ML4CPS
MACHINE LEARNING FOR CYBER-PHYSICAL-SYSTEMS

Machine Learning for Cyber Physical Systems

Proceedings of the Conference

ML4CPS 2024

Fraunhofer IOSB

HELMUT SCHMIDT UNIVERSITÄT
Universität der Bundeswehr Hamburg

OPEN ACCESS

Oliver Niggemann - Jürgen Beyerer - Alexander Diedrich -
Christian Kühnert - Alexander Windmann

Editors

Machine Learning for Cyber Physical Systems

Proceedings of the Conference ML4CPS 2024

Editors

Prof. Jürgen Beyerer
Dr. Christian Kühnert
Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung
Karlsruhe, Germany

Prof. Oliver Niggemann
Dr. Alexander Diedrich
Alexander Windmann
Helmut-Schmidt-Universität Hamburg
Hamburg, Germany

https://doi.org/10.24405/16610

Preface

Cyber Physical Systems are characterized by their ability to adapt and learn from their environment. Applications include advanced condition monitoring, predictive maintenance, diagnosis tasks, and many other areas. All these applications have in common that Machine Learning and Artificial Intelligence are the key technologies. However, applying ML and AI to CPS poses challenges such as limited data, less understood algorithms, and the need for high algorithm reliability. These topics were a focal point at the 7th ML4CPS—Machine Learning for Cyber-Physical Systems Conference in Berlin, held from the 20th to the 21st, where industry and research experts discussed current advancements and new developments.

Prof. Dr. Oliver Niggemann
Prof. Dr. Jürgen Beyerer
Dr. Alexander Diedrich
Dr. Christian Kühnert
Alexander Windmann

Contents

METRICS FOR THE EVALUATION OF LEARNED CAUSAL GRAPHS BASED ON GROUND TRUTH

Josephine Rehak[1], **Alexander Falkenstein**[1], **Frank Doehner**[1], **and Jürgen Beyerer**[1,2]
[1]Karlsruhe Institute of Technology, Kaiserstraße 12, Karlsruhe
[2]Fraunhofer IOSB, Fraunhoferstraße 1, Karlsruhe

ABSTRACT

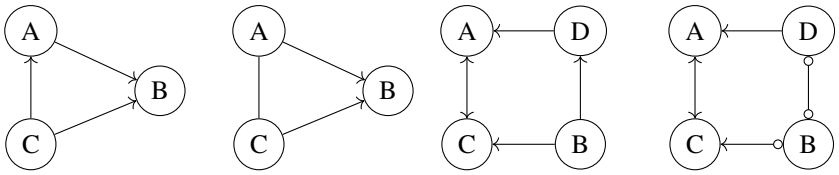
The self-guided learning of causal relations may contribute to the general maturity of artificial intelligence in the future. To develop such learning algorithms, powerful metrics are required to track advances. In contrast to learning algorithms, little has been done in regards to developing suitable metrics. In this work, we evaluate current state of the art metrics by inspecting their discovery properties and their considered graphs. We also introduce a new combination of graph notation and metric, which allows for benchmarking given a variety of learned causal graphs. It also allows the use of maximal ancestral graphs as ground truth.

Keywords causal graph · metric · causal discovery · ground truth · bayesian network structure learning · causal structure learning · ancestral graph · acyclic graph

1 Introduction

Causal Discovery (CD) is a domain of artificial intelligence, that targets the identification of cause-effect relationships in data [12]. For the evaluation of CD algorithms, it has become a standard approach to perform novel algorithms on popular benchmarking datasets, for which the ground truth is known [9, 18]. By choosing the metric with desired properties, the learned graph can be compared against the ground truth and the discovery capabilities of the algorithm can be assessed by the resulting score [4]. In this style of evaluation, the metrics and its properties play a vital role. This is why in this work, we will investigate current metrics of CD by assessing their applicability on different graph types learned by algorithms. We highlight desired criteria and inspect how different state of the art metrics facilitate these. Finally, we propose a new metric called the normed causal edit distance metric (nCED).

In Section 2, we give a basic introduction to CD. In Section 3, we first introduce a new graph notation to unify the several established causal graph types before proposing our new metric. In Section 4, we cover related work in the field of CD metrics and prior metric evaluations. In Section 5, we take a closer look into the general topic of metric evaluation. Finally, Section 6 concludes the paper.



(a) DAG (b) PDAG (c) MAG (d) PAG

Figure 1: Depicted are examples of common types of causal graphs in their established representation.

2 Fundamentals of Causal Discovery

2.1 Causal Graphs

In general, it is defined that a causal relation is present from variable a to b written as $a \to b$, if the value of b depends on the value of a. If there is no such causal influence present in either direction, they are independent $a \perp\!\!\!\perp b$.

Networks of such causal relations are represented as graphs $G = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Nodes are random variables, representing chosen events or states, while edges indicate the causal relations. In our definition, two variables a, b can be connected by a total of two edges $e_{a,b} \in \mathcal{E}$ and $e_{b,a} \in \mathcal{E}$.

In this work, we employ two different classes of causal graphs. Their main difference is the class of acyclic graphs and the corresponding learning algorithms assume causal sufficiency, the absence of latent variables. Ancestral graphs on the other hand do not. Each class has two representatives as depicted in Figure 1.

Starting with the class of acyclic graphs, Directed Acyclic Graphs (DAGs) [19] allow edges to be either present, indicated by an arrowhead pointed at the caused variable, or absent indicated by missing arrowheads or by the absence of any representation. Cycles in the directed edges are forbidden. This forbids minimal cycles like $a \leftrightarrow b$.

Complete Partial Directed Acyclic Graphs (CPDAGs) may contain besides present and absent edges, also undirected edge pairs like $a - b$. In this case, it is a causal relation exists between a and b but its direction, $e_{a,b}$ or $e_{b,a}$, is unknown. Under the constraint that no cycles are created, $a - b$ may be oriented in either direction to gain DAGs. These DAGs form a Markov equivalence class for given CPDAG [17].

For the class of ancestral graphs, we consider Maximal Ancestral Graphs (MAGs). They allow one-directed and bidirected edges, but forbid any cycles formed by a combination of bidirected and directed edges. In ancestral graphs, bidirected edges $a \leftrightarrow b$ imply the presence of a latent variable l with $l \to a$ and $l \to b$. DAGs are a subset of MAGs which do not contain any bidirected edges [33]. Similar to CPDAGs, a Partial Ancestral Graph (PAG) [23, 24, 25] may additionally include unknown edges represented by circular arrowheads, implying a Markov equivalence class of several MAGs. These unknown edges may be present in single and opposing edges. As CD algorithms for MAG learning have not been developed yet, they will be of less importance in this work.

2.2 Established Metrics

Several metrics that use a ground truth for the evaluation of the discovered graph are currently established in literature.

To explain them, we make use of the following definitions. The count of True Positives edges (TP) is the number of discovered present edges that are also present in the ground truth. The count of True Negatives (TN) is the number of all edges that are neither present in the discovered graph and the ground truth. The count of False Positives (FP) is the number of discovered edges that are not present in the ground truth, and the False Negatives (FN) is the number of edges in the discovered graph that are not present but are missing in the ground truth [13]. We specify FN and FP as undirected, if only the presence of the edge is considered to be compared to the ground truth, but not its orientation.

The first metric, we introduce is the F_1 score [28]. It considers only the present edges and calculates the harmonic mean of precision and recall when comparing the inferred graph to the ground truth.

$$F_1 \text{ score} = \frac{2\,\text{TP}}{2\,\text{TP} + \text{FP} + \text{FN}} \tag{1}$$

The Receiver Operator Curve Area Under Curve (ROC AUC) metric [10, 21] measures the True Positive Rate (TPR) on the False Positive Rate (FPR) several times to plot a curve.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3}$$

The ROC AUC is measured once for an 'empty' graph with a TPR and FPR of zero, for the actual discovered graph, and also for a graph with maximized TPR and FPR to 100 percentage points. Then, the area under this curve is calculated. A ROC AUC of 1 indicates a discovered causal graph that is identical to the ground truth with a TPR of hundred percent and an FPR of zero percent. A ROC AUC of fifty percentage points indicates a discovery performance similar to a random guesser. Zero percentage points indicate the inverse of a successfully CD. The Precision Recall Curve Area Under Curve (PRC AUC) [7] operates similar. Instead of TPR and FPR, it uses precision and recall to calculate the area under curve. The Structural Hamming Distance (SHD) [31, 2] counts the overall number of changes in edges that are required to transform the discovered graph into the ground truth. In the established definition, the allowed changes include only the addition and the deletion of edges.

$$\text{SHD} = \text{FN} + \text{FP} \tag{4}$$

The adapted version for CPDAGs [31] converts the ground truth and the predicted graph to CPDAGs and then counts the operations required.

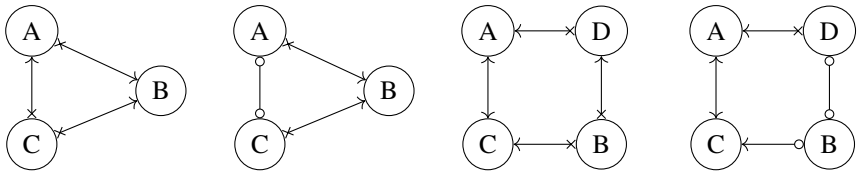
$$\text{SHD CPDAG} = \text{undirected FN} + \text{undirected FP} + \text{FN} + \text{FP} \tag{5}$$

No version of SHD is available for MAGs and PAGs.

3 Causal Edit Distance Metric

3.1 Universal Causal Graph Representation

To make all the various graph types comparable, we transfer them to a unified representation. Consider causal graphs $G = (\mathcal{V}, \mathcal{E})$ with variables V and edges $\mathcal{E}$. The edges $\mathcal{E}$ can be represented as an adjacency matrix $\mathbf{E}$. Each edge $e_{i,j} \in \mathbf{E}$ represents knowledge about a present causal relation from variable i to j and also has a counterpart



(a) DAG (b) PDAG (c) MAG (d) PAG

Figure 2: To compute the nCED, the example graphs of Figure 1 are adapted to match the new universal notation described in Section 3.1.

$e_{j,i}$ pointing in the opposite direction. Edges may take on the values $e_{i,j} \in \{-1, 0, 1\}$. $e_{i,j} = 0$ indicates the absence of a causal relation from i to j. This is represented by $\times$ arrowheads in the universal representation. $e_{i,j} = -1$ indicates the edge is undirected. This is represented by the arrowhead $\circ$ in the universal graph representation. $e_{i,j} = 1$ in the adjacency matrix or $\rightarrow$ in the universal graph representation represent the presence of a causal relation. Hence, for $N = |\mathcal{V}|$, the adjacency matrix is $\mathbf{E} \in \{-1, 0, 1\}^{N \times N}$. The adjacency matrix representation allows cyclic causal relations if such graphs are not actively prohibited. Causal self-references of variables, the smallest possible cycles, are excluded $\forall i \in \mathcal{V}, e_{i,i} = 0$.

3.2 Calculating the Causal Edit Distance

The causal edit distance (CED) between two graphs G and G^* is using a comparison between the corresponding adjacency matrix elements $e_{i,j} \in \mathbf{E}$ and as $e^*_{i,j} \in \mathbf{E}^*$ follows.

$$\text{CED}(G, G^*) = \sum_{\substack{(i,j) \in V^2 \\ i \neq j}} f(e_{i,j}, e^*_{i,j}) \tag{6}$$

For its computation, we require the following function f using the parameter $0 < k < 0.5$.

$$f(e_{i,j}, e^*_{i,j}) = \begin{cases} 0 & \text{if } e_{i,j} = e^*_{i,j} \\ k & \text{if } e_{i,j} \neq e^*_{i,j} \wedge e_{i,j} = -1 \\ 1 & \text{else} \end{cases} \tag{7}$$

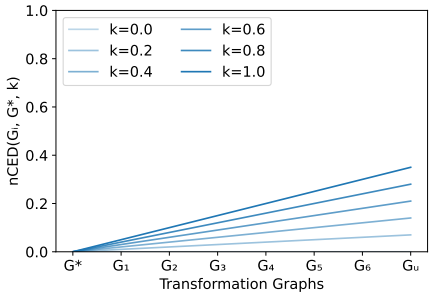
The normalized variant of the Causal Edit Distance metric is called in short nCED. It requires the division of the CED with the maximum possible editing costs $N(N-1)$ given $N = |\mathcal{V}|$ variables.

$$\text{nCED}(G, G^*) = \frac{\text{CED}(G, G^*)}{N(N-1)} \tag{8}$$

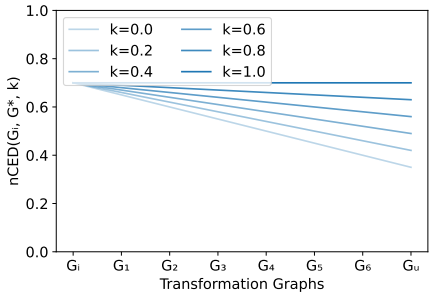
3.3 Analysis on Choosing the Parameter k

Due to the function f, the algorithm punishes each TP and TN with 0, each undirected FP and FN with k and each FP and FN with 1. Given $0 \leq k \leq 1$, k can be freely chosen. Consider, that $k \leq 1$ is required for the normalization step to be valid and $0 \leq k$ for edges to be scored independently.

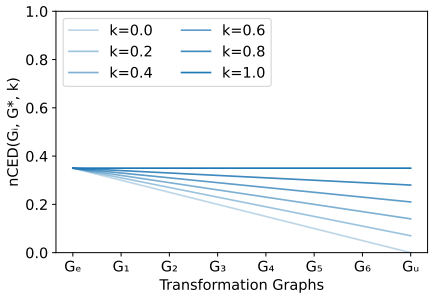
In the following, we propose our own preferences in the choice of k. As undirect TP edges correctly imply the presence of a TP edge, they should be evaluated better than



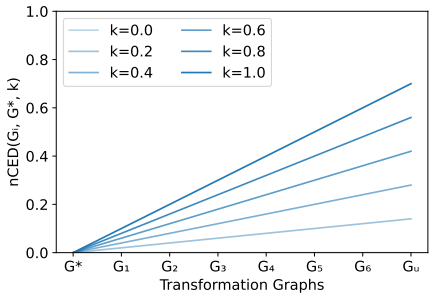
(a) Stepwise transformation from G^* to $G_{u,1}$



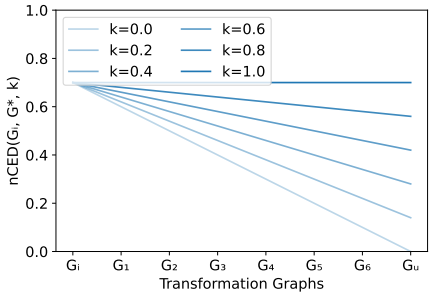
(b) Stepwise transformation from G_i to $G_{u,1}$



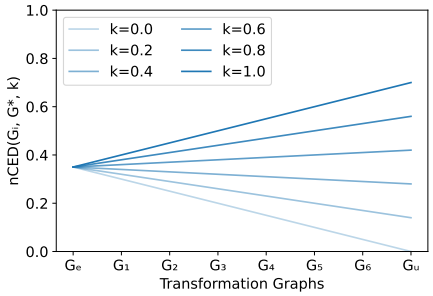
(c) Stepwise transformation from G_e to $G_{u,1}$



(d) Stepwise transformation from G^* to $G_{u,2}$



(e) Stepwise transformation from G_i to $G_{u,2}$



(f) Stepwise transformation from G_e to $G_{u,2}$

Figure 3: Overview of the nCEDs behavior for varying k on stepwise graph transformations of a five variable DAG.

