69	125 ± 77	197 ± 36	480 ± 0
	15.4 | 11.2 | 0.8	**11.3 | 8.5 | 0.8**	**8.6 | 5.7 | 0.7**	**5.9 | 4.2 | 0.6**	**1 | 1.2 | 1**

Listing 2: Removed triples from the semantic model to make it more difficult to find the right label.

```
FM_txt15_m1_t1 isDetectInDS_Direct ADXL345_3_x–Axis
FM_txt15_m1_t1 isDetectInDS_Context ADXL345_3_x–Axis
FM_txt15_m1_t2 isDetectInDS_Direct ADXL345_3_y–Axis
FM_txt15_m1_t2 isDetectInDS_Context ADXL345_3_y–Axis
FM_txt16_m3_t1 isDetectInDS_Direct ADXL345_4_z–Axis
FM_txt16_m3_t1 isDetectInDS_Context ADXL345_4_z–Axis
FM_txt16_m3_t2 isDetectInDS_Direct ADXL345_4_x–Axis
FM_txt16_m3_t2 isDetectInDS_Context ADXL345_4_x–Axis
FM_txt16_m3_t2 isDetectInDS_Direct ADXL345_4_z–Axis
FM_txt16_m3_t2 isDetectInDS_Context ADXL345_4_z–Axis
FM_txt16_conveyor_fm_driveshaft_slippage isDetectInDS_Direct ADXL345_4_z–Axis
FM_txt16_conveyor_fm_driveshaft_slippage isDetectInDS_Context ADXL345_4_z–Axis
FM_txt16_i4_lightbarrier_fm_1 isIndicatedBy Symp_FalsePositiveSignalsIntermittent
FM_txt16_con_big_gear_tooth_broken hasLabel FM_txt16_con_big_gear_tooth_broken
SM_Motor_1 hasFunction Func_SM_M1_Drive_Conveyor_Belt
FM_txt15_i1_lightbarrier_fm_2 hasLabel Label_txt15_i1_lightbarrier_fm_2
FM_txt15_i3_lightbarrier_fm_2 isDetectInDS_Direct SM_Light_Barrier_3
FM_txt15_i3_lightbarrier_fm_2 isDetectInDS_Context SM_Light_Barrier_3
SM_Light_Barrier_3 hasPotentialFailureMode FM_txt15_i3_lightbarrier_fm_2
SM_Light_Barrier_1 hasPotentialFailureMode FM_txt15_i1_lightbarrier_fm_1
FM_txt15_i1_lightbarrier_fm_1 isDetectInDS_Direct SM_Light_Barrier_1
FM_txt15_i1_lightbarrier_fm_1 isDetectInDS_Context SM_Light_Barrier_1
VGR_Compressor_7 hasPotentialFailureMode FM_txt18_pneumatic_leakage_fm_2
FM_txt18_pneumatic_leakage_fm_2 isDetectInDS_Direct VGR_Comp_7_SSCMRRN03PD2A3
FM_txt18_pneumatic_leakage_fm_2 isDetectInDS_Context VGR_Comp_7_SSCMRRN03PD2A3
MPS_Motor_3 hasPotentialFailureMode FM_txt16_m3_t2
FM_txt15_m1_t1 hasLabel Label_txt15_m1_t1_high_wear
SM_Motor_1 hasPotentialFailureMode FM_txt15_m1_t2
```

for data-driven anomaly detection. Further details can be found in Section 2.2.

The *Siam 1D CNN FC* model comprises three convolutional layers with 256, 64, and 32 filters, each with a kernel length of 5, 5, and

Table 15
Detailed overview of results for finding the root cause (i.e., Label or Affected Component) of an anomaly with a corrupted knowledge graph.

			KG Corrupted I		KG Corrupted II	
			Siam CNN2D GCN+GSL	**Oracle** (Masking)	**Siam CNN2D GCN+GSL**	**Oracle** (Masking)
Label	**SPARQL Q1-L**	Queries executed	1,648 ± 333	759 ± 0.0	1,854 ± 369	1,064 ± 0.0
		Labels provided	1,483 ± 341	1,126 ± 13	1,141 ± 258	680 ± 4
		Correct Labels	116 ± 21	342 ± 0.0	68 ± 10	141 ± 0.0
		QR / LR / Cov.	14.2 | 12.8 | 0.4	2.2 | 3.3 | 0.7	27.3 | 16.8 | 0.2	7.5 | 4.8 | 0.3
		Adj. QR / LR/ Cov	**21.6 | 13.3 | 0.7**	**10.3 | 6.4 | 1**	**29.4 | 14.9 | 0.6**	**23.7 | 11.3 | 1**
	SPARQL Q1-L + Con.	Queries executed	1,916 ± 334	759 ± 0.0	2,122 ± 376	1,064 ± 0.0
		Labels provided	615 ± 150	1,055 ± 11	438 ± 110	610 ± 5
		Correct Labels	79 ± 20	300 ± 0.0	31 ± 10	99 ± 0.0
		QR / LR / Cov.	24.3 | 7.8 | 0.2	2.5 | 3.5 | 0.6	68.5 | 14.1 | 0.1	10.7 | 6.2 | 0.2
		Adj. QR / LR/ Cov	**28.2 | 11.7 | 0.7**	**13 | 7.4 | 1**	**36.3 | 14 | 0.6**	**26.4 | 12.4 | 1**
	SDNR Q1-L + Con.	Queries executed	1,097 ± 198	484 ± 0.0	1,185 ± 253	563 ± 0.0
		Labels provided	1,544 ± 333	2,060 ± 0.0	1,792 ± 412	2,486 ± 0.0
		Correct Labels	212 ± 45	480 ± 0.0	189 ± 32	480 ± 0.0
		QR / LR / Cov.	5.2 | 7.3 | 0.7	1 | 4.3 | 1	6.3 | 9.5 | 0.6	1.2 | 5.2 | 1
		Adj. QR / LR/ Cov	**5.2 | 7.3 | 0.7**	**1 | 4.3 | 1**	**7.9 | 9.8 | 0.6**	**1.2 | 5.2 | 1**
	CBR Q1-L + Con.	Queries executed	1,153 ± 226	484 ± 0.0	1,104 ± 259	484 ± 0.0
		Labels provided	1,236 ± 279	1,147 ± 13.5	1,393 ± 309	1,149 ± 11.1
		Correct Labels	198 ± 37	480 ± 0.0	203 ± 36	480 ± 0.0
		QR / LR / Cov.	5.8 | 6.2 | 0.6	1 | 2.4 | 1	5.4 | 6.9 | 0.6	1 | 2.4 | 1
		Adj. QR / LR/ Cov	**7.5 | 6.8 | 0.7**	**1 | 2.4 | 1**	**5.4 | 6.9 | 0.6**	**1 | 2.4 | 1**
Component	**SPARQL Q1-C**	Queries executed	1,721 ± 384	961 ± 0.0	1,777 ± 393	1,040 ± 0.0
		Labels provided	434 ± 111	413 ± 4	326 ± 70	203 ± 2
		Correct Labels	94 ± 18	197 ± 0.0	79 ± 13	109 ± 0.0
		QR / LR / Cov.	18.3 | 4.6 | 0.3	4.9 | 2.1 | 0.4	22.5 | 4.1 | 0.2	9.5 | 1.9 | 0.2
		Adj. QR / LR/ Cov	**24.7 | 5.9 | 0.6**	**20 | 5 | 1**	**27.4 | 5.9 | 0.6**	**25.7 | 5.8 | 1**
	SPARQL Q1-C + Con.	Queries executed	2,059 ± 362	961 ± 0.0	2,116 ± 374	1,040 ± 0.0
		Labels provided	152 ± 41	366 ± 3	106 ± 21	157 ± 5
		Correct Labels	50 ± 18	155 ± 0.0	34 ± 13	67 ± 0.0
		QR / LR / Cov.	41.2 | 3 | 0.2	6.2 | 2.4 | 0.3	62.2 | 3.1 | 0.1	15.5 | 2.3 | 0.1
		Adj. QR / LR/ Cov	**33.2 | 6.0 | 0.6**	**22.7 | 5.5 | 1**	**35.8 | 6.3 | 0.6**	**28.4 | 6.4 | 1**
	SDNR Q1-C + Con.	Queries executed	1,181 ± 225	484 ± 0.0	1,201 ± 236	484 ± 0.0
		Labels provided	440 ± 100	579 ± 2.0	535 ± 121	513 ± 2.0
		Correct Labels	196 ± 39	480 ± 0.0	191 ± 37	480 ± 0.0
		QR / LR / Cov.	6 | 2.2 | 0.6	1 | 1.2 | 1	6.3 | 2.8 | 0.6	1 | 1.1 | 1
		Adj. QR / LR/ Cov	**6.3 | 2.3 | 0.6**	**1 | 1.2 | 1**	**7.2 | 3.0 | 0.6**	**1 | 1.1 | 1**
	CBR Q1-C + Con.	Queries executed	1,153 ± 226	484 ± 0.0	1,105 ± 259	484 ± 0.0
		Labels provided	841 ± 154	594 ± 9	981 ± 200	594 ± 9
		Correct Labels	198 ± 37	480 ± 0	203 ± 36	480 ± 0
		QR / LR / Cov.	5.8 | 4.2 | 0.6	1 | 1.2 | 1	5.4 | 4.8 | 0.6	1 | 1.2 | 1
		Adj. QR / LR/ Cov	**5.8 | 4.2 | 0.6**	**1 | 1.2 | 1**	**5.4 | 4.8 | 0.6**	**1 | 1.2 | 1**