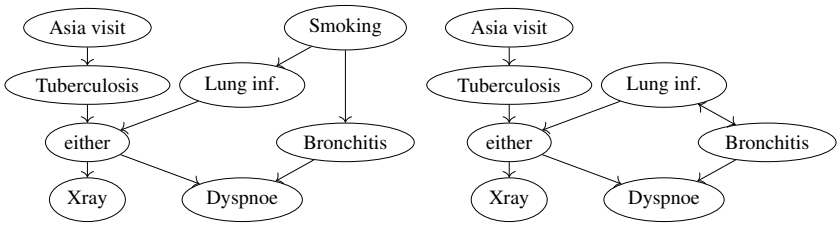
FN edges, but worse than TP edges as they do not imply it with certain orientation.

$$f(e_{i,j} = 1, e^*_{i,j} = 1) < f(e_{i,j} = -1, e^*_{i,j} = 1) < f(e_{i,j} = 0, e^*_{i,j} = 1) \tag{9}$$

Likewise, undirected FP edges should be scored worse than TN edges, as they imply faulty information, but they should be evaluated better than FP edges, since the certain discovery of a FP edge is more severe.

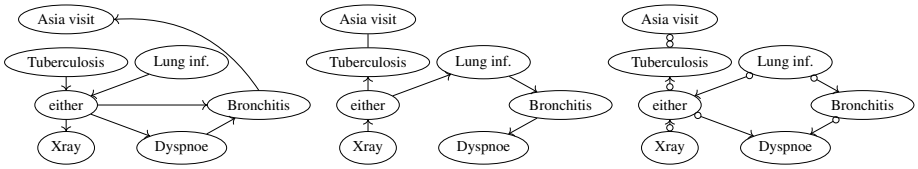
$$f(e_{i,j} = 0, e^*_{i,j} = 0) < f(e_{i,j} = -1, e^*_{i,j} = 0) < f(e_{i,j} = 1, e^*_{i,j} = 0) \tag{10}$$

Accordingly, the score of an undirected graph nCED(G_u, G^*) should be better than the score of an empty graph nCED(G_e, G^*) and better than the score of a graph with flipped



(a) Original DAG (b) Created MAG

Figure 4: The ground truth DAG and the created MAG for the Asia dataset. By eliminating the variable smoking, we create a hidden confounder between the lung infection and the bronchitis variable. The resulting graph is a ground truth MAG.



(a) Example DAG learned by HCS algorithm (b) Example CPDAG learned by PC algorithm (c) Example PAG learned by FCI algorithm

Figure 5: Example graphs as they were created by the different learning algorithms on application on the adapted Asia dataset.

edges nCED(G_i, G^*), but worse than nCED(G^*, G^*). This should hold for undirected graphs with individual undirected edges $G_{u,1}$ (as is possible in PAGs), but also where opposing edges are undirected $G_{u,2}$ (as is the case in CPDAGs).

We observed the behavior of the metric for different k by stepwise transforming G_e, G_i and G^* into $G_{u,2}$, which results in the plots shown in Figure 3. One observes nCED$(G^*, G^*) <$ nCED$(G_{u,2}, G^*)$ and nCED$(G_{u,2}, G^*) <$ nCED(G_i, G^*) to hold true for any k. But while nCED$(G_e, G^*) <$ nCED$(G_{u,1}, G^*)$ is valid for any k, we see in Figure 3f that nCED$(G_e, G^*) <$ nCED$(G_{u,2}, G^*)$ only holds if $k < 0.5$. This is why we propose to choose $0 < k < 0.5$.

Alternate desired choices of k might include $k > 0.5$, so that the algorithm treats undirected edges worse than FN and FP. It may be of use if the consideration of undirected edges is to be punished and only the learning of TP and TN is desired.

For $k = 0$, the nCED does not differ between TP and undirected TP edges. This might be of use if the orientations of an edge matters less than the knowledge of its presence.

3.4 Maximum Ancestral Graphs as Ground Truth

As the nCED may evaluate DAGs, CPDAGs, MAGs and PAGS because of the universal representation, and DAGs are a subclass of MAGs, we can also allow G^* to be a MAG instead of a DAG. As the edge representations of DAGs and MAGs are identical in the universal representation, this does not require an adaptation of nCED's definition. To our knowledge, nCED is the only metric that is capable of this. As a consequence, if $k > 0$ and a bidirected edge is present in the ground truth PAG, the nCED does not allow DAG-learning CD algorithms to achieve the optimal score anymore.

Table 1: Results of the nCED with a MAG as ground truth for graphs learned on the adapted Asia dataset. The lower the score, the better performed the structure learning algorithm.

	nCED (ours)					
	$k = 0$	$k = 0.1$	$k = 0.2$	$k = 0.3$	$k = 0.4$	$k = 0.5$
HCS	0.17	0.17	0.17	0.17	0.17	0.17
PC	0.12	0.13	0.14	0.15	0.17	0.18
FCI	0.12	0.14	0.14	0.15	0.17	0.18

In the following, we demonstrate how the novel nCED metric is able to evaluate results of structure learning algorithms using a PAG as ground truth. Regarding CD learning algorithms, we employed the Peter-Clark (PC) [30] algorithm, which delivers as result a CPDAG, the Fast Causal Inference (FCI) [34] algorithm delivers a PAG, while the hillclimb search (HCS) algorithm [15] delivers a DAG. For independence test the Fisher Z-test was used, and for score-based CD the Bayesian information criterion. Unfortunately, we know of no algorithm which is able to learn MAGs.

As a benchmark, we used the Asia dataset. It describes the causal relations of visits to Asia and smoking to the corresponding lung diseases and the medical options for identifying the disease. We created a ground truth MAG, by eliminating the smoking variable from the dataset as shown in Figure 4. In its place, it has a bidirected edge implying a latent variable between the bronchitis and the lung infection variable. After learning ten graphs using the introduced learning algorithms, including the example graphs shown in Figure 5, we calculated the averaged nCED scores for different k using the created MAG. The results are depicted in Table 1.

4 Related Work

4.1 Metric Types in Causal Research

In CD research, two different approaches exist to evaluate the learned causal graphs [5]. We focus on metrics, which inspect graph elements to compare the discovered graph against a provided ground truth.

As an alternative, scoring functions, which are commonly used as objective functions for score-based structure learning, can be used to evaluate the fit of a provided graph to given data. Established is for example the use of the Minimum Description Length [26], the Bayesian Information Criterion [29] and the Bayesian Dirichlet equivalence [14] for the score-based learning of causal graphs [3].

4.2 Evaluations of Metrics for Structure Learning

Several publication investigated the use of metrics using ground truth for the purpose of causal discovery.

De Jongh et al. [8] applied the Greedy Thick Thinning CD algorithm and the Greedy Equivalence Search algorithm on three datasets and inspected the resulting CPDAGs and DAGs. The examined metrics included the number of FN edges, the number of FP edges, the sum of undirected and directed TP edges, the number of TP edges, the number of TP edges with correct orientation, the number of edges with incorrect orientation, a

weighted sum of FN, FP and undirected and directed TP edges, a weighted combination of FN, FP, TP , and finally the Structural Hamming Distance (SHD) as the sum of TP and FN. They identified the SHD as a useful metric, but they also advise against the use of a single metric, as the choice on a single metric leads to information loss.

Constantinou et al. [5] did a general inspection of causal discovery metrics by inspecting and comparing the different types of metrics described above. The inspected metrics such as the TP, FP, TN, FN, Precision, Recall, the F1 measure, the SHD and the DAG Dissimilarity Metric (DDM) [6] and discovered imbalances in the metric scores. The use of metrics based on scoring functions may be biased due to the scoring function itself. Metrics using ground truth are reported to be biased by imbalances in the number of edges or independencies in the ground truth. TP, FP, TN, FN as well as Precision and Recall were reported to be biased as each does not integrate enough TP, FP, TN and FN values sufficiently to be independently meaningful.

Of the described publications, none investigated the use of current methods such as the Precision Recall Curve and the Receiver Operator Curve Area under Curve [22, 20]. Additionally, neither of them investigated metrics for ancestral graphs.

4.3 Metrics for Graph Comparison

In graph theory, there exists related literature regarding the comparison of graphs [11, 32]. Popular is for example, the graph edit distance (GED) metric [1] to compare the fit of two given graphs G_1 and G_2. It is the sum of costs over all required edge and node insertions, substitutions and deletions to transform G_1 into G_2. Since commonly, the nodes are fixed in CD, exact matching algorithms may be used as CD metrics.

As the GED itself is not in use in current CD, it was not part of our investigation. Instead, we investigate the SHD. It is a descendant of the GED, but other than the GED, it only computes the summed costs for the substitution, insertion and deletion of edges, since the nodes are expected to be fixed in CD. Additionally, its costs for each operation are set to one.

5 Metric Evaluation

5.1 Overview of Causal Discovery Metric Criteria

We pose a specific set of criteria on any metric m with a corresponding score s that uses solely the ground true graph G^* to evaluate the goodness of fit to any discovered causal graph G.

Lower and upper bound criterion We prefer s to be bound by maximum and minimum values. While the use of one bound is common to indicate a perfect match between G and G^*, the use of a second bound is helpful for orientation.

G^* identity criterion The criterion of G^* identity assumes if $m(G^*, G^*) = m(G^*, G)$ holds, then $G = G^*$ holds.

TP sensitivity criterion and FP / FN sensitivity criterion Since the true causal graph is assumed to contain only present and absent edges, a metric is required to be sensitive to changes regarding TP and FN edges given the present edges of the ground truth, and also to be sensitive to changes in TN or FP edges regarding the absent edges of the ground truth.

Scoring consistency criterion We try to assure consistent scoring for variations in TP / FN edges and for TN / FP edges by performing the required changes on G and checking for consistent linear changes in scores. For the TP and FP edges, this entails the continuous increase or decrease of absent edges in G and for the TP and FP edges the continuous increase or decrease of directed edges in G. This criterion holds if TP $= 0$. This allows comparability between different 'wrong' learned graphs.

Finally, we prefer metrics to handle any of the currently established graph types to make the causal discovery metrics comparable. This includes DAGs, PDAGs, MAGs and PAGs.

5.2 Investigation on Metric Applicability

To illustrate the applicability of the existing and the novel metric, we demonstrate their use on two example datasets. One is the established Asia dataset [16] with 10.000 data entries with 8 variables each. The second one is the popular Sachs datasets [27] containing 10.000 data entries with 11 variables each. For both, the ground truth is known. Three different CD algorithms were exercised on given datasets. MAGs were excluded as no respective discovery algorithm exists. Again the HCS, the PC Algorithm and the FCI algorithm were employed. All of the examined metrics were calculated for the resulting graphs. The scores were collected for 10 learned graphs each. In Table 2, we depict the resulting score average. We can observe that most metrics are primarily defined for DAGs. Several metrics such as the ROC AUC, PRC AUC, F_1 score, TPR and FPR do not consider undirected edges and only evaluate TP, FP, FN, and TN directed edges even if G^* contains an edge at this location. As PAGs and CPDAGs may contain directed edges as well, these metrics can be transferred without considering the undirected edges. Besides our own metric, only the CPDAG version of the SHD is defined for CPDAGs. As its definition deviates from the SHD for DAGs, we list it in a separate column. In our investigations, we did not find existing custom designed metrics for MAGs and PAGs.

5.3 Experiment on Scoring Consistency and Sensitivity

For this experiment, we considered a generic DAG with $N = 5$ and containing the maximum number of allowed directed edges resulting in seven directed edges and thus seven possible random structure changes. To inspect the metric behavior regarding TP - FN consistency, we stepwise changed the true DAG G^* into an empty DAG G_e randomly eliminating a TP and adding a FN one by one. Regarding consistency in TN / FP direction, we stepwise changed an empty graph G_e into an 'inverse' graph G_i by randomly adding FP directed edges directly opposing the edges of the ground truth. The score results are shown in Figure 6.

We observe in Figure 6a the PRC AUC to be sensitive, but to not perform consistent linear regarding constant changes in TP and FN edges. The zero gradient of the FPR proofs it to not be sensitive to said changes. Regarding sensitivity in TN and FP edges in Figure 6b, the TPR, F1 and PRC AUC show non-responsive by having a gradient of zero.The only metrics sensitive for changes in both individual trials show to be the ROC AUC, the SHD and the nCED.

5.4 Elaboration on Metric Bounds and Normalization

To inspect the bounds of each metric, we inspected the definition of the metrics. The overview is presented in Table 3. The SHD metric only comes without an upper bound.

Table 2: Example on how the examined metrics perform in an actual application scenario. The higher the scores of the ROC AUC, PRC AUC, F_1, TPR and FPR metrics, the better performed the causal structure learning algorithm. For each SHD and nCED variant, lower scores are better.

(a) Asia dataset

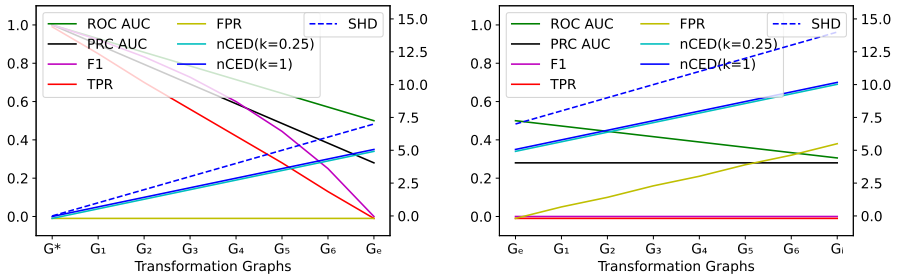
	ROC AUC	**PRC AUC**	**F_1 score**	**TPR**	**FPR**	**SHD** (DAG)	**SHD** (CPDAG)	**nCED** ($k = 0.2$)	**nCED** ($k = 0.4$)
HCS	0.62	0.19	0.33	0.38	0.12	12	-	0.20	0.20
PC	0.47 *	0.13 *	0.06 *	0.06 *	0.12 *	-	11	0.11	0.15
FCI	0.49 *	0.14 *	0.12 *	0.12 *	0.13 *	-	-	0.11	0.15

* The metric only considers present and absent directed edges, unaware of other edges types.

(b) Sachs dataset

	ROC AUC	**PRC AUC**	**F_1 score**	**TPR**	**FPR**	**SHD** (DAG)	**SHD** (CPDAG)	**nCED** ($k = 0.2$)	**nCED** ($k = 0.4$)
HCS	0.72	0.33	0.51	0.53	0.09	17	-	0.21	0.21
PC	0.47 *	0.13 *	0.06 *	0.06 *	0.12 *	-	10	0.22	0.26
FCI	0.57 *	0.16 *	0.25 *	0.29 *	0.16 *	-	-	0.23	0.26

* The metric only considers present and absent directed edges, unaware of other edges types.



(a) Stepwise transformation of TP edges of G^* to FN

(b) Stepwise transformation of TN edges in G_e to FP

Figure 6: Overview of the metrics behavior given the sequential use of random structure changes on a five variable DAG. The scale of the SHD is shown on the plots right side. We observe some metrics to have a gradient of zero and thereby to be not responsive for the induced changes. Of the responsive metrics, we observe most perform consistently indicated by a linear increase or decrease.

All other metrics come with an upper and lower bound and are therefore considered to be normalized. While most metrics use sums of FN, TP, FP or TN, but never all of them together, the nCED uses the number of graph variables for normalization. Mind, that $N = \text{TP} + \text{TN} + \text{FN} + \text{FP}$ holds and that is why the nCED considers each for normalization.

Table 3: Bounds and normalizability of the metrics described in Section 2.2

	Lower bound	**Upper bound**	**Normalization**	**Normalizing factor**
ROC AUC	✓	✓	✓	TP + FN and FP + TN
PRC AUC	✓	✓	✓	TP + TN and FP + TP
F_1 score	✓	✓	✓	$\frac{TP}{TP+TN}$ and $\frac{TP}{FP+TP}$
TPR	✓	✓	✓	TP + FN
FPR	✓	✓	✓	FP + TN
SHD (DAG)	✓	✗	✗	-
SHD (CPDAG)	✓	✗	✗	-
nCED (ours)	✓	✓	✓	$N(N-1)$

5.5 Elaboration on G^* identity

The criterion states that besides G^* no other graph G should exists for which the inspected metric evaluates $m(G, G^*) = m(G^*, G^*)$. For this investigation, we inspected if we were able to find evidences for each metric contradicting the criterion.

SHD for CPDAGs transforms the ground truth DAG to its CPDAG representation. This transformation is commonly surjective as unambiguous edge orientations are removed from the graph and thus several DAGs can share the same CPDAG representation. This is why the G^* identity criterion does not hold for the DAGs sharing the CPDAG representation with G^*.

Several metrics, such as the F1 score, PRC AUC, ROC AUC, TPR and FPR, do not consider undirected edges and they count them as absent edges. This is why the criterion does not hold for graphs identical to G^*, but with undirected edges instead of TN edges.

For $k = 0$, the nCED metric evaluates undirected edges like TP edges. Thus the criterion does not hold for graphs with arbitrary undirected TP and TP edges.

6 Conclusion

In this paper, we performed several inspections on established metrics for the use with ground truths. We proposed a new graph notation and a novel metric called the normalized Causal Edit Distance (nCED) that allows benchmarking of MAGs, PAGs, CPDAGs and DAGs. Additionally, we investigated if current causal discovery metrics fulfill a proposed set of criteria and found several peculiarities which should be taken into account before their application. Also, we found deficiencies in the applicability of current metrics as many perform on DAGs, but few perform on CPDAGs and even less on PAGs.

7 Acknowledgements

This work benefited from the Dagstuhl seminar 24031 "Fusing Causality, Reasoning and Learning for Fault Management and Diagnosis".

References

[1] Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*, 2015.

[2] S. Acid and L. M. de Campos. Searching for bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490, 2003.

[3] R. R. Bouckaert. *Bayesian belief networks: from construction to inference*. PhD thesis, 1995.

[4] L. Cheng, R. Guo, R. Moraffah, P. Sheth, K. S. Candan, and H. Liu. Evaluation methods and measures for causal learning algorithms. *IEEE Transactions on Artificial Intelligence*, 3(6):924–943, 2022.

[5] A. C. Constantinou. Evaluating structure learning algorithms with a balanced scoring function. *arXiv preprint arXiv:1905.12666*, 2019.

[6] A. C. Constantinou, N. Fenton, and M. Neil. How do some bayesian network machine learned graphs compare to causal knowledge? *arXiv preprint arXiv:2101.10461*, 2021.

[7] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[8] M. de Jongh and M. J. Druzdzel. A comparison of structural distance measures for causal bayesian network models. *Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series*, pages 443–456, 2009.

[9] C. C. Emezue, A. Drouin, T. Deleu, S. Bauer, and Y. Bengio. Benchmarking bayesian causal discovery methods for downstream treatment effect estimation. In *ICML 2023 Workshop on Structured Probabilistic Inference and Generative Modeling*, 2023.

[10] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine learning*, 31(1):1–38, 2004.

[11] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010.

[12] C. Glymour, K. Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 10:524, 2019.

[13] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson. Further thoughts on precision. In *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, pages 129–133. IET, 2011.

[14] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20:197–243, 1995.

[15] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[16] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

[17] C. Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of UAI 1995*, pages 403–410, 1995.

[18] G. Menegozzo, D. Dall'Alba, and P. Fiorini. Cipcad-bench: continuous industrial process datasets for benchmarking causal discovery methods. In *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, pages 2124–2131. IEEE, 2022.

[19] J. Pearl. *Causality*. Cambridge university press, 2009.

[20] A. Pérez-Suay and G. Camps-Valls. Causal inference in geoscience and remote sensing from observational data. *IEEE Transactions on Geoscience and Remote Sensing*, 57(3):1502–1513, 2018.

[21] W. Peterson, T. Birdsall, and W. Fox. The theory of signal detectability. *Transactions of the IRE professional group on information theory*, 4(4):171–212, 1954.

[22] S. R. Pfohl, T. Duan, D. Y. Ding, and N. H. Shah. Counterfactual reasoning for fair clinical risk prediction. In *Machine Learning for Healthcare Conference*, pages 325–358. PMLR, 2019.

[23] T. Richardson and P. Spirtes. Ancestral graph markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

[24] T. S. Richardson. *Models of feedback: interpretation and discovery*. PhD thesis, Carnegie-Mellon University, 1996.

[25] T. S. Richardson and P. Spirtes. Causal inference via ancestral graph models. *Oxford Statistical Science Series*, pages 83–105, 2003.

[26] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[27] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[28] Y. Sasaki et al. The truth of the f-measure. *Teach tutor mater*, 1(5):1–5, 2007.

[29] G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[30] P. Spirtes, C. N. Glymour, R. Scheines, and D. Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[31] I. Tsamardinos, L. E. Brown, and C. F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[32] P. Wills and F. G. Meyer. Metrics for graph comparison: a practitioner's guide. *Plos one*, 15(2):e0228728, 2020.

[33] J. Zhang. Causal reasoning with ancestral graphs. *Journal of Machine Learning Research*, 9:1437–1474, 2008.

[34] J. Zhang and P. Spirtes. Detection of unfaithfulness and robust causal inference. *Minds and Machines*, 18(2):239–271, 2008.

RETROFITTING CYBER-PHYSICAL PRODUCTION SYSTEMS WITH RADIO-BASED SENSORS AND ML

Christian Kühnert, Steffen Wallner, Lars Wessels, Andreas Wunsch and Mathias Ziebarth
Fraunhofer IOSB
76131 Karlsruhe, Germany
christian.kuehnert@iosb.fraunhofer.de
steffen.wallner@iosb.fraunhofer.de
lars.wessels@iosb.fraunhofer.de
andreas.wunsch@iosb.fraunhofer.de
mathias.ziebarth@iosb.fraunhofer.de

ABSTRACT

Manufacturing companies usually isolate their production networks from other networks to ensure security against external attacks and to guarantee a fail-safe 24/7 operational service. However, these measures make it technically and organizationally complex to install new sensors or deploy new software in the production process. As a result, machine learning is only used to a limited extent in manufacturing, as these models require regular adaptations. To tackle this challenge, one possible solution is to install an additional network that is not connected to the production network. This network can be utilized for rapid prototyping of new sensors, advanced data analysis, or the deployment of machine learning models. One possible solution is to install a radio-based low-power, long-range network, having the property to capture data over large distances with only little power consumption. This paper examines the potential of retrofitting cyber-physical systems with such a network in combination with machine learning methods. The results are evaluated through three practical use cases: monitoring a workspace with a molding machine, monitoring the cycles of a washing machine, and predicting the daily consumption profile of a main water pipeline.

Keywords LoRaWAN · Machine Learning · Time-Series · Cyber-Physical Systems

1 Introduction

Manufacturing companies typically have multiple IT networks, with the production network being particularly important. This network involves critical business procedures

that are mainly related to the production process. Therefore, it is essential to ensure a secure and fail-safe operating environment for this network. However, such protection also has disadvantages. Integrating new sensors and actuators on both hardware and software fronts, collecting data, or simply updating software components can quickly become time-consuming and utterly complex. Attempting to add machine learning (ML) models to the manufacturing process, which usually require regular updates, can result in even higher levels of technical and personnel effort, making it in some cases even impractical. In summary, there is a conflict between production networks that require 24/7 availability and enhanced security and the concept of rapid prototyping to explore possibilities of process improvements or the concept of continuous integration to install and update new software components.
A possible solution to this issue is to set up an additional wireless network that is distinct from the production network itself. This additional network can be used to quickly connect new sensors to existing equipment, as well as to collect and evaluate their data. Furthermore, the measurement data can be easily analyzed and processed with advanced data analysis approaches such as ML models.
Additionally, in many cases high-resolution data is not necessary and data at a minute-level interval or even lower is largely sufficient. This has also the advantage that sensors can function on battery power for a prolonged duration, negating the need for an external power supply, and reduce the amount of cabling in the system. This means, that sensors can be installed in so-called deploy and forget scenarios. A device is installed and requires minimal to no maintenance, allowing it to operate autonomously over a long period without the need for regular human intervention. Those characteristics and having the ability to cover longer distances than conventional WiFi, expands the application range further. Examplary applications in industry are the tracking of assets [26], environmental monitoring [27, 28], structural health monitoring [29] or the integratino into the supply chain management [30].

In summary, there exists a research gap in examining the integration of radio-based sensors with advanced data analytics and machine learning. This integration could potentially lead to the quick generation of new process knowledge and enable the fast development of applications. A frequently used low-power long-range network type, which is evaluated in this publication, is LoRaWan [7].
The main research question in this publication is to validate the suitability of LoRaWAN in combination with ML for the production domain. The paper is structured as follows: Section 2 summarizes the state-of-the-art of LoRaWAN with ML. Section 3 outlines the ML-workflow been set up and in section 4 three practical use cases are explained and thoroughly evaluated. Finally, a summary is provided along with future prospects.

2 LoRaWAN Network

LoRaWAN, stands for Long Range Wide Area Network and is designed for radio-based long-range communication between low-power devices. Usually, it is used in the context of the Internet of Things (IoT) where devices usually send or receive information with lower sample rates compared to a production process. The network structure of LoRaWAN is based on a star-shaped topology as illustrated in Fig. 1 and more specifically, the network consists of the following four components:

- Devices: End devices that perform the final measurement. They are wirelessly connected to the gateways, often used batteries for power supply and can be placed anywhere. They provide measurements or receive information within a specified sampling interval, typically minutes, hours, or days.

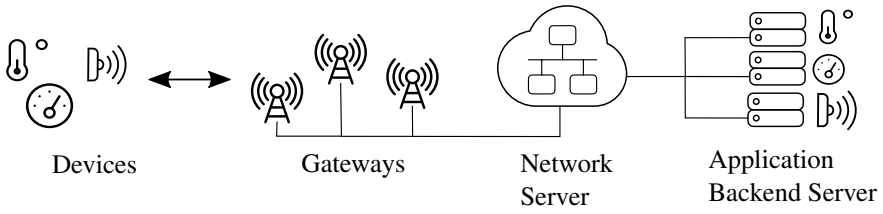


Figure 1: LoRaWAN network topology with its components.

- Gateways: Collect the measurements from the devices and transmit them to the network server. Transmission to the network server is typically done via WiFi or Ethernet.
- Network Server: Decodes the packets sent by the devices, performs security checks and forwards the packets to the application server.
- Application Server: Receives the measurements and contains the decision logic for what to do with the information. This server contains the tools for the ML-workflow described in section 3.

Comparison and Core Features: Compared to other radio-based networks like 5G [32] or Wi-Fi [31], LoRaWAN covers much greater distances but at the cost of lower data rates, meaning that LoRaWAN is not suitable for applications that require the transmission of large amounts of data. Against other low-power wide-area systems like SigFox [33], LoRaWAN provides more flexibility in network deployment, since it allows public and private network setups. Compared to NB-IoT [34], LoRaWan does not rely on existing cellular network infrastructure. Figure 2, shows a comparison of different technologies in terms of their bandwidth and range.

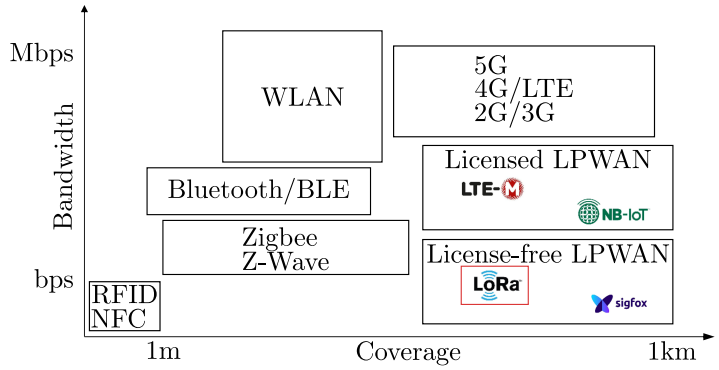


Figure 2: Application areas depending on bandwidth and coverage of various radio-based Networks.

In summary, the aim of LoRaWan is to transmit data over very long distances while keeping the power consumption of the device as low as possible. Since many LoRaWAN devices are battery-operated and having the aim of running over several years, the frequency of transmitted messages needs to be kept as low as possible. To give an impression how LoRaWan is operating, table 1 summarizes the key features [3].