3 and strides of 2, 2, and 1 respectively. Following each convolution operation is the application of the ReLU activation function and batch normalization. Subsequently, a drop-out layer with a rate of 0.05 is applied before three Fully Connected (FC) layers with 512, 256, and 128 units are implemented. Batch normalization is applied to each FC layer followed by the ReLU activation function. The projector head (cf. Section 2.2) consists of two fully connected layers that are followed by ReLU and batch normalization as well as a final fully connected layer with 128, 32, and 128 units respectively. Drop-out regularization with a rate of 0.2 is applied to the projector head's input.

The *Siam 2D CNN GCN + GSL* model comprises three blocks of 2D convolutions with 128, 64, and 3 kernels of size (5,1), (5,1), and (3,1) respectively. Each convolution is followed by batch normalization and a ReLU activation function. The output of the last convolutional layer is then concatenated with the Knowledge Graph embeddings of each data stream. Batch normalization and fully connected layers with 384 and 256 units are applied data-stream wise. This is followed by three layers of graph convolutions with a kernel size of 128, batch normalization and the ReLU activation function. The input for each graph convolutional is concatenated with the corresponding Knowledge Graph embeddings, as well as the adjacency matrix derived from the semantic Knowledge Graph. Global attention pooling with 128 units is applied to obtain a vector representation. The projector head consists of two fully connected layers that are followed by ReLU and batch

Listing 3: In addition to the triples of Listing 2, the following ones are also removed from the semantic model to make it more difficult to find the right label.

```
FM_txt15_m1_t1  hasLabel  Label_txt15_m1_t1_low_wear
FM_txt15_m1_t2  isDetectInDS_Direct  AccSensor_ADXL345_3_x—Axis
FM_txt15_m1_t2  isDetectInDS_Context  AccSensor_ADXL345_3_x—Axis
FM_txt15_m1_t2  isDetectInDS_Direct  AccSensor_ADXL345_3_z—Axis
FM_txt15_m1_t2  isDetectInDS_Context  AccSensor_ADXL345_3_z—Axis
FM_txt16_conveyor_fm_driveshaft_slippage  isDetectInDS_Direct  ADXL345_4_x—Axis
FM_txt16_conveyor_fm_driveshaft_slippage  isDetectInDS_Context  ADXL345_4_x—Axis
FM_txt16_conveyor_fm_driveshaft_slippage  isDetectInDS_Direct  ADXL345_4_y—Axis
FM_txt16_conveyor_fm_driveshaft_slippage  isDetectInDS_Context  ADXL345_4_y—Axis
FM_txt16_m3_t2  isDetectInDS_Direct  AccSensor_ADXL345_4_y—Axis
FM_txt16_m3_t2  isDetectInDS_Context  AccSensor_ADXL345_4_y—Axis
FM_txt16_m3_t1  hasLabel  Label_txt16_m3_t1_high_wear
FM_txt18_pneumatic_leakage_fm_1  hasLabel  Label_txt18_pneumatic_leakage_fm_1
```

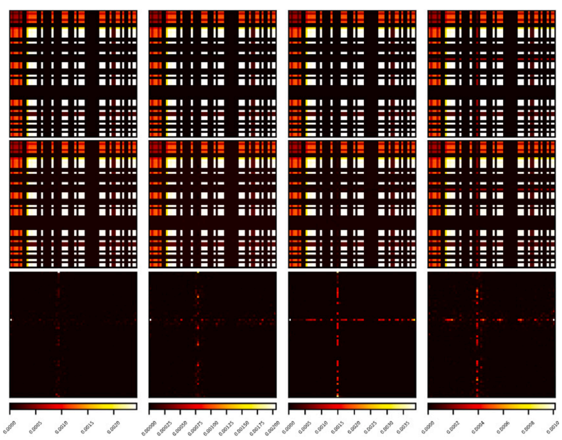

Fig. 14. Example of a reconstruction error-based explanation of the test example with index 3385 of used data set which is correctly detected as true positive during the anomaly detection by MSCRED. The first row is the input of the four signature matrices, the second row their reconstruction and the last row the reconstruction error. The explanation highlights the reconstruction error observed in the correlations of one data stream, namely txt16_i4 where the light barrier observes an anomalous signal.

Table 16
Ablation study Siam CNN1D FC and Siam CNN2D GCN.

Algo	Test		
	Roc Auc	Avg PR	PR Auc
Siam CNN1D FC	75.93 ± 9.70	39.63 ± 7.15	39.54 ± 9.70
w/o Prediction Head	74.66 ± 7.09	32.73 ± 7.98	32.62 ± 7.95
kMean instead kNN	62.07 ± 16.42	31.35 ± 18.21	31.42 ± 18.23
Siam CNN2D GCN+GSL	**92.78 ± 1.23**	**72.89 ± 2.93**	**72.98 ± 1.23**
Siam CNN2D GCN (wo GSL)	87.46 ± 1.89	58.48 ± 3.88	58.22 ± 3.89
w/o Prediction Head	86.13 ± 1.82	57.74 ± 4.21	58.55 ± 4.13
w. Gradient Stop	85.89 ± 2.24	55.03 ± 3.27	54.97 ± 3.29
kMean instead kNN	80.78 ± 2.70	43.00 ± 4.41	42.92 ± 4.41

normalization as a final fully connected layer, with units of 128, 4, and 128 respectively. Drop-out regularization with a rate of 0.2 is applied to the projector head's input. The Graph Structure Learning (GSL) module uses an equation corresponding to an undirected graph, with masking of the predefined adjacency matrix and a reduction on the outgoing edges of any node to a maximum of 5 (k-NN). Post-processing adds a self-loop and applies non-symmetric normalization to $\hat{A} = \tilde{D}^{-1}\tilde{A}$in case of an asymmetric adjacency matrix that corresponds to a directed

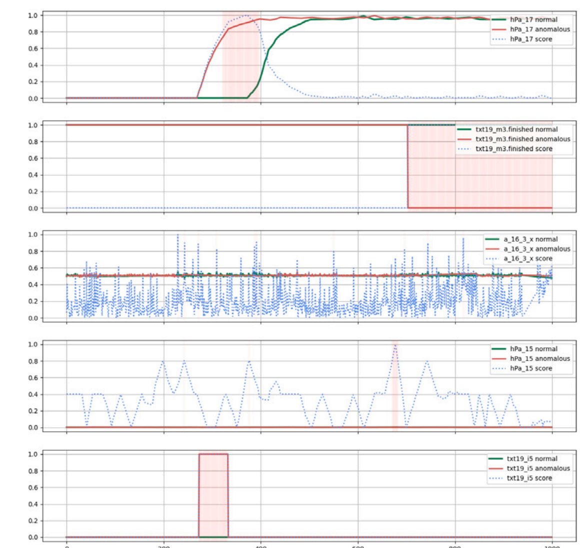

Fig. 15. Example of a counterfactual explanation of the test example with index 7 of the used data set (red lines) employing the healthy training example with index 13,226 (green lines) which is detected as false positive during the anomaly detection. An expert can inspect the data streams and would recognize that only the second and last one are from the same workstation. The pressure increase measured by an air pressure sensor in the first data stream is earlier as in the example, but has the same shape. The third and fourth data stream only show minor divergences to the healthy one and can theoretically be rejected, whereas for the acceleration data, statistical features such as Kurtosis would be more appropriate for detecting faults. Finally, the expert has to decide if the pattern of the light barrier disruption of $txt19\_i5$ and the motor activity of $txt19\_m3$ has any anomalous signs. If not, the expert can reject the predicted anomaly and verify the healthy condition as it is labeled.

graph.[9]

The gradient stop was removed from both reported models, resulting in an increase in performance (cf. Appendix F). The training was conducted with a batch size of 64 pairs, a learning rate of 0.0001 and the Adam optimizer. Early stopping was employed after no improvement of the training loss on 100 batches of the training data, and the model with the lowest training loss was used for evaluation. The hyperparameters were tuned manually against the validation split. Both architectures correspond to the best performing models known to the author. Hyperparameters of the encoder (excluding the head) are initialized from previous (Klein et al., 2021) as a starting point, with grid search applied for fine-tuning. For further

[9] https://github.com/tkipf/gcn/issues/91#issuecomment-469181790, last accessed on 03/05/2020.