Table 1: LoRaWAN core features according to [3].

Characteristic	LoRaWAN
Data rate	290 bps - 50 kbps
Battery lifetime	8-10 years
Power efficiency	Very high
Security	Yes (AES)
Range	2-5 km urban, 15 km suburban, 45 km rural

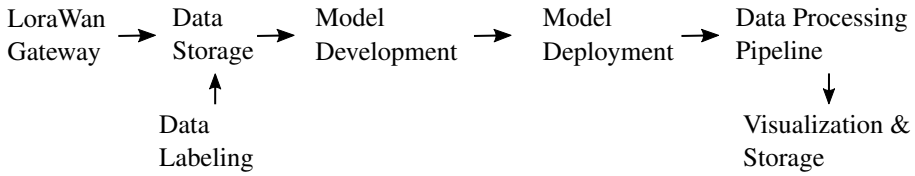


Figure 3: Used ML-workflow for the use cases running on the application back-end server shown in section 2 .

2.1 State-of-the-Art: Applications of LoRaWAN and Machine-Learning for Cyber-Physical Production Systems

In most cases, LoRaWAN is used in the IoT context, while the smart city context is one of its main applications. To have the ability to collect measurements over a long distance while needing only a low bandwidth for data collection, makes the smart city context a particularly suitable domain. Several scenarios have therefore already been analysed in the literature. Examples are the measurement of air quality [22] (7 devices, 2 min sample time), the implementation of a device tracking [18] (1 device, maximum 30 sec sample time) in a city in the Netherlands or the monitoring of the urban lighting infrastructure of a town in Italy [16] (1 device, 8 sec sample rate). Another example is the application for precision irrigation in the agricultural sector [25] (8 devices, 1 min sample time). None of the cited literature sources mentioned there were any issues with the sampling rate.
With regard to the combination of LoRaWAN and ML, far less literature is available. Some recent studies exist, which, however do not target production systems. A recent example can be found in [5], which investigates how ML can be used to improve the resource management of LoRaWAN. In [17] (unknown sample time) LoRaWAN is used with 24 gateways and in combination with ML for indoor localization. Within [23] (3 min sample time) a solution is presented how to monitor the behavior of pig movements using LoRaWAN and ML for clustering. As a side note it needs to be mentioned that the authors were able to increase battery life by a factor of three by running the ML model directly on the edge device. Less data had to be transmitted between the device and the gateway. Finally, [1] presents results on the application of ML with LoRaWAN devices to detect human presence and estimate the number of occupants in an office building. These examples already demonstrate how radio-based sensors can assist to integrate ML into CPS systems. This paper takes it a step further by establishing a complete MLOps environment. The results are demonstrated across multiple use cases.

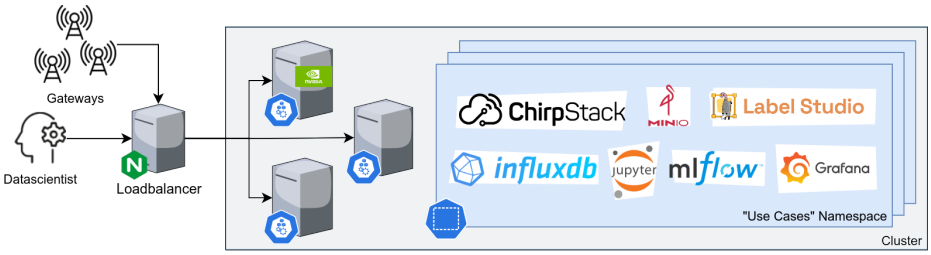


Figure 4: User and gateways access the services centralized via a load balancer.

3 ML-Workflow and Infrastructure

For the investigated use cases in section 4, ML means much more than just training and validating a model. Depending on the use case, the ML model needs to be deployed after training, it needs to run on a cyclic basis repeatedly and results need to be provided on a UI or written back to a database. Therefore, an IT-infrastructure is set-up which has the capability to implement complete ML workflows. This includes the labeling of measurements, the data analysis, model development, model deployment, cyclic execution of complete data processing pipelines, and finally the visualization of the results and the possibility to write them back into a database. The complete ML-workflow is sketched in Fig. 3. To set-up the workflow, the following libraries and tools were used: An InfluxDB for data storage [9], Jupyter Lab [10] for model development, MLflow [15] for model deployment with MinIO [14] as object storage. For the deployment of the entire data processing pipeline, MageAI [13] is utilized and visualization is done in Grafana [8]. The labeling of the measurements is performed using Label Studio [12]. Grafana and InfluxDB were chosen as the platforms of choice due to the exclusive capture of time series data in the use cases. Given the low sampling rates, polling can be used to query data from the InfluxDB, therefore MageAI was the tool of choice. The project Jupyter is the de facto standard for developing ML models, while MLflow was selected for model management due to its ease of use. Finally, Label Studio was choosen for data annotation as this is one of the few freely available tools in this area.

Because of the large number of tools involved, the services relevant to the workflow are hosted in containers. The containers are operated in a Kubernetes cluster [11] for efficient sharing of compute power (CPU and GPU), memory, and storage resources. As an orchestration engine, Kubernetes takes care of the optimal distribution of the containers to the available nodes. What is relevant for the use cases is that the currently scarce GPU resources can be shared without any problems. The services are executed separately from other workloads in their own namespace, even if they are assigned to different nodes. Figure 4 shows the distributed services for the use cases and how they are accessed.

4 Investigated Scenarios

In the following, within three use cases the existing IT-infrastructure has been extended with LoRaWAN, one or several new sensors have been installed and advanced data analysis like ML has been used on the collected data. It must be noticed, that the use cases do not describe the ML algorithms in detail, but rather how the task was solved using them. The relevant machine-learning libraries have been referenced within the described scenario.

Figure 5: Multisensor installed on injection molding machine (left); dashboard with results of ML model (right)

4.1 Scenario 1: Runtime Monitoring of an Injection Molding System

The initial use case concerns a process monitoring task, specifically the runtime monitoring of an injection molding machine. The objective is to determine whether the molding machine is running or not, and whether the workspace is currently occupied or not. The results are displayed on a dashboard, while the dashboard should be automatically updated when changes are made.

Used Sensor: A multisensor platform, as depicted in Figure 5, is glued to the molding machine. This sensor measures both, the current surrounding noise level and the current light intensity. In a subsequent step the noise level is utilized to determine whether the machine is running or not by assessing if the noise is high or low. The level of light intensity is exploited to ascertain whether the workspace is currently occupied or not. It is assumed that the light intensity is high when the workstation is occupied and low when no one is with the molding machine. Regarding the noise sensor, the preprocessing is performed directly on the multisensor platform by constantly applying an exponential moving average filter over the last three minutes of data. The value of the light sensor is not preprocessed and therefore transmitted directly. The data transmission interval for both measurements in this use case is 60 sec.

ML-Tools and Workflow: For this use case, all tools from the described ML workflow in Fig. 3 are required. In an initial step, data must be labeled regarding the following four classes: (1) Molding machine is running, workspace is occupied; (2) Molding machine is running, workspace is not occupied; (3) Molding machine stopped, workspace is occupied; (4) Molding machine stopped, workspace is not occupied. The next step covers the training of the ML model using a classic decision tree from the scikit-learn library [21]. Finally, the decision tree is integrated into the cyclically triggered data processing pipeline, which checks every 30 sec for new data and performs a classification. The results of the classification are written back to a database and visualized on a dashboard. Please refer to Fig. 5 for the resulting dashboard.

Results: Of course, the task could have also been solved using two thresholds, one for the noise and one for the light sensor. However, the aim was to test the entire ML workflow in conjunction with LoRaWan. Regarding LoRaWan, it can be stated that all measurement data was transferred to the application server without any failures. The server timestamp written in the time series database has fluctuations in the sampling rate

of less than one second, which is sufficiently tolerable for this scenario. The ML task was extremely simple in this scenario, and misclassifications only occurred e.g. when employees forgot to turn off the lights. Regarding the sampling rate of $60\ \mathrm{sec}$ and the ML pipeline which ran every $30\ \mathrm{sec}$, it can take up to $90\ \mathrm{sec}$ for a change to become visible on the dashboard. The results showed that this is not immediately intuitive for the employee and one must first get used to the possibly longer latency times.

4.2 Scenario 2: Washing Cycle Counting with Acoustic Sensor

In this use case, a washing machine has an attached acoustic sensor shown in Fig. 6 in the upper right plot. The objective is to determine the number of the washing machine's cycles given a certain time frame. The challenge is that the washing machine is located in a living space, which means that in addition to the sounds of the washing machine, there are also conversations, doors being opened and closed, and a radio in the room that is occasionally turned on. Furthermore, the duration of a washing cycle depends on the load of the machine.

Used Sensor: The acoustic sensor is mounted on the wall next to the washing machine and measures the noise level once per second. In a next step the noise level is smoothed using a simple moving average filter with a time constant of $t = 15\ \mathrm{sec}$. Once per minute, the device transmits the current noise level to the gateway. In this scenario, data has been collected for about two weeks.

ML-Workflow and Tools: In terms of the ML-workflow, data storage, data labeling and the developed model are required. Data labeling is not mandatory, but in that case, it is used to evaluate the performance of the ML model. The upper left plot in figure 6 shows the signature of a washing cycle. Depending on the load of the machine, the duration of a wash cycle is a few minutes longer or shorter, but on average takes about $75\ \mathrm{min}$. As ML algorithm Dynamic Time Warping (DTW) [24, 20] from the Python package *pyts* [4] is used to find the signature of a washing cycle within the recorded time series of the acoustic sensor. For DTW, the used distance measure is the squared distance. Fig. 6 shows the results of the classification. In summary, a total of six washing cycles were performed during the period, of which five were found by the ML model.

Results: In this scenario, the focus was on two questions. The first question was whether the noise sensor could measure the signature of the washing process, given the slow sampling rate and the amplitude of the measured noise level. The second question was, whether the data quality was sufficient to use ML, in this case dynamic time warping, to detect the washing cycles. The results showed that both questions were successfully answered. However, it is important to note that due to the low sampling rate of $1\ \mathrm{min}$, the measured values must be used in the raw format. With respect to the sampling rate, a pre-filtering, or any other data-processing steps to improve data quality did not bring any improvement.

4.3 Scenario 3: Predicting Drinking Water Consumption

In the third scenario, a pulse generator was added to an old water meter in a water distribution network in southern Germany, as shown in Fig. 7 on the left. There are two purposes to upgrade the traditional water meter to a smart meter: (1) to facilitate reading, as someone had to previously climb into the shaft to read the water meter, and (2) to obtain information about current flow rates through the pipe on an hourly basis. Regarding the ML task, this information is utilized to forecast the water flow through

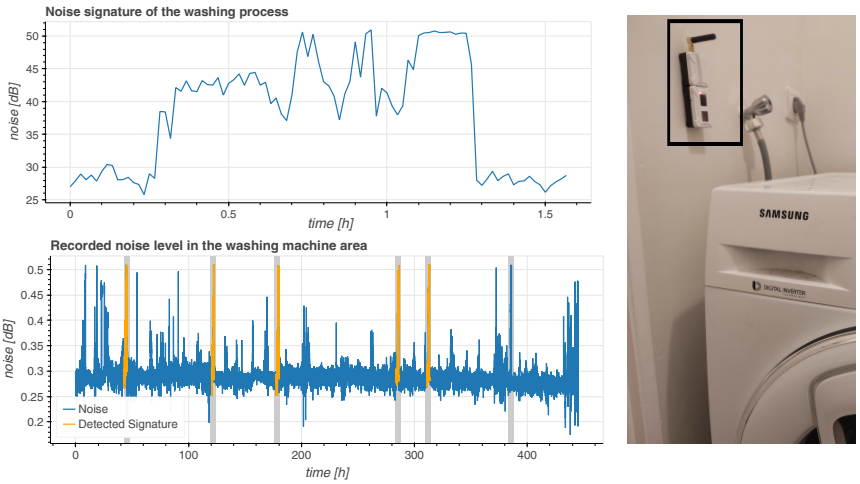


Figure 6: Signature of a washing cycle (up left), installed acoustic sensor close to the washing machine (up right), detected signatures through DTW (down left). Labeled washing cycles are shown in gray.

the pipe for the next 24 hours. This information is crucial for water works as it can be used to facilitate energy-saving efforts by synchronizing pumping activities with times of low energy prices.

Used Sensor: As mentioned, an old water meter has be upgraded by a pulse generator to measure the flow rate. The measurements of the pulse generator are sent to the gateway at a sampling rate of 15 $\mathrm{minutes}$. In this application, the pulse generator is configured to generate one pulse per cubic meter of water. As there is no electricity in the area of the water meter, the pulse generator is powered by a battery.

ML-Workflow and Tools: The main task of the ML model is the prediction of the water consumption for the next 24 hours. The results of the prediction are to be written back to a database so that both, the operator of the water work, as well as the control system can access them. Regarding the ML workflow described in section 3 all tools except the dashboard are needed. To predict the drinking water consumption, the NHiTS algorithm [2] within the *PyTorch Forecasting 1.0.0* [19] library is used. NHiTS is a neural network model based on several multi layer perceptrons containing ReLU nonlinearities. Even though the installed LoRaWAN sensor transmits higher resolution data, as a preprocessing step data is aggregated to an hourly basis to slightly reduce computation time. Fig. 7 shows the results for a 14-day period in April 2023. Besides the mean prediction for each time step (red), the figure shows the 80 $\%$ prediction interval (yellow) between the first and ninth percentile. Compared to a baseline model which is based on 7-day lagged observed data (dashed blue line) resulting in a mean-squared-error of 3.23, the NHiTS model performs reasonably better with a mean-squared-error of 1.27.

Results: The results of the scenario can be interpreted in many ways. First of all, it shows that there exist meaningful scenarios where sensors must be battery-operated due to the lack of a power supply. It is worth noting that during the evaluation period of over

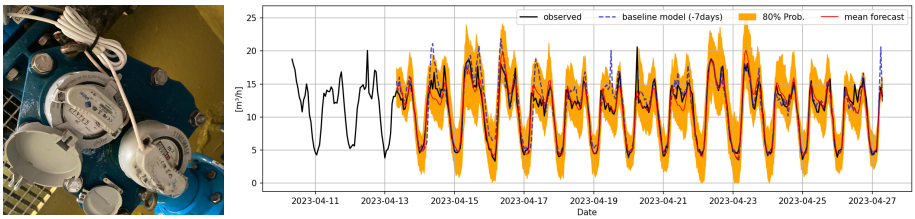


Figure 7: Installed pulse generator (left). Prediction of hourly drinking water consumption using in comparison with a 7-day lagged baseline model (right).

a year, there were no issues with either power supply or data transmission. Hence, the installation of the LoRaWAN pulse generator reduced employee workload and achieved a higher sampling rate of measured flow rates. Regarding the ML task, the NHiTS model achieved good results in forecasting the water consumption profile for the next 24 hours. By using the entire ML workflow, the model training and deployment was easy and quick.

5 Conclusion and Future Prospects

Production networks are typically isolated from other networks to ensure security and to enable fail-safe operation. However, this means that the integration of new sensors or deploying complete data processing pipelines can quickly become complex and time-consuming. To address this issue, the publication assessed the effectiveness of incorporating an extra radio-based network based on LoRaWAN, along with advanced data analytics and ML. The concept was investigated using the three use cases, monitoring of a molding machine, recognizing the signature of washing machine cycles, and predicting the water consumption profile on a water distribution network over a 24-hour period. All use cases demonstrated how the combination of radio-based technology and ML can bring added value for the underlying process. The main challenge in implementing the use cases was that the three domains—(1) sensor installation, (2) data science, and (3) MLOps—had to be integrated into one structure to operationalize the application.
There are many prospects for further research in this field. Regarding ML and data processing, it may be advantageous to conduct more data processing directly on the sensor. For example, sensors could analyze data on the edge and only send extracted features, intermediate results, or final results to the gateway. This approach has already been partially implemented in the first two use cases and has produced satisfactory results. Finally, the range of applications could be extended. Particularly with respect to environmental monitoring, there are many application areas. For instance, the cause of the fish die-off in the Oder River [35] could have been identified more quickly. Additionally, LoRaWAN sensors could have been used as an early warning system during the flood disaster in the Ahr Valley [36]. These are all areas, in which a fast sample time is not important.

References

[1] Adeogun R., Rodriguez I., Razzaghpour M., Berardinelli G., Christensen P.H., and P. E. Mogensen. In: Indoor Occupancy Detection and Estimation using Machine Learning and Measurements from an IoT LoRa-based Monitoring System, Global IoT Summit (GIoTS), pp. 1-5, doi: 10.1109/GIOTS.2019.8766374, 2019

[2] Challu C., Olivares K. G., Oreshkin B. N., Garza F., Mergenthaler-Canseco M., Dubrawski A.: N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting, arXiv, doi: 10.48550/arXiv.2201.12886, 2022

[3] de Carvalho Silva J., Rodrigues J., Albert A., Solic P., Aquino A.: LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities. In: 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech), pp. 1–6., 2017

[4] Faouzi J., Janati H., pyts: A python package for time series classification, Journal of Machine Learning Research, **21**(46), pp. 1-6, url: `http://jmlr.org/papers/v21/19-763.html`, 2020

[5] Farhad A., Pyun J.Y.: LoRaWAN Meets ML: A Survey on Enhancing Performance with Machine Learning. In: Sensor **15**(23), MDPI, doi: 10.3390/s23156851, 2023

[6] Kleppmann N., Hartung B., Bernard T., Theiss J.: ML4Heat: Betriebsoptimierung von Fernwärmenetzwerken. In: atp Magazin, **10**, pp. 28-31, 2021

[7] LoRa Alliance, LoRaWAN Specification, 2016, 70 p,`https://lora-alliance.org/lorawan-for-developers/`. Last accessed 20 Dec 2023

[8] Grafana, `https://grafana.com`. Last accessed 20 Dec 2023

[9] Influx data, `https://www.influxdata.com`. Last accessed 20 Dec 2023

[10] Jupyter-Lab, `https://jupyter.org`. Last accessed 20 Dec 2023

[11] Jupyter-Lab, `https://github.com/kubernetes/kubernetes`. Last accessed 20 Dec 2023

[12] Label-Studio, `https://labelstud.io`. Last access 20 Dec 2023

[13] MageAI, `https://www.mage.ai`. Last accessed 20 Dec 2023

[14] MinIO, `https://min.io`. Last accessed 20 Dec 2023

[15] ML-Flow, `https://mlflow.org`. Last accessed 20 Dec 2023

[16] Pasolini G., Buratti C., Feltrin L., Zabini F., De Castro C., Verdone R., Andrisano O.: Smart City Pilot Projects Using LoRa and IEEE802.15.4 Technologies, Sensors, 18(4), p 1118, doi: 10.3390/s18041118, 2018

[17] Perković T., Dujić R. L., Šabić, J., Šolić P.: Machine Learning Approach towards LoRaWAN Indoor Localization. In: Electronics **2**, MDPI, doi:10.3390/electronics12020457, 2023

[18] Podevijn N., Plets D., Trogh J., Martens L., Suanet P., Hendrikse K.: Joseph W.: TDoA-Based Outdoor Positioning with Tracking Algorithm in a Public LoRa Network, Wireless Communications and Mobile Computing 2018, 1864209, 2018

[19] PyTorch-Forecasting `https://github.com/jdb78/pytorch-forecasting`, last access 04 Dec 2023

[20] Sakoe H., Chiba S.: Dynamic programming algorithm optimization for spoken word recognition, in IEEE Transactions on Acoustics, Speech, and Signal Processing, **26**(1), pp. 43-49, doi: 10.1109/TASSP.1978.1163055, 1978

[21] Scikit-Learn, `https://scikit-learn.org`. Last access 20 Dec 2023

[22] Simo A., Dzitac S. , Dzitac I., Frigura-Iliasa M., Mihai F.: Air quality assessment system based on self-driven drone and LoRaWAN network. In:Computer Communications, v. 175, pp. 13-24, doi: 10.1016/j.comcom.2021.04.032, 2021

[23] Suresh, V. M., Sidhu R., Karkare P., Patil A., Lei Z., Basu A.: Powering the IoT through embedded machine learning and LoRa. In:4th World Forum on Internet of Things (WF-IoT), pp. 349-354, doi: 10.1109/WF-IoT.2018.8355177, 2018

[24] Vintsyuk T.K.: Speech discrimination by dynamic programming. Cybern Syst Anal 4, 52-57, doi: 10.1007/BF01074755, 1968

[25] Zhang H., He L., Di Gioia F., Choi D., Elia A., Heinemann P.: LoRaWAN based internet of things (IoT) system for precision irrigation in plasticulture fresh-market tomato, In: Smart Agricultural Technology **2**, doi: 10.1016/j.atech.2022.100053, 2022

[26] Priyanta I. F., Golatowski F., Schulz T. and Timmermann D.: Evaluation of LoRa Technology for Vehicle and Asset Tracking in Smart Harbors, In: IECON 2019, doi: 10.1109/IECON.2019.8927566, 2019

[27] Bates, H.; Pierce, M.; Benter, A.: Real-Time Environmental Monitoring for Aquaculture Using a LoRaWAN-Based IoT Sensor Network, In: Sensors 21, doi: 10.3390/s21237963, 2021

[28] Sobhi, S.; Elzanaty, A.; Selim, M.Y.; Ghuniem, A.M.; Abdelkader, M.F.: Mobility of LoRaWAN Gateways for Efficient Environmental Monitoring in Pristine Sites, In: Sensors 23, doi: 10.3390/s23031698, 2023

[29] Polonelli T., Brunelli D., Guermandi M. and Benini L.: An accurate low-cost Crackmeter with LoRaWAN communication and energy harvesting capability, In: Emerging Technologies and Factory Automation, pp. 671-676, doi: 10.1109/ETFA.2018.8502592, 2018

[30] Garrido-Hidalgo C., Olivares T., Ramirez F., Roda-Sanchez L.: An end-to-end Internet of Things solution for Reverse Supply Chain Management in Industry 4.0, Computers in Industry, 112, doi: 10.1016/j.compind.2019.103127, 2019

[31] IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications., IEEE-SA. 14 December 2016. doi:10.1109/IEEESTD.2016.7786995, 2016

[32] Pirinen P.: A brief overview of 5G research activities, 1st International Conference on 5G for Ubiquitous Connectivity, doi: 10.4108/icst.5gu.2014.258061, 2014

[33] Mikhaylov K., et al.: Communication Performance of a Real-Life Wide-Area Low-Power Network Based on Sigfox Technology, ICC, doi: 10.1109/ICC40277.2020.9148645, 2020

[34] Xu J., Yao J., Wang L., Ming Z., Wu K. and Chen L.: Narrowband Internet of Things: Evolutions, Technologies, and Open Issues, IEEE Internet of Things Journal, 5, doi: 10.1109/JIOT.2017.2783374., 2018

[35] https://www.dw.com/en/could-a-mass-fish-die-off-in-the-oder-river-happen-again/a-65971681, last access 15 Feb 2023

[36] https://www.dw.com/en/opinion-germanys-ahr-valley-flood-disaster-management-is-the-next-tragedy/a-62461022, last access 15 Feb 2023

XAI FOR ANOMALY ANALYSIS BY POWER PLANT OPERATORS - A CASE AND USER STUDY

Marcel Dix[1], **Jan Koltermann**[2], **Sebastian Mieck**[2], **Boris Pastler**[1], **and Benjamin Kloepper**[1,3]
[1]Industrial AI, ABB AG Corporate Research Center, Germany
[2]Power Plant Analytics, LEAG Lausitz Energy Power Plants AG, Germany
[3]Intelligent Industry, Capgemini Invent, Germany

ABSTRACT

Detecting and analyzing abnormal situations in power plants is a crucial task for plant operators. This paper explores the potential of Explainable AI (XAI) techniques in supporting operators by providing comprehensive insights into the origins and causes of anomalies. A case study was conducted at the LEAG power plant in Boxberg, Germany — one of Europe's largest power plants. The study utilized a dashboard developed in this study to present AI-predicted anomalies to operators. To enhance the plausibility of anomalies and understanding of their root causes, the dashboard incorporates five basic explanatory techniques from the XAI literature: case-based explanations, feature importance, counterfactual explanations, contextualization, and natural language explanations. The case study includes a user evaluation with power plant operators from LEAG to assess the effectiveness of XAI techniques in enhancing operators' analysis and understanding of abnormal situations. The results confirm the value of these techniques, and their implications are discussed in this paper.

Keywords AI, Explainable AI (XAI), Anomaly Detection, Power Plant, Operator

1 Introduction

The growing trend of digitalization in power plants results in the continuous increase of vast amounts of data — primarily time series data from hundreds of digital sensors in the plant. The volume of historical production data being processed and stored can easily accumulate to several terabytes. All this data represents a great potential for novel AI use cases that could contribute to more efficient and safer operation of power plants. Here, a relevant AI use case often considered in literature is to assist plant operators in the detection of abnormal situations in the plant [1, 2]. Once an abnormal situation is detected, the operator must investigate the situation, identify its cause, and determine an appropriate course of action in response to the situation. This part of the operator's job is particularly challenging: It requires the observation of dozens or even hundreds of signals spread across multiple plant sections and operator screens, and relies heavily on the operator's personal experience and current mental freshness.

An important limitation of AI is that many machine learning models lack transparency to explain why the model reached a particular prediction [3]. To overcome this limitation, the research field “Explainable AI” (XAI) [4, 5] has received increasing attention also in the industrial domain recently. This paper presents a case and user study of how XAI can support the evaluation, analysis, and understanding of potentially abnormal situations detected by a machine learning-based anomaly detection method. This approach goes beyond the common use of XAI to provide insight into the reasoning process of machine learning models, but investigates whether the XAI output also provides valuable insight into the current process conditions. For this study, a dashboard has been developed at ABB Corporate Research Germany, as part of the EU-funded research project EXPLAIN [6], and in collaboration with LEAG. In the provided user scenario, the dashboard serves as an AI-based operator support system, e.g. as an additional operator screen in the control room, where anomalies, detected by an underlying anomaly detection model, are displayed, and the model output is explained to the operator. The explanation of anomalies is realized with the help of serveral visual *explainers* that have been developed in this study, based on five underlying XAI techniques from literature, namely: showing similar anomalies (case-based explanations), showing the signal contributions to an anomaly (feature importance), showing the actual vs. expected signal trend (counterfactual explanations), showing related alarms (contextualization), and generating text-based explanations for the numerical data (natural language explanations). In the evaluation with several power plant operators, we are able to demonstrate the explanatory value of the presented solution for this industrial domain.

2 Related Work

The debate in literature about "Explainable AI" (XAI) concerns the need that predictions from AI models appear plausible to the user. Some ML models, such as linear models and decision trees, are inherently interpretable [7]. On the other hand, many ML models, and here most notably deep neural networks, present themselves rather as opaque black boxes to the user [4]. Various XAI techniques have been built to provide insights into the general reasoning of ML models or explaining specific model predictions. The general focus of XAI is to increase trust and to enable better supervision and debugging of ML models. In our case study, we extend this scope of XAI and investigate, whether XAI appplied to ML models using process data as inputs can help process operators (specifically power plant operators) to gain insights into the condition of the industrial processes and support them in the task of analyzing abnormal situations.

An important categorization for XAI techniques is by the type of explanations they generate. Here, feature importance methods [8, 9], prototypes (creation of a prototype, illustrating how a class typically looks [10]), case-based explanations (e.g., finding similar cases in the training dataset [11]), contextualization (explaining by adding, e.g., domain-specific context information) [12], natural language explanations (i.e. rephrasing a model output in natural language) [13], causal mechanisms (e.g., rule generation [11]), and counterfactual explanations [14] can be distinguished. The proposed solution in this paper presents an example implementation for five of the above-named XAI techniques, as mentioned in the introduction.

Many of these XAI techniques are limited to specific data types, especially tabular, image, and textual data [15]. While these data types can be found in power plants, too (e.g., in the form of tabular alarm logs, IR thermal images for equipment monitoring, or textual data in electronic operator shift books), the monitoring of plants is primarily based on *time series data* (i.e. sensor readings from the automation equipment in the plant). After initially receiving little attention in XAI research [16, 17], time series

data has recently gained increasing attention, with more publications appearing on the topic [18]. For example, in 2017, LIME, originally developed by Ribeiro et al. [8] to explain tabular, textual, and image data, was extended by Metzenthin [19] to include time series data, resulting in the creation of the new library LIME-for-Time. Another popular technology that has been applied to time series is SHAP [9], but both LIME-for-Time and SHAP suffer from being limited to univariate time series data [20]. Indeed, a notable part of prior art we found is limited to explain univariate time series, whereas many industrial AI use cases, including the one discussed in this paper, involve *multivariate* time series. Furthermore, we noted a lack of literature providing industrial *examples* [21]. This study aims to contribute towards filling these gaps.