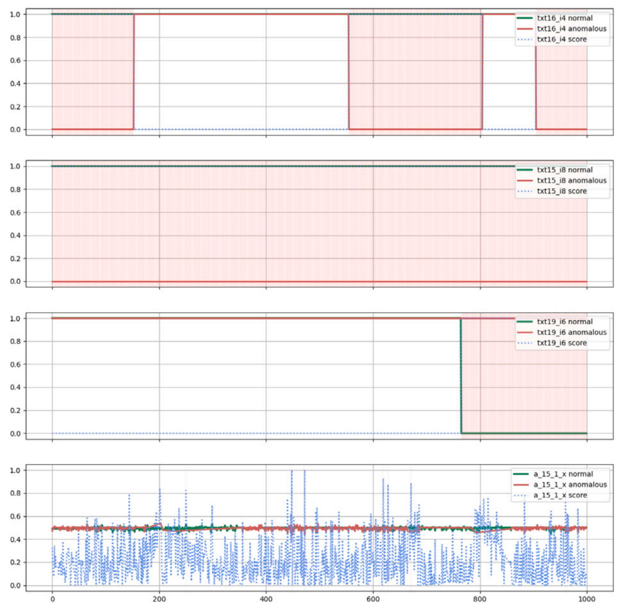

Fig. 16. Example of a counterfactual explanation of the test example with index 3386 of the used data set (red line) utilizing the healthy training example with index 18,579 (green line). The explanation presents the anomalous data stream txt16_i4 at the top and hence most relevant as cause. For an expert, the anomaly is recognizable through the multiple changes of the signal compared to the normal expected signal in this situation. The further deviations in data stream txt15_i8, txt19_i6, and a_15_1_x are not relevant for the anomaly and an expert has to exclude them as causes.

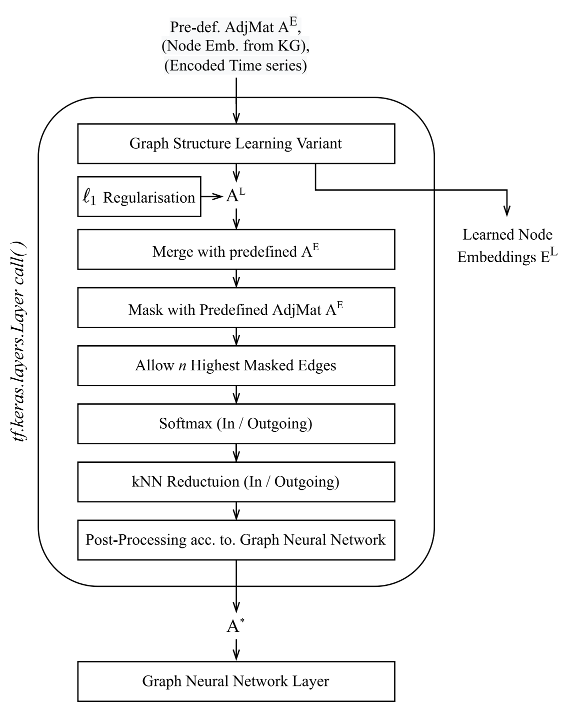

Fig. 17. Schematic depiction of the operations conducted by the graph structure learning module.

details, please refer to the implementation: https://github.com/PredM/SimSiamDistanceBasedAnomaly.

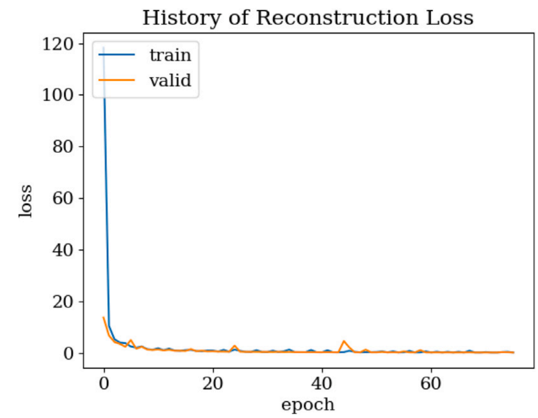

Fig. 18. Training and validation loss of the third MSCRED model which starts with 118.31719970703125 and 13.658352851867676 and is reduced to 0.07308381795883179 and 0.07253975421190262 at epoch 72, which is the state of the model that is selected for the evaluation in this case.

H.3. Implementation of the graph structure learning component used in siam CNN2D GCN + GSL

For investigating approaches to learning dependencies between data streams of an underlying system and integrating expert knowledge about them effectively (cf. Section 3.2.2), a graph structure learning (Zhu et al., 2021) component is implemented[10] as a custom layer of the deep learning framework Keras (Chollet et al., 2015). For processing a sample (i.e., invoke of the *call()*-function), the sequence of operations as depicted in Fig. 17 is executed based on the given parameters. The input of the graph structure learning module consists of a predefined adjacency matrix A^E, previously learned node embeddings corresponding to each data stream, or the node features (i.e., time series data) itself. The graph structure learning module is implemented with the following variants are presented in Section 2.4 in Eqs. (5)–(10).

H.4. Application of MSCRED

The MSCRED framework (Zhang et al., 2019a) is an example of a deep learning reconstruction-based anomaly detection approach.

Preparation of input signature matrices

The data set's example (i.e., a 4-s window of time series) is divided into four signature matrices, as shown in Fig. 19, with a step size of 1 s or 250 entries and $\mathcal{W} = \{63, 125, 186, 250\}$. Thus, each example corresponds to one signature matrix example and makes the results comparable to other AD approaches that work on raw time series data.

Parameterization of MSCRED and learning procedure

The hyperparameters of the MSCRED model remain unchanged in the original proposed model (Zhang et al., 2019a) because no parameters are found that consistently improved the reconstruction error. The training process involves mini-batches of 128 examples, and early stopping is employed when the reconstruction error measured on a hold-out-split of the failure-free training data does not decrease any further. The training is terminated after 100 epochs or if there are no further enhancements after three consecutive epochs. Adam optimizer with a learning rate of 0.001 and batch size of 128 was selected as it yielded the best results on the validation set. A plot of the training and validation loss can be found in Fig. 18.

[10] https://github.com/PredM/SimSiamDistanceBasedAnomaly/blob/c9dbe9b380b064aa1b27cebdd173f3725500267a/neural_network/BasicNeuralNetworks.py#L1073C59-L1073C59.

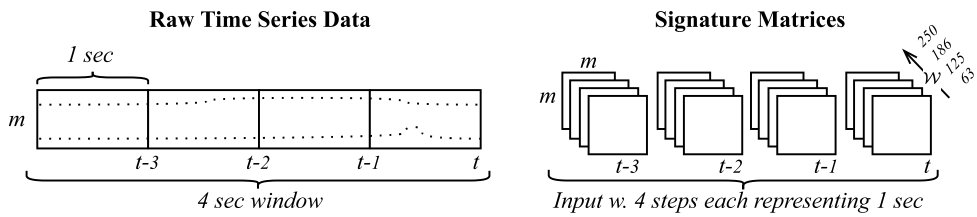

Fig. 19. Conversion of a time series window with m data streams to a representation with correlation matrices. Parameters are as used in during the evaluation. The $\mathcal{W}$ in the upper-right corner indicates the different lengths for computing the correlation, which are defined as 63, 125, 186 and 250 time steps and 250 corresponding to 1 s.

Definition of anomaly detection threshold

In Zhang et al. (2019a), the method for determining the appropriate threshold θ for defining the anomaly score $s(t)$ at a time point t for a single correlation between two data streams is not specified in detail. Consequently, the threshold is empirically determined by selecting the value that yields the highest ROC AUC score on the validation data set. The threshold τ for determining whether $s(t)$ is normal or abnormal is defined by maximizing the f1-score on the validation data set, as per the paper (Zhang et al., 2019a).

MSCRED implementation

The MSCRED framework (Zhang et al., 2019a) is re-implemented using TensorFlow and Keras, as the implementation provided in the paper is not executable.[11] This reconstruction-based anomaly detection approach is selected due to its superior performance compared to other baseline approaches on the authors' data, as well as its innovative encoder architecture specifically designed for time series data of technical systems and its ability to identify causative data streams. For further information, please refer to the repository of the implementation.[12]

Appendix I. SPARQL query strategies

The implementation of SPARQL Q3-L is shown in Listing 4. This query makes use of the compositional structure by providing all labels that are associated with data streams belonging to the same higher-level component (cf. Table 10). The SPARQL query SPARQL Q1-C is presented in Listing 5. The constraint from Section 3.4.2 is incorporated into query Q1-L, as illustrated in Listing 6. This query includes two variables, *Func_constraint_part* and *Symp_constraint_part*, which are used to impose function-based (i.e., component activity) and symptom-based constraints, respectively, based on the anomalous data stream *ds_name*.

Listing 4: SPARQL query implementation for SPARQL Q2-L which retrieves possible labels for an anomalous data stream (here: txt16_i4) by making use of the component structure modeled in the Knowledge Graph through the properties *isHostedBy* and *hasComponent*. The prefixes are similar to those defined in Listing 1.

```
SELECT  DISTINCT ?labels
WHERE {
 {
   ?component ftonto:is_associated_with_data_stream  "txt16_i4"^^xsd:string.
   ?workstation  ftonto:hasComponent ?component.
 }UNION{
   ?sensorStream  ftonto:is_associated_with_data_stream  "txt16_i4"^^xsd:string.
   ?sensor  ftonto:hasComponent ?sensorStream  .
   ?sensor  sosa:isHostedBy ?component  .
   ?workstation  ftonto:hasComponent ?component  .
 }{
   {
     ?workstation  ftonto:hasComponent ?components.
     ?failureModes predm:isDetectableInDataStreamOf_Context ?components.
   }UNION{
     ?failureModes predm:isDetectableInDataStreamOf_Context  ?component.
   }
 }{
   ?failureModes predm:hasLabel ?labels
 }
}
```

Listing 5: The implementation of query strategy Q1-C is used to obtain the components affected by the anomalous data stream (txt16_i4). The prefixes used in this query are similar to those defined in Listing 1. The graph pattern modeled in this query is presented in Fig. 7

```
SELECT DISTINCT ?items
WHERE {
    ?component  ftonto:is_associated_with_data_stream
           "txt16_i4"^^xsd:string.
    ?failureModes  predm:isDetectableInDataStreamOf_Direct
           ?component.
    ?items  fmeca:hasPotentialFailureMode  ?failureModes.
}
```

Listing 6: A SPARQL query (Listing 1) is used to obtain labels based on a given anomalous data (*ds_name*), which correspond to Q1-L. The variable *Symp_constraint_part* contains a constraint based on symptoms and *Func_constraint_part* contains a constraint for an active function, both dependent on the processed data stream *ds_name*. The prefixes are similar to those defined in Listing 1.

```
SELECT   DISTINCT ?labels
WHERE {
  {
    ?component  ftonto:is_associated_with_data_stream
           "ds_name"^^xsd:string.
    ?component  fmeca:hasPotentialFailureMode  ?failureModes.
    ?failureModes  predm:hasLabel  ?labels.
    Func_constraint_part.
    Symp_constraint_part.
  }
  UNION{
    ?component  ftonto:is_associated_with_data_stream
           "ds_name"^^xsd:string.
    ?failureModes  fmeca:isDetectableInDataStreamOf_Direct
           ?component.
    ?failureModes  predm:hasLabel  ?labels.
    Func_constraint_part.
    Symp_constraint_part.
  }
}
```

I.0.1. Implemented constraint in the SPARQL queries

The constraint from Section 3.4.2 is incorporated into query Q1-L, as illustrated in Listing 6. This query includes two variables, *Func_ constraint_part* and *Symp_constraint_-part*, which are used to impose function-based (i.e., component activity) and symptom-based constraints, respectively, based on the anomalous data stream *ds_name*. The constraint is applied as follows: If the mean of the data stream *txt15_m1.finished* is less than 0.2 during a time series window, then the functions *PredM#Func_SM_M1_Drive_Con-veyor_Belt* and *PredM#Func_ SM_CB_ transport_workpieces* are considered as active; otherwise, they are inactive. The same applies to another conveyor belt monitored by data streams *a_16_3x*, *a16_3_y*, and *a16_3_z* with actuator *txt16-m3.finished* and functions *PredM#FuncSMM1DriveConvey-orBelt* and *PredM#Func SMCBtransportworkpieces*. If an ano-maly is detected in a pressure sensor data streams (e.g., *hPa15*, *hPa17*, and *hPa18*) or any of the valve or compressor data streams, then functions *PredM#FuncSMP neu-maticSystemProvidePressure*, *PredM#FuncMPSBFPneu-maticSystem ProvidePressure*, and *PredM#FuncVGRPneu-maticSystemProvidePressure* are considered as active only if the compressor's mean value is higher

[11] https://github.com/7fantasysz/MSCRED/issues, last accessed 04/02/2022

[12] https://github.com/PredM/MSCRED-Deep-Autoencoder-Anomaly-Detection, last accessed on 08/08/2023

than 0.8; otherwise, they are inactive. This suggests that anomalies related to the pneumatic system can only be detected when it is active and pressure has been generated. The symptom *Symp_constraint_part* is generated by considering only the light barriers and position switches. The symptom is applied as follows: If the absolute sum of changes[13] is equal to or less than 1, then *PredM#Symp_ContinuousSignal* is observed; otherwise, *PredM#Symp_IntermittentSignal* is observed due to the presence of at least two changes in the signal.

If the function *PredM#Func_SM_M1_Drive_Conveyor_-Belt* is active during the time series window, then the *Func_con-straint_part* of Listing 6 should include the part shown in Listing 7. Conversely, if it is not active, it should be negated by *FILTER NOT EXISTS {Func_constraint_-part}*. If the symptom *PredM#Symp_FalsePositiveSignal-Continuous* is detected in the input data, Listing 8 should be included in Listing *Func_constraint_part*; however, if it is not identified by the simple feature, it should not be negated.

Listing 7: Example for *Func_constraint_part* of Listing 6

```
{
  predm:Func_SM_M1_Drive_Conveyor_Belt  fmeca:definesFailureMode
        ?failureModes.
} UNION {
  predm:Func_SM_CB_transport_workpieces  fmeca:definesFailureMode
        ?failureModes.
}
```

Listing 8: Example for *Symp_constraint_part* of Listing 6

```
?failureModes  fmeca:isIndicatedBy
      predm:Symp_FalsePositiveSignalContinuous.
```

Appendix J. Formulas for anomalous data stream identification measures

For the diagnosis of an anomaly, a ranked list $c \in \mathbb{R}^m$ is typically provided by the AD model (e.g., Su et al., 2019b; Zhao et al., 2020b; Zhang et al., 2019a). The most relevant data stream is given by c_i where $i = 0$. To evaluate the quality of the information contained in r, two metrics, namely $HitRate\%@P$ and $Hits@k$, are commonly used (e.g., Su et al., 2019b; Zhao et al., 2020b; Zhang et al., 2019a). These metrics require that for each anomalous example x^j, there is a specific set of data streams C^j with a variable number of elements $|C^j|$ that are selected manually by expert knowledge as relevant and are dependent on the class label as shown in Table 11. The metric $Hits@k$ counts the number of examples $x^j \in \mathcal{D}_{test}$ for which any of the first k entries of c_i is included in the set C^j. More formally that can be written as

$$\text{Hits@k} = \frac{\left|\left\{ x^j \in \mathcal{D}_{test} \mid \underset{i \le k}{\exists} c_i^j \in C^j \right\}\right|}{|\mathcal{D}_{test}|} \tag{18}$$

For instance, for $k = 3$, a hit would be counted if any of the first three entries of c is contained in the set C.

The $HitRate@p\%$ metric is another measure employed in this context (e.g., Su et al., 2019b; Zhao et al., 2020b). The parameter k is adjusted based on the number of attributes $|C|$ of the example being examined and the parameter p. Therefore, for each example $x^j \in \mathcal{D}_{test}$, the number of $k^j = |C^j| * p\%$ is used instead of a fixed value for k as in the case of $Hits@k$.

$$\text{HitRate@p\%} = \frac{\left|\left\{ x^j \in \mathcal{D}_{test} \mid \underset{i \le k^j}{\exists} c_i^j \in C^j \right\}\right|}{|\mathcal{D}_{test}|} \tag{19}$$

If $p = 100$, the value of k^j is equivalent to the gold standard number given in $|C^j|$. Increasing $p\%$ increases the number of entries considered in c, as well as a higher $|C^j|$ increases the likelihood of a hit since a larger portion of c^j is considered. For both $Hits@k$ and $HitRate@p\%$, a value of 1 indicates optimal performance, with higher values being preferable.