3 Case Study

The LEAG power plant in Boxberg, Germany, commenced operations in the 1970s, and today consists of four plant units with a total capacity of 2575 megawatts (gross). The specific unit considered in this study is called "Unit Q" which has a capacity of 900 megawatts (gross) [22]. Operations began in 2000 and are monitored around the clock from the unit's central control room by a team of 4-5 operators. The operators work in rotating shifts of 8-12 hours. A typical shift is characterized by daily routines with extended periods of inactivity for the operator. Operators would describe a "good shift" as one with little to no activity. However, the complexity and amount of activity sharply increases during abnormal situations, and then, the operator's experience plays a critical role in identifying and resolving these anomalies. Here, an AI-based operator support system could be valuable in helping to detect abnormal situations more easily.

LEAG provided an extensive dataset for this case study, comprising a ten-year export of the control system archive for power plant Unit Q. The dataset is over 3TB in size and contains signal data from individual measurement points (sensor data like flows, levels, temperatures, and pressures) as well as alarm and event data. The chosen technique to detect abnormal plant conditions makes use of a Dense autoencoder neural network architecture. The functionality of this approach has been validated in previous studies for process industries [23, 24, 25, 26]. The focus of this case study was the development of a dashboard solution to visualize and particularly *explain* the predicted anomalies from this anomaly detection model to the operator. To provide these explanations, technical concepts for several visual *explainers* were developed, that are presented in this paper. The explanations should enable the operator to asses whether the presented situation is indeed an anomaly or not (true or false positive). The proposed dashboard solution is presented in the following section.

4 The Anomaly Explanation Dashboard

Fig. 1 depicts the developed dashboard. The underlying anomaly detection model makes predictions for anomalies for the given plant data, and the dashboard visualizes and explains these predictions to the user. The explainers and feedback components are marked in Fig. 1 by the numbers (D1)-(D8). Two additional elements, (N1) and (N2), have been added after our evaluation workshop, based on suggestions received from interviewees ("N" for "new"). It is worth noting that several elements, namely (D1)-(D4), build on the signal-based input and output data from the *same* underlying Autoencoder anomaly detection model. One explainer-area (D5/D6) supplements these signal-based explanations with alarm log data which is textual data. The textual elements (N1/N2) use text generation techniques to transform some of the numerical information on the dashboard into natural language.

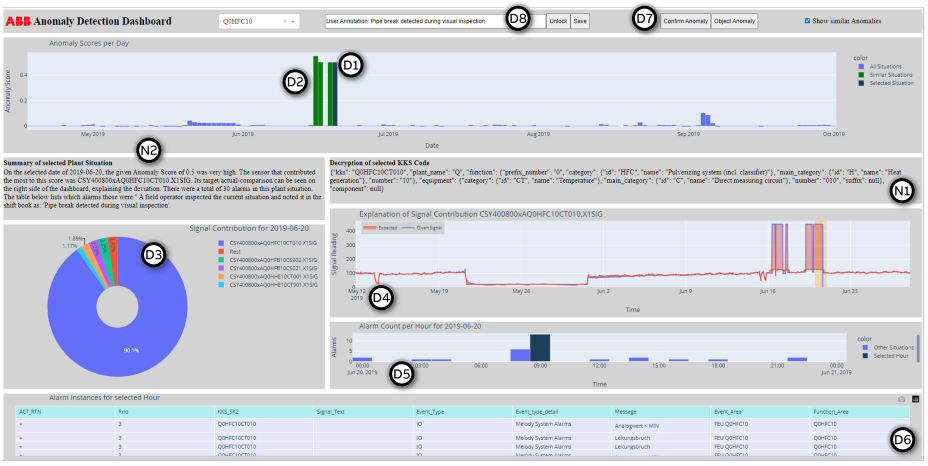


Figure 1: The anomaly detection dashboard, with its explainers and feedback options.

The Explanation Methods

The user starts by selecting a plant area to analyze from a dropdown list on the top left of the dashboard. In the example in Fig. 1, the coal pulverizing system "Q0HFC10" is selected. The histogram below then shows the detected anomalies in this plant area over time. The presence and strength of an anomaly is indicated by an anomaly score. The score is calculated based on the reconstruction error from the underlying Autoencoder model for reconstructing the multivariate time series for the given plant situation, as presented in [24, 25]. To view the explanations for an anomaly, the user can select it from the histogram. The selected situation is then highlighted by dark blue color (D1).

The first explainer is realized in the same histogram: When a plant situation has been selected (D1), the dashboard also searches for any similar plant situations, and highlights them in green (D2). In XAI literature, showing similar situations corresponds to the explanation technique "*case-based explanations*" [11]. Showing similar anomalies can significantly help operators in understanding a given situation, as the root-cause for some of the similar past situations may have already been investigated and thus is known. Say, there was a similar anomaly two weeks ago, and after visual inspection by a field operator it turned out that the problem was due to a clogging issue in a roller mill. Knowing that the present anomaly is again very similar to the one from two weeks ago, provides an indicator that it could be again the same clogging issue.

To find similar anomalies, a separate anomaly clustering model was trained, using the predicted anomalies from the autoencoder as input to the model training. As presented in [24], it is possible to calculate the reconstruction error *per signal* in the multivariate signal data, which provides a kind of "fingerprint" for anomalies. For case-based explanations, these anomaly fingerprints where used as input features to the clustering algorithm, to determine similar anomalies with similar fingerprints. Several clustering algorithms from scikit-learn [27] have been explored for this purpose. First exploratory analysis provided best results for the GaussianMixture clustering. However, more in-depth analysis is needed here which is a further research in this project.

The donut chart in (D3) provides insights into "*feature importance*", another explanation technique [8, 9]. It shows how much different signals contributed to an anomaly. The complexity of equipment in a plant's functional area can provide diverse potential root causes for an anomaly. Being able to pinpoint the anomaly to a specific sensor signal, or a

small number of signals, can significantly reduce the search space for field operators. The calculation of the signal contributions makes use again of the signal-wise reconstruction errors. Aggregating the error values for each sensor during the given time window of the anomaly (as presented in [24]) constitutes the total contribution of that sensor to the anomaly that is then displayed in the donut-chart (D3).

When selecting a signal from the donut, the trend chart in (D4) presents two trend lines: A blue trend representing how the signal *actually* progressed in the given (anomalous) situation, and a red trend how it *should have* progressed in the normal case. Being able to directly compare a trend to the expected normal trend can provide additional explanation for the potential anomaly root-cause. This explanation corresponds best to the "*counterfactual explanation*" technique [14] from the state of the art: It presents the model's concepts of the normal state. The calculation of the normal and actual signal trends is again making use of the autoencoder for predicting the anomaly. This model was trained using normal data when the plant operated within normal bounds [24, 25]. When anomalous data is given to the model, a reconstruction error occurs because the model reconstructs this data more closely to how it had learned it. The explainer in (D4) makes use of this fact by visualizing the input signals to the model as actual trends, and the reconstructed output as approximation of the normal trends. In difference to this reconstruction-based approach, the state-of-art counterfactual explanations usually represent the sample with the smallest changes where the model prediction flips (here: from 'anomaly' to 'normal'). Finding the sample with the smallest change usually involves a search or optimization algorithm, whereas using the reconstructed data has the benefit that it is available without additional computation.

The explainers up to this point build on *time series data* from sensor signals. As mentioned, the research team was provided also with the historic alarm logs from the plant. Alarm data contains a lot of textual information, such as an alarm title, and a more detailed alarm description text. Correlating this textual information with an anomaly can help operators to get more context information about the anomaly. In the XAI literature, this explanatory technique is highlighted as "*contextualization*" [12], i.e. to explain by adding domain-specific context information. In the dashboard it is realized in (D5) by showing the amount of alarms that have occurred during an anomalous situation, and (D6) showing the textual alarm details for these alarms.

The aim of the explanations in the dashboard is to enable operators to form their own judgment as to whether those explanations seem plausible if the presented situation is indeed a true anomaly or not (true or false positive). Beyond this operational use, we also aim to provide operators the possibility to give feedback, to be able to retune the models in case predictions were not satisfactory, which is a further research in this project. For this purpose, there is the option in (D7) that allows the user to confirm or object to anomalies. A further feedback option in (D8) allows operators to textually annotate anomalies, e.g. by entering their conclusion. The feedback provided can help model developers to include or exclude data during model retraining, as proposed in [28]. However, textual annotations could also be presented to operators, as additional information in the dashboard to explain similar anomalies in future.

5 User Evaluation

In September 2023, an evaluation workshop was carried out with operators from the LEAG power plant in Boxberg. The workshop aimed to assess how helpful operators found the developed solution in detecting and explaining abnormal plant situations. For the evaluation, we were able to interview five operators during their regular operator training at the LEAG power plant simulator & training center in Lübbenau, Germany

[29]. This came with the benefit of being in an environment similar to their actual control room in Boxberg. Among the respondents, there was a good mix of different levels of work experience, from trainees with only a few months of experience, to operators having been there since the plant was built 40 years ago and seem to know it from the ground up. The respondents' job roles included *control room operators* and *field operators*. A control room operator's primary workplace is in the control room, where the plant is monitored and operated. Field operators are also located in the control room, but are required to go out into the field when a problem is detected, to visually inspect and correct the problem. Younger operators typically start out as field operators and move up to become control room operators as they gain experience. During their training day, we were given a free time slot to present our dashboard to all the operators in the large group. In a second step, we then conducted individual interviews, always with one operator at a time, while the other operators continued with their training. In the interviews, we asked each operator for their feedback on the dashboard using a semi-structured questionnaire. Each interview lasted approximately 15 minutes.

In the first part of the interview, we wanted to get to know the relevance of detecting anomalies today. For all interviewees, the detection of anomalies is an important part of their work. Today, operators detect anomalies by observing the updates of the alarms and message logs, and by constantly monitoring various signal trends. By interpreting anomalous changes in those trends based on their experience, it is possible to detect anomalies and to narrow down their root-causes. However, this analysis requires a lot of experience. Moreover, not all problems can be detected in the control room. There is always the need for a field operator to go out and visually inspect the issue. For example, when standing in front of an engine, one may *hear* that the engine has strange running noises, which is not possible from the control room, as one field operator pointed out.

The frequency of anomalies can vary significantly. There are days, when nothing happens. Operators, however, are in constant anticipation that something *can* occur. Many anomalies can arise, e.g., due to varying coal quality grades coming in from the open pit mines nearby Boxberg. Also plant startup and shutdown phases, e.g. for maintenance or revisions, are a source of anomalies that need to be handled.

In the interviews, all respondents found the presented solution to be a very helpful tool, because it helps to detect and narrow down plant disturbances, which is a crucial part of their work. However, there was also consensus that AI may only alert the operator about anomalies, but should never autonomously intervene when anomalies are detected. The operator should remain in the position to assess the situation themselves, to determine whether it is indeed an anomaly or not, before any actions are taken. That is precisely why it is important to explain anomalies to operators. With regard to a useful integration of the dashboard into the control system, operators could imagine having it on a separate screen, or bringing it up on an existing screen on demand (e.g. in case of an anomaly-alert) because screens are already quite full today. Field operators did not raise a need for an own mobile app, because they also look at the screens together with the control room operators, to discuss anomalies, before going out to the plant to fix them. Here, the presented dashboard was found helpful also by field operators, to discover where to go.

In the second part of the interviews, we asked operators to rate how helpful they found the different dashboard elements (D1)-(D8), for explaining anomalies, and for giving feedback to them. For each dashboard element, we asked interviewees for their rating from "not helpful at all" (0 points) to "absolutely helpful" (4 points). Interviewees could also provide further comments that we noted down in a comment field at each question. The given operator ratings are presented in Table 1, and their corresponding comments, are discussed in the following.

Table 1: Operator's ratings for the evaluated explainers and feedback options (D1)-(D8).

		Number of responses					average rating
		not helpful at all (0 points)	slightly helpful (1 point)	moderately helpful (2 points)	very helpful (3 points)	absolutely helpful (4 points)	(on the scale 0 - 4 points)
Anomaly detection							
D1	Anomaly scores histogram				4	1	3,2
Anomaly explanation							
D2	Highlight similar situations			1	1	3	3,4
D3	Signal contribution ("donut")					5	4
D4	Actual-vs-normal trend comparison				1	4	3,8
D5, D6	Alarm count & details				3	2	3,4
Operator feedback							
D7	confirm/object anomaly predictions		1	1	2	1	2,6
D8	Annotate anomalies with a text			1	1	3	3,4

Overall, the indication of anomalies with help of the histogram (D1) was rated as very helpful. One operator suggested additionally displaying a threshold line to indicate at which point an anomaly score should be considered very critical, and to alert the operator once this threshold is exceeded, reducing the need for constant monitoring of the dashboard. Furthermore, such a threshold would not have to be fixed, but could be individually adjustable by the user (e.g. through a slider), as some operators may prefer earlier notifications, having a greater need for information than others. Among the various concepts for explaining anomalies, the donut chart in (D3), that analyzes which sensor signal contributed most to an anomaly, was rated to be the most helpful, receiving the highest rating from all respondents. During troubleshooting it would help to determine where to direct the field operator to locate and rectify the issue.

Identifying and showing similar past situations in the dashboard (D2) would help especially when there is already an investigation into why these situations occurred, as field operators do keep a log book about what has been repaired in the past. In those cases the log book entries could suggest a known solution to a similar new problem.

The actual-vs-normal trend comparison in (D4) was mostly rated as "absolutely helpful". Here, interviewees noted that this comparison is not directly displayed to operators in the operator screens today. This means, if there is an anomaly, operators cannot directly see, which upper and lower limits are configured for a sensor. To see how the limits for a signal are set, one would have to check in the so-called control blocks of the process control system, and then it would be possible to compare these normal values with the actual sensor readings in the trend displays, to achieve a kind of actual-vs-normal comparison. However, this process is cumbersome and involves many manual steps.

A similar rationale was provided to us regarding the alarm analysis in (D5/D6). As shown in Table 1, respondents rated this as "very useful" to "extremely useful" for explaining an anomaly. Initially, we were skeptical whether information about alarms would add value to the dashboard, given that all alarms are already displayed in the screens of the control system. However, operators pointed out that there is a benefit to directly see which alarms are related to an anomaly. This speeds up the search process. Otherwise, one would always have to manually search for the relevant alarms in the control system. And speed is crucial when having to make swift decisions.

One suggestion raised in the interviews was to translate the encoded sensor names, shown e.g. in the legend of (D3), into clear text. These codes follows the so-called "KKS" industry standard [30] for identifying different types of equipment in a power plant. A more experienced control room operator is used to read these codes to identify the equipment. However, one cannot expect a young field operator to understand a given KKS name at first glance, when having to go out into the plant. One dashboard extension made after the workshop was therefore the component (N1) to automatically decode the selected sensor names. Due to time constrains, the decoded output is a simple json string, and the visualization of this string can certainly be improved.

Another idea from the discussions was to summarize all the (numerical) explanations for a given anomaly in a short textual paragraph in a natural language format. This is the second addition (N2) made to the dashboard after the workshop interviews, and corresponds to the explanatory technique from XAI literature to "*rephrase model predictions in natural language*" [13]. In the current solution, this is achieved using a template text for the paragraph with space holders, and rules how to textually interpret the analysis from the explainers to fill these place holders. For example, if the predicted anomaly score in (D1) is above a 75 percentile threshold, then this score is translated to be "very high" in the text summary, as shown in the example of (N2) in Fig. 1.

The last part of the interviews concerned the "operator feedback" (D7/D8), i.e. the possibility to provide feedback to an anomaly. This part raised the most concerns from operators. With regard to options (D7) that should allow an operator to confirm anomalies or object to them, interviewees raised the concern about potential "chronic dissenters" who might always click "object". This would lead to many true anomalies being labeled as "false positives", resulting in the unfortunate consequence that those anomaly would not be displayed anymore in future. The option (D8) to annotate anomalies with a short textual description reminded interviewees of their *shift books* they have today, which are also used to log incidents, and looked at again when a similar incident arises. Hence, interviewees rated this feature as helpful, also in our dashboard. However, there might be a challenge when different operators provide varying explanations for an anomaly. Here, the concern was raised that multiple, conflicting annotations could be made for an anomaly, resulting in the need to search for the annotation which are correct, or most useful, whenever a similar anomaly occurs again.

6 Conclusion

In our evaluation interviews with power plant operators, the explainer concepts were predominantly rated as very helpful by the operators to investigate the root causes of the identified anomalies. Most concerns expressed by operators were related to the issue of incorporating user feedback in the tool, e.g., how to deal with contradictory feedback, or how to deal with overly cautious feedback where true anomalies could be labeled as false predictions and therefore not shown anymore in future. It is worthwhile noting that this goes beyond the common objective of XAI, which tries to provide insight into the reasoning of the model. Instead, the combination of model and XAI provides the operator with insight into the process state.

The focus of our research so far has been on developing the explainers to make anomalies understandable, resulting in the dashboard presented in this paper. Our further research will shift more towards the issue of *interactive machine learning (IML)* [28] and also *explanatory interactive machine learning (XIL)* [31], where the given operator feedback in the tool should enable the re-tuning of the anomaly detection models. Here, the expressed concerns raised in our interviews will be valuable input for implementing this feedback component properly. A limitation of the research demonstrator is that it operates *offline* on the exported plant data. To deploy such a solution in a live plant, various technical questions must be addressed, which is not a focus of this research.

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) as per resolution of the German Parliament under the funding code 01IS22030A [6]. We gratefully acknowledge this support. The authors are responsible for the contents of this contribution.

References

[1] Wenjing Lyu and Jin Liu. Artificial intelligence and emerging digital technologies in the energy sector. *Applied energy*, 303:117615, 2021.

[2] Hubert Szczepaniuk and Edyta Karolina Szczepaniuk. Applications of artificial intelligence algorithms in the energy sector. *Energies*, 16(1):347, 2022.

[3] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[4] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[5] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. In *Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8*, pages 563–574. Springer, 2019.

[6] EU-funded research project EXPLAIN - web page. `https://explain-project.eu/`. Accessed: 2023-10-23.

[7] Nicolas Pfeuffer, Lorenz Baum, Wolfgang Stammer, Benjamin M Abdel-Karim, Patrick Schramowski, Andreas M Bucher, Christian Hügel, Gernot Rohde, Kristian Kersting, and Oliver Hinz. Explanatory interactive machine learning. *Business & Information Systems Engineering*, pages 1–25, 2023.

[8] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[9] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[10] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] Rich Caruana, Hooshang Kangarloo, John David Dionisio, Usha Sinha, and David Johnson. Case-based explanation of non-case-based learning methods. In *Proceedings of the AMIA Symposium*, page 212. American Medical Informatics Association, 1999.

[12] Freddy Lecue. On the role of knowledge graphs in explainable ai. *Semantic Web*, 11(1):41–51, 2020.

[13] Erik Cambria, Lorenzo Malandri, Fabio Mercorio, Mario Mezzanzanica, and Navid Nobani. A survey on xai and natural language explanations. *Information Processing & Management*, 60(1):103111, 2023.

[14] Ruth MJ Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI*, pages 6276–6282, 2019.

[15] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel A Keim. Towards a rigorous evaluation of xai methods on time series. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4197–4201. IEEE, 2019.

[16] Ioana Giurgiu and Anika Schumann. Explainable failure predictions with rnn classifiers based on time series data. *arXiv preprint arXiv:1901.08554*, 2019.

[17] Sarthak Manas Tripathy, Ashish Chouhan, Marcel Dix, Arzam Kotriwala, Benjamin Klöpper, and Ajinkya Prabhune. Explaining anomalies in industrial multivariate time-series data with the help of explainable ai. In *2022 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 226–233. IEEE, 2022.

[18] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Díaz-Rodríguez. Explainable artificial intelligence (xai) on timeseries data: A survey. *arXiv preprint arXiv:2104.00950*, 2021.

[19] Emanuel Metzenthin. Lime-for-time. `https://github.com/emanuel-metzenthin/Lime-For-Time`, 2017.

[20] Felix Mujkanovic, Vanja Doskoč, Martin Schirneck, Patrick Schäfer, and Tobias Friedrich. timexplain–a framework for explaining the predictions of time series classifiers. *arXiv preprint arXiv:2007.07606*, 2020.

[21] Arzam Kotriwala, Benjamin Klöpper, Marcel Dix, Gayathri Gopalakrishnan, Dawid Ziobro, and Andreas Potschka. Xai for operations in the process industry-applications, theses, and research directions. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, pages 1–12, 2021.

[22] Official website of the LEAG Power Plant in Boxberg, Germany. `https://www.leag.de/de/geschaeftsfelder/kraftwerke/kraftwerk-boxberg/`. Accessed: 2023-10-23.

[23] Marcel Dix, Benjamin Kloepper, Jean-Christophe Blanchon, and Elise Thorud. A formula for accelerating autonomous anomaly detection. *ABB Review*, 2:2021, 2021.

[24] Marcel Dix, Ashish Chouhan, Srishti Ganguly, Sripada Pradhan, Devika Saraswat, Surbhi Agrawal, and Ajinkya Prabhune. Anomaly detection in the time-series data of industrial plants using neural network architectures. In *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 222–228. IEEE, 2021.

[25] Marcel Dix and Srishti Ganguly. Ai-based anomaly detection of asset failures in industrial process plants. *atp magazin*, 64(5):60–67, 2022.

[26] Arzam Kotriwala, Ruomu Tan, Pablo Rodriguez, Marcel Dix, Benedikt Schmidt, and Anne Lene Rømuld. Plant operator support using industrial artificial intelligence. *atp magazin*, 65(10):70–76, 2023.

[27] Scikit-Learn - Overview of clustering methods. `https://scikit-learn.org/stable/modules/clustering.html`. Accessed: 2023-10-23.

[28] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

[29] LEAG power plant simulator in Luebbenau, Germany. `https://www.konferenzcenter-spreewald.de/de/kraftwerkssimulator`. Accessed: 2023-10-23.

[30] VGBE-Standard: Kraftwerk-Kennzeichensystem (KKS). `https://www.vgb.org/shop/kks.html/`. Accessed: 2023-10-23.

[31] Stefano Teso and Kristian Kersting. Explanatory interactive machine learning. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 239–245, 2019.

ROOT CAUSE ANALYSIS USING ANOMALY DETECTION AND TEMPORAL INFORMED CAUSAL GRAPHS

Josephine Rehak[1], **Shahenda Youssef**[1], **and Jürgen Beyerer**[1,2]
[1]Karlsruhe Institute of Technology (KIT), Kaiserstraße 12, Karlsruhe
[2]Fraunhofer IOSB, Fraunhoferstraße 1, Karlsruhe

ABSTRACT

In industrial processes, anomalies in the production equipment may lead to expensive failures. To avoid and avert such failures, the identification of the right root cause is crucial. Ideally, the search for a root cause is backed by causal information such as causal graphs. We have extended a framework that fuses causal graphs with anomaly detection to infer likely root causes. In this work, we add the use of temporal information to draw temporal valid conclusions about the potential propagation of anomalous information in causal graphs. The use of the framework is demonstrated on a robotic gripping process.

Keywords causal graphs · anomaly detection · multivariate timeseries · root cause analysis

1 Introduction

The investigation of root causes has bugged humans since the beginning of engineering [9, 12]. Since then, a variety of approaches have been published to infer root causes [7]. Only a minority of works are based on causal research. Consider that the spread of anomalous information throughout any system or process has to adhere to the laws of causality. Causal research inspects in-depth the interplay between the variables in given data and allows the identification of causes and effects [13]. Thus, it may support the identification of a fault's root cause and the factors contributing to it [8, 4].

We show in this publication how to draw conclusions about the root cause of anomalies and failures by considering their fundamental temporal laws. Thereby, we add to the causal reasoning framework introduced by [14]. The fundamental setup of the framework is to first detect the anomalies, to determine the features that made a data entry anomalous, to retrace their potential causes in the corresponding provided causal graph, and finally to evaluate the match between the potential anomalies of a root cause according to the graph and the actual detected anomalies. In this publication, we make use of coarse temporal information in the form of process steps to allow temporal sound deductions about the potential root causes in the graph. It is important to note that the anomaly

detection step is not part of this contribution. In this matter, we refer to existing work as summarized by [6].

We have structured our work in the following manner. First in Section 2, we briefly introduce the required fundamentals to then show in Section 3 the related work in the causal research domain. Section 4 is dedicated to the adaptation of the root cause algorithm to integrate coarse temporal information in its deductions. Section 5 shows in an example robotic gripping scenario how the deductions of the algorithms improve. Finally, Section 6 concludes the work.

2 Fundamentals

Causal Graphs
A common way to represent causal relations is in the form of graphs $G = (\mathcal{V}, \mathcal{E})$. These graphs commonly consist of a set of nodes representing variables $\mathcal{V}$ and a set of edges $\mathcal{E}$. The latter represents direct causal relations between variables [13]. For timeseries data, the use of summary graphs is common [3]. In them, each variable is associated with a timeseries. An edge $a \to b$ exists between the variables $a \in \mathcal{V}$ and $b \in \mathcal{V}$ if there exists any sample $a_i \in a$ causing an effect in a sample of $b_j \in b$ if $a \neq b$, such that a_i precedes or is equitemporal $i \leq j$ to b_j in time or if $a = b$ that a_i precedes b_j in time $i < j$. We call it a causal path $a \to^* c$ from a to c if a sequence of directed edges from a_i to c_j is present [2].

Temporal Law of Causation
It is known that the effect of an event may never precede its cause [5]. Under the assumption that no other influence is present and if it is known that two variables are causal related $a \to b$, then b can never occur before a. Likewise, if $a \to b \to c$ is given, c may only occur after a and b.

Jaccard Similarity
The Jaccard similarity [10] is a similarity measure often used for recommender systems. It divides the intersection of two sets $\mathcal{A} \cap \mathcal{B}$ by their union $\mathcal{A} \cup \mathcal{B}$.

$$\text{Jaccard} = \frac{\mathcal{A} \cap \mathcal{B}}{\mathcal{A} \cup \mathcal{B}} \tag{1}$$

A result of one indicates a perfect match, while a result of zero indicates two different sets.

3 Related Work

3.1 Causal Research for Root Cause Analysis

Causal Methods and Anomaly detection methods have been combined before in the literature for various application areas.

Yang et al. [15] published a general framework to detect anomalies and to identify root causes in multivariate time series. They learned causal structure from data, identified modules in the investigated system, and performed anomaly detection on each separate module. The location of the anomalies is supposed to represent the root causes.

Koutroulis et al. [11] combined causal graphs and anomaly detection methods to robustly detect anomalous cyber security attacks in the control system of a water treatment testbed. They trained structural causal models (SCMs) offline with attack free data. In online scenarios, the SCMs allowed the computing of prediction errors to identify critical

dependencies in the data. Each error sequence was assessed with the average treatment effect and hypothesis tests. Root cause candidates are indicated by providing the set of sensors associated with the attacks.

Agrawal et al. [1] used one Bayesian Network (BN) for each type of root cause to diagnose root causes in a coal-fired power plant. Each was provided by experts in advance. Based on fixed thresholds, the variables were detected to be either normal or anomalous. The anomalies were propagated in the BN and the magnitude and the type of root cause were indicated.

The framework RootCLAM by Han et al. [8] learns a Variational Causal Graph Autoencoder from normal data to then identify for anomalous data the root cause features where in comparison to normal data, the presence of a foreign influence is evident.

None of the shown approaches provided the possibility to argue over potential root causes. Also, none of them assessed the relation between the root cause and the sensor measurements themselves.

3.2 Temporal Information for Root Cause Analysis

The procedure EasyRCA [4] uses a given acyclic summary causal graph (ASCL) to identify root causes in multivariate timeseries data. According to their work, root causes may be variables with anomalies, whose first appearing anomaly might not have been propagated by any other causing variable under the assumption the anomaly may not precede its cause in time in the ASCL. Additionally, root cause variables are identified by comparing the adapted total effect estimations on direct relations of a normal and an anomalous regime to detect variables affected by foreign influences.

Other than in our work, the summary graph is required to be acyclic and root causes are not known beforehand.

3.3 Preceding Work

In [14], we published the first algorithm to infer the probability of a root cause to be the cause for observed anomalies. It makes use of the Jaccard Similarity Root Cause Score (JRCS) to evaluate the fit between the set of potentially affected measured variables $\mathcal{V}_{G,r}$ for a root cause r and the set of discovered anomalous variables $\mathcal{V}_A$.

$$\text{JRCS}(r) = \frac{\mathcal{V}_A \cap \mathcal{V}_{G,r}}{\mathcal{V}_A \cup \mathcal{V}_{G,r}} \tag{2}$$

This version of the algorithm only uses variables for inference on the root cause without considering temporal aspects.

4 Temporal Informed Algorithm for Root Cause Analysis

4.1 Prerequisites

To perform the updated RCA procedure, we split the process into process steps $p \in \mathcal{P}$, given $\mathcal{P} = \{1, 2, ..., |\mathcal{P}|\}$. Also, we require the order of assignment of process steps p to correspond to the temporal and causal order.

We will work with an adaptation of the summary graphs $G = (\mathcal{V}, \mathcal{E})$ introduced before. In them, we may not access the timeseries of a subset of variables $\mathcal{V}_L \subset \mathcal{V}$, as they are latent or unmeasured. Integrating these latent variables is optional. They support human

understanding, but may be omitted by introducing direct causal relations between its causing and caused non-latent variables.