[13] Formula found on https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html#tsfresh.feature_extraction.feature_calculators.absolute_sum_of_changes, last accessed on 06/15/2022.

Data availability

I have shared the link to the data in the footnote.

References

Aamodt, A., Plaza, E., 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Commun. 7 (1), 39–59.

Abele, E., Chryssolouris, G., Sihn, W., Metternich, J., Maraghy, H.E., Seliger, G., Sivard, G., ElMaraghy, W., Hummel, V., Tisch, M., Seifermann, S., 2017. Learning factories for future oriented research and education in manufacturing. CIRP Ann 66 (2), 803–826. http://dx.doi.org/10.1016/j.cirp.2017.05.005, URL: http://www.sciencedirect.com/science/article/pii/S0007850617301440.

Alaverdyan, Z., Jung, J., Bouet, R., Lartizien, C., 2020. Regularized siamese neural network for unsupervised outlier detection on brain multiparametric magnetic resonance imaging: Application to epilepsy lesion screening. Med. Image Anal. 60, http://dx.doi.org/10.1016/j.media.2019.101618, URL: https://hal.archives-ouvertes.fr/hal-02995591.

Ali, N., Hong, J., 2018. Failure detection and prevention for cyber-physical systems using ontology-based knowledge base. Comput. 7 (4), 68. http://dx.doi.org/10.3390/computers7040068.

Alimohammadi, H., Nancy Chen, S., 2022. Performance evaluation of outlier detection techniques in production timeseries: A systematic review and meta-analysis. Expert Syst. Appl. 191, 116371. http://dx.doi.org/10.1016/j.eswa.2021.116371, URL: https://www.sciencedirect.com/science/article/pii/S095741742101664X.

Antwarg, L., Shapira, B., Rokach, L., 2019. Explaining anomalies detected by autoencoders using SHAP. CoRR abs/1903.02407, arXiv:1903.02407, URL: http://arxiv.org/abs/1903.02407.

Beckh, K., Müller, S., Jakobs, M., Toborek, V., Tan, H., Fischer, R., Welke, P., Houben, S., von Rüden, L., 2021. Explainable machine learning with prior knowledge: An overview. CoRR abs/2105.10172, arXiv:2105.10172, URL: https://arxiv.org/abs/2105.10172.

Bergman, L., Cohen, N., Hoshen, Y., 2020. Deep nearest neighbor anomaly detection. CoRR abs/2002.10445, arXiv:2002.10445, URL: https://arxiv.org/abs/2002.10445.

Bergmann, R., 2002. Experience management: Foundations, development methodology, and internet-based applications. Lecture Notes in Computer Science, vol. 2432, Springer, http://dx.doi.org/10.1007/3-540-45759-3.

Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O., 2013. Translating embeddings for modeling multi-relational data. In: Burges, C.J.C., Bottou, L., Ghahramani, Z., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. pp. 2787–2795, https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.

Boschin, A., Jain, N., Keretchashvili, G., Suchanek, F.M., 2022. Combining embeddings and rules for fact prediction (invited paper). In: Int. Research School in Artif. Intell. in Bergen. In: OASIcs, vol. 99, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 4:1–4:30. http://dx.doi.org/10.4230/OASIcs.AIB.2022.4.

Bouhadra, K., Forest, F., 2024. Knowledge-based and Expert Systems in Prognostics and Health Management: a Survey. International Journal of Prognostics and Health Management (ISSN: 2153-2648) 15 (2), http://dx.doi.org/10.36001/ijphm.2024.v15i2.3986.

Breit, A., Waltersdorfer, L., Ekaputra, F.J., Sabou, M., Ekelhart, A., Iana, A., Paulheim, H., Portisch, J., Revenko, A., Teije, A.T., Van Harmelen, F., 2023. Combining machine learning and semantic web: A systematic mapping study. ACM Comput. Surv. 55 (14s), http://dx.doi.org/10.1145/3586163.

Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R., 1993. Signature verification using a "siamese" time delay neural network. Int. J. Pattern Recognit. Artif. Intell. 7 (4), 669–688. http://dx.doi.org/10.1142/S0218001493000339.

Bulla, C., Birje, M.N., 2021. Improved data-driven root cause analysis in fog computing environment. J. Reliab. Intell. Environ. 1–19.

Burge, S., 2011. The systems engineering tool box. URL: https://www.burgehugheswalsh.co.uk/Uploaded/1/Documents/FFMEA-Tool-v2.pdf, Available at https://www.burgehugheswalsh.co.uk/Uploaded/1/Documents/FFMEA-Tool-v2.pdf, (Accessed 16 July 2021).

Cao, Q., Giustozzi, F., Zanni-Merk, C., De Bertrand de Beuvron, F., Reich, C., 2019. Smart condition monitoring for industry 4.0 manufacturing processes: An ontology-based approach. Cybern. Syst. 50, 1–15. http://dx.doi.org/10.1080/01969722.2019.1565118.

Cao, Q., Zanni-Merk, C., Samet, A., Reich, C., de Beuvron, F.d., Beckmann, A., Giannetti, C., 2022. KSPMI: A knowledge-based system for predictive maintenance in industry 4.0. Robot. Comput. Integr. Manuf. 74, 102281. http://dx.doi.org/10.1016/J.RCIM.2021.102281.

Castellani, A., Schmitt, S., Squartini, S., 2020. Real-world anomaly detection by using digital twin systems and weakly supervised learning. IEEE Trans. Ind. Inform. 17 (7), 4733–4742.

Chen, X., He, K., 2020. Exploring simple siamese representation learning. CoRR abs/2011.10566, arXiv:2011.10566, URL: https://arxiv.org/abs/2011.10566.

Chen, J., Hu, P., Jiménez-Ruiz, E., Holter, O.M., Antonyrajah, D., Horrocks, I., 2020. OWL2Vec*: Embedding of OWL ontologies. CoRR abs/2009.14654, arXiv:2009.14654, URL: https://arxiv.org/abs/2009.14654.

Chollet, F., et al., 2015. Keras. https://keras.io.

Clevert, D., Unterthiner, T., Hochreiter, S., 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In: 4th Int. Conf. on Learning Representations, ICLR 2016. URL: http://arxiv.org/abs/1511.07289.

Dalzochio, J., Kunst, R., Pignaton, E., Binotto, A., Sanyal, S., Favilla, J., Barbosa, J., 2020. Machine learning and reasoning for predictive maintenance in industry 4.0: Current status and challenges. Comput. Ind. 123, 103298. http://dx.doi.org/10.1016/j.compind.2020.103298.

Darban, Z.Z., Webb, G.I., Pan, S., Aggarwal, C.C., Salehi, M., 2022. Deep learning for time series anomaly detection: A survey. http://dx.doi.org/10.48550/arXiv.2211.05244, CoRR abs/2211.05244, arXiv:2211.05244.

De Paepe, D., Vanden Hautte, S., Steenwinckel, B., Moens, P., Vaneessen, J., Vandekerckhove, S., Volckaert, B., Ongenae, F., Van Hoecke, S., 2021. A complete software stack for IoT time-series analysis that combines semantics and machine learning - Lessons learned from the dyversify project. Appl. Sci. 11 (24), http://dx.doi.org/10.3390/app112411932, URL: https://www.mdpi.com/2076-3417/11/24/11932.

Delaney, E., Greene, D., Keane, M.T., 2020. Instance-based counterfactual explanations for time series classification. arXiv preprint arXiv:2009.13211.

Delaney, E., Greene, D., Keane, M.T., 2021. Instance-based counterfactual explanations for time series classification. In: Case-Based Reasoning Research and Development - 29th International Conference, ICCBR 2021. Vol. 12877, Springer, pp. 32–47. http://dx.doi.org/10.1007/978-3-030-86957-1_3.

Dempster, A., Petitjean, F., Webb, G.I., 2020. ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Know. Discov. 1–42.

Deng, A., Hooi, B., 2021. Graph neural network-based anomaly detection in multivariate time series. In: Thirty-Fifth AAAI Conf. on Artif. Intell., AAAI 2021. AAAI Press, pp. 4027–4035, URL: https://ojs.aaai.org/index.php/AAAI/article/view/16523.

Diedrich, A., 2023. On Diagnosing Cyber-Physical Systems (Ph.D. thesis). Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg.

Farbiz, F., Habibullah, M.S., Hamadicharef, B., Maszczyk, T., Aggarwal, S., 2022. Knowledge-embedded machine learning and its applications in smart manufacturing. J. Intell. Manuf. 1–18.

Fatemi, B., Asri, L.E., Kazemi, S.M., 2021. SLAPS: Self-supervision improves structure learning for graph neural networks. In: Advances in Neural Information Processing Systems. pp. 22667–22681, URL: https://proceedings.neurips.cc/paper/2021/hash/bf499a12e998d178afd964adf64a60cb-Abstract.html.

Foorthuis, R., 2020. On the nature and types of anomalies: A review. CoRR abs/2007.15634, arXiv:2007.15634, URL: https://arxiv.org/abs/2007.15634.

Fourure, D., Javaid, M.U., Posocco, N., Tihon, S., 2021. Anomaly detection: How to artificially increase your F1-score with a biased evaluation protocol. In: Dong, Y., Kourtellis, N., Hammer, B., Lozano, J.A. (Eds.), Machine Learning and Know. Discovery in Databases. Applied Data Science Track - European Conference, ECML PKDD 2021. In: Lecture Notes in Comput. Science, vol. 12978, Springer, pp. 3–18. http://dx.doi.org/10.1007/978-3-030-86514-6_1.

Franciosi, C., Eslami, Y., Lezoche, M., Voisin, A., 2024. Ontologies for prognostics and health management of production systems: overview and research challenges. Journal of Intelligent Manufacturing 1–31.

Garofalo, M., Pellegrino, M.A., Altabba, A., Cochez, M., 2018. Leveraging knowledge graph embedding techniques for industry 4.0 use cases. In: Cyber Defence in Industry 4.0 Systems and Related Logistics and IT Infrastructures. IOS Press, pp. 10–26.

Giunchiglia, E., Stoian, M.C., Lukasiewicz, T., 2022. Deep learning with logical constraints. http://dx.doi.org/10.48550/arXiv.2205.00523, CoRR abs/2205.00523, arXiv:2205.00523.

Goix, N., 2016. How to evaluate the quality of unsupervised anomaly detection algorithms? CoRR abs/1607.01152, arXiv:1607.01152, URL: http://arxiv.org/abs/1607.01152.

Golan, I., El-Yaniv, R., 2018. Deep anomaly detection using geometric transformations. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. pp. 9781–9791, URL: https://proceedings.neurips.cc/paper/2018/hash/5e62d03aec0d17facfc5355dd90d441c-Abstract.html.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M.R., Venkatesh, S., van den Hengel, A., 2019. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: 2019 IEEE/CVF Int. Conf. on Comput. Vision, ICCV 2019. IEEE, pp. 1705–1714. http://dx.doi.org/10.1109/ICCV.2019.00179.

Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.Á., Guo, Z.D., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M., 2020. Bootstrap your own latent: A new approach to self-supervised learning. CoRR abs/2006.07733, arXiv:2006.07733, URL: https://arxiv.org/abs/2006.07733.

Guidotti, R., 2022. Counterfactual explanations and how to find them: literature review and benchmarking. Data Min. Know. Discov. 1–55.

Günel, A., Meshram, A., Bley, T., Schuetze, A., Klusch, M., 2013. Statistical and semantic multisensor data evaluation for fluid condition monitoring in wind turbines. In: Proc. 16th Intl. Conf. on Sensors and Measurement Technology, Germany. pp. 604–609.

Guo, J., Li, Z., Li, M., 2019. A review on prognostics methods for engineering systems. IEEE Trans. Reliab. 1–20. http://dx.doi.org/10.1109/TR.2019.2957965.

Guo, S., Wang, Q., Wang, L., Wang, B., Guo, L., 2016. Jointly embedding knowledge graphs and logical rules. In: Proc. of the 2016 Conf. on Empirical Methods in Natural Language Processing. ACL, Austin, Texas, pp. 192–202. http://dx.doi.org/10.18653/v1/D16-1019, URL: https://aclanthology.org/D16-1019.

Hagendorfer, E.J., 2021. Knowledge incorporation for machine learning in condition monitoring: A survey. In: ISEEIE 2021: Int. Symp. on Electrical, Electronics and Information Eng., 2021. ACM, pp. 230–240. http://dx.doi.org/10.1145/3459104.3459144.

Hajek, P., 1998. The Metamathematics of Fuzzy Logic. Kluwer.

Haller, A., Janowicz, K., Cox, S.J.D., Lefrançois, M., Taylor, K., Phuoc, D.L., Lieberman, J., García-Castro, R., Atkinson, R., Stadler, C., 2019. The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semant. Web 10 (1), 9–32. http://dx.doi.org/10.3233/SW-180320.

Harris, S., Seaborne, A., 2013. SPARQL 1.1 query language. In: W3C Recommendation. Available at https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

Hashemi, S., Mäntylä, M., 2021. Detecting anomalies in software execution logs with siamese network. CoRR abs/2102.01452, arXiv:2102.01452, URL: https://arxiv.org/abs/2102.01452.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. CoRR abs/1512.03385, arXiv:1512.03385, URL: http://arxiv.org/abs/1512.03385.

Hitzler, P., et al., 2012. OWL 2 web ontology language primer. In: W3C Recommendation. Available at https://www.w3.org/TR/2012/REC-owl2-primer-20121211/.

Hubauer, T., et al., 2018. Use cases of the industrial knowledge graph at siemens. In: Proc. of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks. In: CEUR Workshop Proc., vol. 2180, CEUR-WS.org.

Huong, T.T., Bac, T.P., Long, D.M., Luong, T.D., Dan, N.M., Thang, B.D., Tran, K.P., et al., 2021. Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach. Comput. Ind. 132, 103509.

Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F., 2020. A survey on contrastive self-supervised learning. CoRR abs/2011.00362, arXiv:2011.00362, URL: https://arxiv.org/abs/2011.00362.

Ji, S., Pan, S., Cambria, E., Marttinen, P., Philip, S.Y., 2021. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Trans. Neural Netw. Learn. Syst..

Jimenez, J.J.M., Schwartz, S., Vingerhoeds, R., Grabot, B., Salaün, M., 2020. Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. J. Manuf. Syst. 56, 539–557.

Kalayci, E.G., et al., 2020. Semantic integration of bosch manufacturing data using virtual knowledge graphs. In: 19th ISWC. In: LNCS, vol. 12507, Springer, pp. 464–481.

Karlsson, I., Rebane, J., Papapetrou, P., Gionis, A., 2020. Locally and globally explainable time series tweaking. Know. Inf. Syst. 62 (5), 1671–1700.

Khoshnevisan, F., Fan, Z., 2019. RSM-GAN: A convolutional recurrent GAN for anomaly detection in contaminated seasonal multivariate time series. CoRR abs/1911.07104, arXiv:1911.07104, URL: http://arxiv.org/abs/1911.07104.

Kim, S., Choi, K., Choi, H., Lee, B., Yoon, S., 2021. Towards a rigorous evaluation of time-series anomaly detection. CoRR abs/2109.05257, arXiv:2109.05257, URL: https://arxiv.org/abs/2109.05257.

Klein., P., Bergmann., R., 2019. Generation of complex data for AI-based predictive maintenance research with a physical factory model. In: Proc. of the 16th Int. Conf. on Informatics in Control, Automation and Robotics - Volume 1: ICINCO. INSTICC, SciTePress, pp. 40–50. http://dx.doi.org/10.5220/0007830700400050.

Klein, P., Malburg, L., Bergmann, R., 2019a. Ftonto: A domain ontology for a fischertechnik simulation production factory by reusing existing ontologies. In: Jäschke, R., Weidlich, M. (Eds.), Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen", Berlin, Germany, September 30 - October 2, 2019. In: CEUR Workshop Proceedings, vol. 2454, CEUR-WS.org, pp. 253–264, URL: http://ceur-ws.org/Vol-2454/paper_10.pdf.

Klein, P., Malburg, L., Bergmann, R., 2019b. Learning workflow embeddings to improve the performance of similarity-based retrieval for process-oriented case-based reasoning. In: Int. Conf. on Case-Based Reasoning. Springer, pp. 188–203.

Klein, P., Weingarz, N., Bergmann, R., 2020. Enhancing siamese neural networks through expert knowledge for predictive maintenance. In: IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning. In: Communications in Computer and Information Science, vol. 1325, Springer International Publishing., pp. 77–92. http://dx.doi.org/10.1007/978-3-030-66770-2_6.

Klein, P., Weingarz, N., Bergmann, R., 2021. Using expert knowledge for masking irrelevant data streams in siamese networks for the detection and prediction of faults. In: 2021 Int. Joint Conf. on Neural Networks. IJCNN, pp. 1–10. http://dx.doi.org/10.1109/IJCNN52387.2021.9533544.

Lamy, J.-B., 2017. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. Artif. Intell. Med. 80, 11–28.

Li, D., Chen, D., Shi, L., Jin, B., Goh, J., Ng, S., 2019. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. CoRR abs/1901.04997, arXiv:1901.04997, URL: http://arxiv.org/abs/1901.04997.

Li, L., Zhang, X., Jin, Z., Gao, C., Zhu, R., Liang, Y., Ma, Y., 2023. Knowledge graph completion method based on quantum embedding and quaternion interaction enhancement. Inf. Sci. 648, 119548. http://dx.doi.org/10.1016/J.INS.2023.119548, https://doi.org/10.1016/j.ins.2023.119548.

Li, Z., Zhao, Y., Han, J., Su, Y., Jiao, R., Wen, X., Pei, D., 2021. Multivariate time series anomaly detection and interpretation using hierarchical inter-metric and temporal embedding. In: Zhu, F., Ooi, B.C., Miao, C. (Eds.), KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021. ACM, pp. 3220–3230. http://dx.doi.org/10.1145/3447548.3467075.

Lira, D.N., Borsato, M., 2018. OntoProg: An ontology-based model for implementing prognostics health management in mechanical machines. Adv. Eng. Inform. 38, 746–759. http://dx.doi.org/10.1016/j.aei.2018.10.006.

Liznerski, P., Ruff, L., Vandermeulen, R.A., Franks, B.J., Müller, K., Kloft, M., 2022. Exposing outlier exposure: What can be learned from few, one, and zero outlier images. http://dx.doi.org/10.48550/arXiv.2205.11474, CoRR abs/2205.11474, arXiv:2205.11474.

Lundberg, S.M., Lee, S., 2017. A unified approach to interpreting model predictions. In: Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA. pp. 4765–4774, URL: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Luo, Y., Xiao, Y., Cheng, L., Peng, G., Yao, D., 2021. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. ACM Comput. Surv. 54 (5), 1–36.

Malburg, L., Brand, F., Bergmann, R., 2023a. Adaptive management of cyber-physical workflows by means of case-based reasoning and automated planning. In: 26th EDOC Workshops. In: LNBIP, vol. 466, Springer, pp. 79–95.

Malburg, L., Hoffmann, M., Bergmann, R., 2023b. Applying MAPE-K control loops for adaptive workflow management in smart factories. J. Intell. Inf. Syst. 1–29.

Malburg, L., Klein, P., Bergmann, R., 2020. Semantic web services for AI-research with physical factory simulation models in industry 4.0. In: Panetto, H., Madani, K., Smirnov, A.V. (Eds.), Proceedings of the International Conference on Innovative Intelligent Industrial Production and Logistics, IN4PL 2020, Budapest, Hungary, November 2-4, 2020. SCITEPRESS, pp. 32–43. http://dx.doi.org/10.5220/0010135900320043.

Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G., 2016. Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. CoRR abs/1608.06154, arXiv:1608.06154, URL: http://arxiv.org/abs/1608.06154.

Masana, M., Ruiz, I., Serrat, J., van de Weijer, J., López, A.M., 2018. Metric learning for novelty and anomaly detection. In: British Machine Vision Conf. 2018, BMVC 2018. BMVA Press, p. 64, URL: http://bmvc2018.org/contents/papers/0178.pdf.

Mazzola, L., Kapahnke, P., Vujic, M., Klusch, M., 2016. CDM-Core: A manufacturing domain ontology in OWL2 for production and maintenance. In: Proc. of the 8th Int. Joint Conf. on Know. Discovery, Know. Eng. and Know. Management - Vol. 2: KEOD, Portugal. pp. 136–143.

Medina-Oliva, G., Voisin, A., Monnin, M., Léger, J., 2014. Predictive diagnosis based on a fleet-wide ontology approach. Knowl. Based Syst. 68, 40–57. http://dx.doi.org/10.1016/J.KNOSYS.2013.12.020, https://doi.org/10.1016/j.knosys.2013.12.020.

Monostori, L., 2014. Cyber-physical production systems: Roots, expectations and R&D challenges. Procedia CIRP 17, 9–13. http://dx.doi.org/10.1016/j.procir.2014.03.115, URL: http://www.sciencedirect.com/science/article/pii/S2212827114003497, Variety Management in Manufacturing.

Müller, C., Lunde, R., Hönig, P., 2020. Generation of a failure mode and effects analysis with smartiflow. In: Proc. of the 30th European Safety and Reliability Conf. Andthe 15th Probabilistic Safety Assessment and Management Conference. pp. 1662–1669.

Nguyen, H., Tran, K., Zeng, X., Koehl, L., Castagliola, P., Bruniaux, P., 2019. Industrial internet of things, big data, and artificial intelligence in the smart factory: A survey and perspective. In: ISSAT International Conference on Data Science in Business, Finance and Industry. pp. 72–76.

Niggemann, O., Lohweg, V., 2015. On the diagnosis of cyber-physical production systems. In: Bonet, B., Koenig, S. (Eds.), Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. AAAI Press, pp. 4119–4126. http://dx.doi.org/10.1609/AAAI.V29I1.9762.

Nunes, P., Santos, J., Rocha, E., 2023. Challenges in predictive maintenance – A review. CIRP J. Manuf. Sci. Technol. 40, 53–67. http://dx.doi.org/10.1016/j.cirpj.2022.11.004, URL: https://www.sciencedirect.com/science/article/pii/S1755581722001742.

Nyulászi, L., Andoga, R., Butka, P., Főző, L., Kovacs, R., Moravec, T., 2018. Fault detection and isolation of an aircraft turbojet engine using a multi-sensor network and multiple model approach. Acta Polytech. Hung. 15 (2), 189–209.

Park, D., Hoshi, Y., Kemp, C.C., 2017. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. arXiv:1711.00614.

Pecht, M., Gu, J., 2009. Physics-of-failure-based prognostics for electronic products. Trans. Inst. Meas. Control. 31 (3–4), 309–322.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Peffers, K., Tuunanen, T., Niehaves, B., 2018. Design science research genres: introduction to the special issue on exemplars and criteria for applicable design science research. Eur. J. Inf. Syst. 27 (2), 129–139. http://dx.doi.org/10.1080/0960085X.2018.1458066, arXiv:https://doi.org/10.1080/0960085X.2018.1458066.

Qiu, C., Pfrommer, T., Kloft, M., Mandt, S., Rudolph, M., 2021. Neural transformation learning for deep anomaly detection beyond images. arXiv preprint arXiv:2103.16440.

Radtke, M.-P., Bock, J., 2022. Expert knowledge induced logic tensor networks: A bearing fault diagnosis case study. In: PHM Society European Conference. pp. 421–431.

Raschka, S., 2020. STAT 451: Machine Learning Lecture Notes. University of Wisconsin–Madison.

Reiss, T., Cohen, N., Bergman, L., Hoshen, Y., 2021. Panda: Adapting pretrained features for anomaly detection and segmentation. In: Proc. of the IEEE/CVF Conf. on Comput. Vision and Pattern Recognition. pp. 2806–2814.

Rezaeianjouybari, B., Shang, Y., 2020. Deep learning for prognostics and health management: State of the art, challenges, and opportunities. Meas. 163, 107929. http://dx.doi.org/10.1016/j.measurement.2020.107929, URL: https://www.sciencedirect.com/science/article/pii/S026322412030467X.

Roelofs, C.M., Lutz, M.-A., Faulstich, S., Vogt, S., 2021. Autoencoder-based anomaly root cause analysis for wind turbines. Energy AI 100065. http://dx.doi.org/10.1016/j.egyai.2021.100065, URL: https://www.sciencedirect.com/science/article/pii/S2666546821000197.

Roth, K., Pemula, L., Zepeda, J., Schölkopf, B., Brox, T., Gehler, P.V., 2022. Towards total recall in industrial anomaly detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, la, USA, June 18-24, 2022. IEEE, pp. 14298–14308. http://dx.doi.org/10.1109/CVPR52688.2022.01392.

Ruff, L., Görnitz, N., Deecke, L., Siddiqui, S.A., Vandermeulen, R.A., Binder, A., Müller, E., Kloft, M., 2018. Deep one-class classification. In: Proc. of the 35th Int. Conf. on Machine Learning, ICML 2018. Vol. 80, PMLR, pp. 4390–4399, URL: http://proceedings.mlr.press/v80/ruff18a.html.

Ruff, L., Vandermeulen, R.A., Franks, B.J., Müller, K., Kloft, M., 2020. Rethinking assumptions in deep anomaly detection. CoRR abs/2006.00339, arXiv:2006.00339, URL: https://arxiv.org/abs/2006.00339.

Schölkopf, B., Williamson, R.C., Smola, A., Shawe-Taylor, J., Platt, J., 1999. Support vector method for novelty detection. Adv. Neural Inf. Process. Syst. 12.

Selcuk, S., 2017. Predictive maintenance, its implementation and latest trends. Proc. Inst. Mech. Eng. B 231 (9), 1670–1679.

Serradilla, O., Zugasti, E., Zurutuza, U., 2020a. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospect. CoRR abs/2010.03207, arXiv:2010.03207, URL: https://arxiv.org/abs/2010.03207.

Serradilla, O., Zugasti, E., Zurutuza, U., 2020b. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospect. arXiv:2010.03207.

Shen, L., Li, Z., Kwok, J., 2020. Timeseries anomaly detection using temporal hierarchical one-class network. Adv. Neural Inf. Process. Syst. 33.

Skvára, V., Pevný, T., Smídl, V., 2018. Are generative deep models for novelty detection truly better? CoRR abs/1807.05027, arXiv:1807.05027, URL: http://arxiv.org/abs/1807.05027, presented at SIGKDD 2018 at the Outlier Detection De-constructed (ODD) Workshop. KDD ODDv5.0 2018.

Sohn, K., Li, C., Yoon, J., Jin, M., Pfister, T., 2021. Learning and evaluating representations for deep one-class classification. In: 9th Int. Conf. on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, URL: https://openreview.net/forum?id=HCSgyPUfeDj.

Steenwinckel, B., De Paepe, D., Vanden Hautte, S., Heyvaert, P., Bentefrit, M., Moens, P., Dimou, A., Van Den Bossche, B., De Turck, F., Van Hoecke, S., Ongenae, F., 2021a. FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. Future Gener. Comput. Syst. 116, 30–48. http://dx.doi.org/10.1016/j.future.2020.10.015, URL: http://www.sciencedirect.com/science/article/pii/S0167739X20329927.

Steenwinckel, B., Heyvaert, P., De Paepe, D., Janssens, O., Vanden Hautte, S., Dimou, A., De Turck, F., Van Hoecke, S., Ongenae, F., 2018. Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses. In: 9th Int. Semantic Sensor Networks Workshop, Co-Located with 17th Int. Semantic Web Conf. (ISWC 2018). Vol. 2213, pp. 17–31.

Steenwinckel, B., Paepe, D.D., Hautte, S.V., Heyvaert, P., Bentefrit, M., Moens, P., Dimou, A., Bossche, B.V.D., Turck, F.D., Hoecke, S.V., Ongenae, F., 2021b. FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. Future Gener. Comput. Syst. 116, 30–48, URL: https://doi.org/10.1016/j.future.2020.10.015.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D., 2019a. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (Eds.), Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. ACM, pp. 2828–2837. http://dx.doi.org/10.1145/3292500.3330672.

Su, Y., Zhao, Y., Niu, C., Liu, R., Sun, W., Pei, D., 2019b. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In: Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G. (Eds.), Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019. ACM, pp. 2828–2837, URL: https://doi.org/10.1145/3292500.3330672.

Sulem, D., Donini, M., Zafar, M.B., Aubet, F., Gasthaus, J., Januschowski, T., Das, S., Kenthapadi, K., Archambeau, C., 2022. Diverse counterfactual explanations for anomaly detection in time series. http://dx.doi.org/10.48550/arXiv.2203.11103, CoRR abs/2203.11103, arXiv:2203.11103, URL: https://doi.org/10.48550/arXiv.2203.11103.

Tamssaouet, F., Nguyen, K.T., Medjaher, K., Orchard, M.E., 2021. Online joint estimation and prediction for system-level prognostics under component interactions and mission profile effects. ISA Trans. 113, 52–63. http://dx.doi.org/10.1016/j.isatra.2020.05.002, URL: https://www.sciencedirect.com/science/article/pii/S0019057820301762.

Theusch, F., Klein, P., Bergmann, R., Wilke, W., Bock, W., Weber, A., 2021. Fault detection and condition monitoring in district heating using smart meter data. In: PHM Society European Conference. Vol. 6, 11–11.

Utkin, L.V., Kovalev, M.S., Kasimov, E.M., 2019. An explanation method for siamese neural networks. CoRR abs/1911.07702, arXiv:1911.07702, URL: http://arxiv.org/abs/1911.07702.

Vollert, S., Atzmueller, M., Theissler, A., 2021. Interpretable machine learning: A brief survey from the predictive maintenance perspective. In: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation. ETFA, pp. 01–08. http://dx.doi.org/10.1109/ETFA45728.2021.9613467.

Von Alan, R.H., March, S.T., Park, J., Ram, S., 2004. Design science in information systems research. MIS Q. 28 (1), 75–105.

von Hahn, T., Mechefske, C.K., 2022. Knowledge informed machine learning using a Weibull-based loss function. CoRR abs/2201.01769, arXiv:2201.01769, URL: https://arxiv.org/abs/2201.01769.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., et al., 2019. Informed machine learning–A taxonomy and survey of integrating knowledge into learning systems. arXiv preprint arXiv:1903.12394.

Wu, L.Y., Fisch, A., Chopra, S., Adams, K., Bordes, A., Weston, J., 2018. Starspace: Embed all the things!. In: AAAI Conf. on Artif. Intell.. pp. 5569–5577.

Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C., 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proc. of the 26th ACM SIGKDD Int. Conf. on Know. Discovery & Data Mining. pp. 753–763.

Xu, F., Liu, X., Chen, W., Zhou, C., Cao, B., 2018a. Ontology-based method for fault diagnosis of loaders. Sensors 18 (3), 729.

Xu, J., Zhang, Z., Friedman, T., Liang, Y., den Broeck, G.V., 2018b. A semantic loss function for deep learning with symbolic knowledge. In: Proc. of the Int. Conf. on Machine Learning, ICML. Vol. 80, PMLR, pp. 5498–5507, URL: http://proceedings.mlr.press/v80/xu18h.html.

Yang, Z., Ishay, A., Lee, J., 2020. Neurasp: Embracing neural networks into answer set programming. In: Bessiere, C. (Ed.), Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020. ijcai.org, pp. 1755–1762. http://dx.doi.org/10.24963/IJCAI.2020/243.

Yeh, C.-C.M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H.A., Silva, D.F., Mueen, A., Keogh, E., 2016. Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th International Conference on Data Mining. ICDM, Ieee, pp. 1317–1322.

Zhang, J., Chen, B., Zhang, L., Ke, X., Ding, H., 2021. Neural, symbolic and neural-symbolic reasoning on knowledge graphs. AI Open 2, 14–35. http://dx.doi.org/10.1016/j.aiopen.2021.03.001.

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., Chawla, N.V., 2019a. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: Proc. of the AAAI Conf. on Artif. Intell.. Vol. 33, pp. 1409–1416.

Zhang, C., Song, D., Chen, Y., Feng, X., Lumezanu, C., Cheng, W., Ni, J., Zong, B., Chen, H., Chawla, N.V., 2019b. A deep neural network for unsupervised anomaly detection and diagnosis in multivariate time series data. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, the Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. AAAI Press, pp. 1409–1416, URL: https://doi.org/10.1609/aaai.v33i01.33011409.

Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., Zhang, Q., 2020a. Multivariate time-series anomaly detection via graph attention network. CoRR abs/2009.02040, arXiv:2009.02040, URL: https://arxiv.org/abs/2009.02040.

Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., Zhang, Q., 2020b. Multivariate time-series anomaly detection via graph attention network. In: 20th IEEE Int. Conf. on Data Mining, ICDM 2020. IEEE, pp. 841–850. http://dx.doi.org/10.1109/ICDM50108.2020.00093.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A., 2016. Learning deep features for discriminative localization. In: Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition. pp. 2921–2929.

Zhou, A., Yu, D., Zhang, W., 2015. A research on intelligent fault diagnosis of wind turbines based on ontology and FMECA. Adv. Eng. Inform. 29 (1), 115–125.

Zhu, Y., Xu, W., Zhang, J., Liu, Q., Wu, S., Wang, L., 2021. Deep graph structure learning for robust representations: A survey. CoRR abs/2103.03036, arXiv:2103.03036, URL: https://arxiv.org/abs/2103.03036.