The adapted summary graphs need to contain a set of unary variables $\mathcal{V}_R \subset \mathcal{V}$ as root cause variables annotated by an expert. Also, we require a set of measured variables $\mathcal{V}_M = \mathcal{V}/(\mathcal{V}_R \cup \mathcal{V}_L)$.

Additionally, it needs annotations of process steps $\mathcal{P}_{a\to b} = \{p \in \mathbb{N}\}$ to be provided for all $a, b \in \mathcal{V}$ if $a \neq b$ and $a \to b$ holds. To create such annotations as $\mathcal{P}_{a\to b}$ manually, one should list all process steps in which an anomaly in a would show effects in b.

As the anomaly detection should be performed in advance and is not part of this contribution, we require it to provide a tuple set $\mathcal{T}_A = \{(m, p), ...\}$ given $m \in \mathcal{V}_M$, $p \in \mathbb{N}$, which contains pairs of the discovered anomalies and the process steps they occurred in.

4.2 Graph Preprocessing

In the preprocessing step, we can calculate $\mathcal{P}_{r\to^* m}$ for each path $r \to^* m$. It represents the maximum possible set of process steps in which $r \in \mathcal{V}_R$ could cause anomalies in $m \in \mathcal{V}_M$ for the specific path. The algorithm to calculate a unique set of $\mathcal{P}_{\{r\to^* m\}}$ is shown in Algorithm 1. Since multiple paths $r \to^* m$ may be possible, we express with $\mathcal{P}_{\{r\to^* m\}}$ the maximum possible set of process steps for a set of paths. To exclude double entries, we use in the following the $unique$ function.

Algorithm 1: Compute valid process steps for path sets

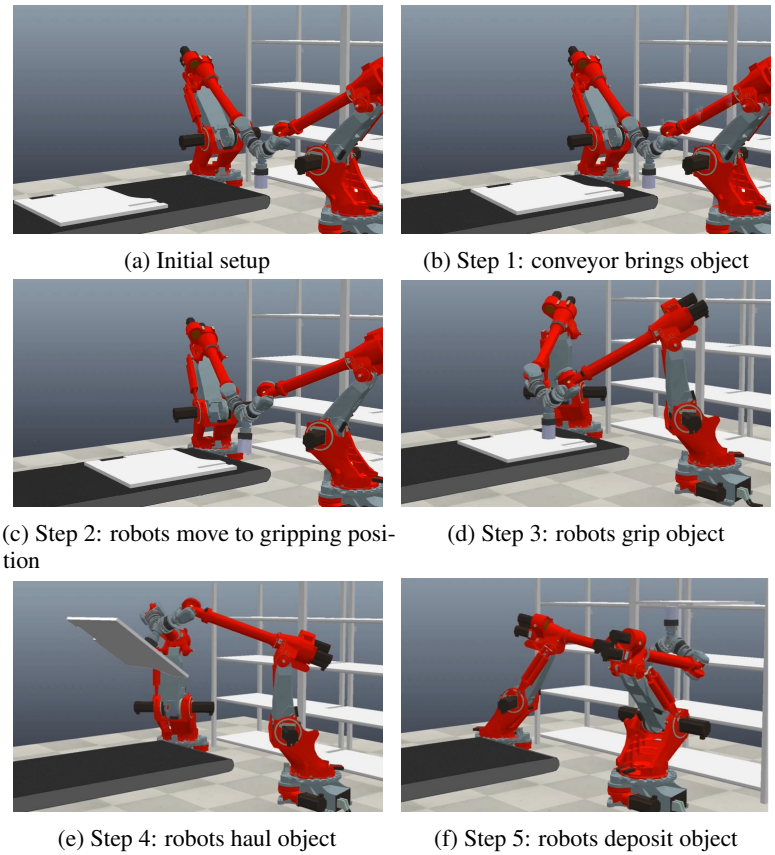
Input : Set of edges $\{r \to^* m\}_G$ from r to m in graph G,
set of assigned process steps $\mathcal{P}_{a\to b}$ for any edge $a \to b$
Output : Set of process steps $\mathcal{P}_{\{r\to^* m\}}$ for each path $r \to^* m$

$\mathcal{P}_{\{r\to^* m\}} \leftarrow \emptyset$
for $r \to^* m \in \{r \to^* m\}_G$ **do**
  $i \leftarrow 0$
  $\mathcal{P}_{\text{temp}} \leftarrow \emptyset$
  for $a \to b \in r \to^* m$ **do**
    $\mathcal{P}_{\text{temp}} \leftarrow \{\mathcal{P}_{a\to b} \geq i\}$
    $i \leftarrow \min(\mathcal{P}_{\text{temp}})$
  $\mathcal{P}_{\{r\to^* m\}} \leftarrow \mathcal{P}_{\text{temp}} \cap \mathcal{P}_{\{r\to^* m\}}$
return unique$(\mathcal{P}_{\{r\to^* m\}})$

As each edge of a path can occur in each process step of its corresponding set and thus allows a variety of process step sequences, we need to ensure the valid temporal sequence of edges for each $r \to^* m$ as is implied by the temporal law. We consider only possible process step sequences which allow the sequential and equitemporal spread of anomalous causal information. The latter is due to the coarse-grained temporal categorization of edges into process steps, which does not allow detailed deductions about the temporal sequence.

Given $\mathcal{P}_{r\to^* m}$, the tuple set $\mathcal{T}_G(r)$ is generated for each r by calculating tuples of measurement variables and their potentially affected process steps. For that, we make use of $\{r \to^* m\}_G$ denoting the set of all possible paths from r to m in G.

$$\mathcal{T}_G(r) = \{(m, p) | \forall\, r \to^* m \in \{r \to^* m\}_G, p \in \mathcal{P}_{\{r\to^* m\}}\} \tag{3}$$



(a) Initial setup

(b) Step 1: conveyor brings object

(c) Step 2: robots move to gripping position

(d) Step 3: robots grip object

(e) Step 4: robots haul object

(f) Step 5: robots deposit object

Figure 1: Depicted is each process step of the robotic gripping process

4.3 Scoring of Potential Root Causes

In this step, the JRCS scores are calculated for each root cause by the use of an updated formula. This time, its resulting score rates the fit between two tuple sets on a scale from zero to one. We assume this similarity rating to reflect the likelihood of a root cause to have caused the anomalies.

$$\text{JRCS}(r) = \frac{|\mathcal{T}_A \cap \mathcal{T}_G(r)|}{|\mathcal{T}_A \cup \mathcal{T}_G(r)|} \tag{4}$$

The higher the JRCS result, the better the inspected root cause $r \in \mathcal{V}_R$ fits the anomalous observation $\mathcal{T}_A$.

5 Application

5.1 Application Environment

We apply the extended algorithm to a simple robotic gripping process. In this application, we have identified five process steps. Each step is depicted in Figure 1 and is described in the following.

In the first step, a conveyor moves the prepared object into the gripping position. A proximity sensor is located at the gripping position and measures the distance from the sensor to the gripping position using a laser. The distance decreases significantly if the object is present. In the second step, the coupled robots will move from their default position to the gripping position at the conveyor in unison. Then in the third process step, the magnetic gripper attached to the coupling piece of the robots is powered to attract and grip the workpiece. In step four, the robots move the gripped object from the gripping position to the deposit position. Due to the weight of the gripped object, the torque measured by the robots significantly increases during movement. The sensor back at the gripping position does not indicate the presence of the object anymore. Instead, a similar proximity sensor at the deposit position starts indicating its presence upon arrival. In the fifth and final process step, the robots disengage the magnetic gripper to place the object in the final deposit position.

We simulated a fault-free scenario and also two additional faulty scenarios. One root cause is a shifted object on the conveyor belt. The other root cause is an object in the gripping position, where the magnetic force of the gripper is not sufficient to grab it. For each, we generated sensor measurements corresponding to $\mathcal{V}_M$.

5.2 Adapted Summary Graph

We manually constructed an adapted summary graph for the described scenario and depicted it in Figure 2. The corresponding variable sets $\mathcal{V}_M$ and $\mathcal{V}_R$ are indicated by the variables' horizontal placement. The required set of process steps of a causal relation is annotated close to its edge. If edges are bidirected, the annotation for edges is the closest one along the edge. We simplified the causal graph by summarizing the robots' axis angle and angle speed measurements as one variable each. Commonly, each of the seven robot axes has individual measurements. To allow the easy depiction of the graph, some variables with identical names occur several times.

5.3 Application of the RCA Algorithm

We performed the algorithm described in Section 4 on the two anomalous datasets described in Section 5.1. The graph in Figure 2 was preprocessed as described, and the results are depicted in Table 1.

For the anomaly detection, we used a supervised version of the Kth Nearest Neighbor algorithm. Within each process step, it calculated the Euclidean distance for each inspected time step to the features of its closest normal time step. A sum was formed over all such distances of a process step and divided by the number of the process step's entries. A fixed threshold decided if the inspected process step was detected as normal or anomalous. The set $\mathcal{T}_A$ was formed by retrieving from each process step the measurements, which contributed to the Euclidean distance. Table 2 shows the anomalous process steps and the corresponding measurement variables that we discovered as anomalous.

Given the detected anomalies ($\mathcal{T}_A$) shown in Table 2 and the potential anomalous measurements per root cause ($\mathcal{T}_G(r)$) in Table 1, we calculated the JRCS for each potential root cause for both inspected faulty simulation scenarios. In Table 3, we show the JRCS result for our new temporal extension and the original algorithm. It becomes visible that without considering the temporal information, the actual root causes cannot be distinguished. This is the case as in both simulated scenarios, the same variables are affected.

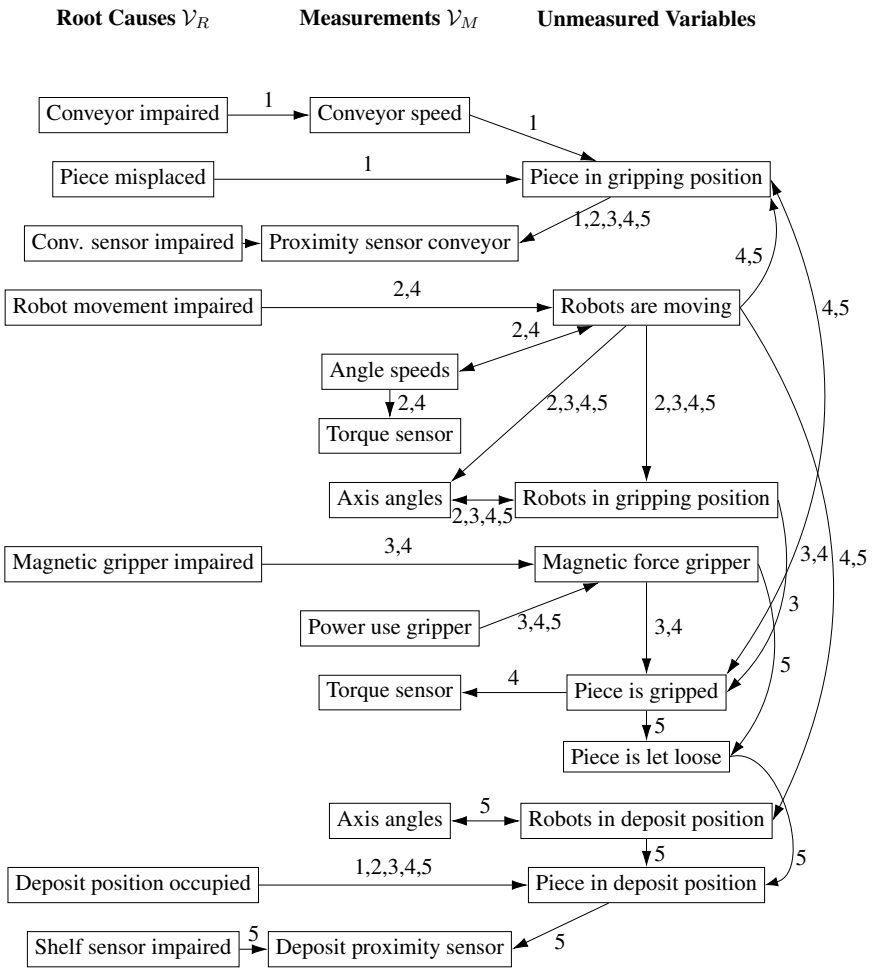


Figure 2: Shown is the adapted summary graph of the robotic setup. The horizontal orientation of the variables indicates their affiliation with the variable sets shown above. The numbers along the edges indicate in which process step an anomaly in the causing variable would be observed in the caused variable. Variables appearing several times are identical.

Table 1: Overview of the potential anomalous measurements affected by the root causes in the shown process steps. Each row corresponds to $\mathcal{T}_G(r)$ for its respective root cause.

		Potential Anomalous Measurements $\mathcal{V}_M$						
		Conveyor Speed	Prox. Sensor Conveyor	Angle Speeds	Torque Sensor	Axis Angles	Power Use gripper	Deposit Prox. Sensor
Pot. root causes $\mathcal{V}_R$	Conveyor impaired	1	1,2,3,4,5	-	4	-	-	5
	Piece misplaced	-	1,2,3,4,5	-	4	-	-	5
	Conv. Prox. Sensor impaired	-	1,2,3,4,5	-	-	-	-	-
	Robot mov. impaired	-	4,5	2,4	2,4	2,3,4,5	-	5
	Magnetic gripper impaired	-	4,5	-	4	-	-	5
	Shelf Prox. Sensor impaired	-	-	-	-	-	-	1,2,3,4,5
	Deposit position occupied	-	-	-	-	-	-	1,2,3,4,5

Table 2: Overview of the discovered anomalous measurements and their corresponding process steps. Each row corresponds to $\mathcal{T}_A$ for its respective scenario.

	Detected Anomalous Measurements $\mathcal{V}_M$						
	Conveyor Speed	Prox. Sensor Conveyor	Angle Speeds	Torque Sensor	Axis Angles	Power Use gripper	Deposit Prox. Sensor
Faulty Scenario I	-	1,2,3,4,5	-	4	-	-	5
Faulty Scenario II	-	4,5	-	4	-	-	5

Table 3: The JRCS for both simulated anomalous scenarios and each root cause if the temporal law is considered or not.

Potential Root Cause	**With Temporal Law**		**Without Temporal Law**	
	Scenario I	Scenario II	Scenario I	Scenario II
Conveyor impaired	0.875	0.500	0.750	0.750
Piece misplaced	**1**	0.571	**1**	**1**
Conv. Prox. Sensor impaired	0.625	0.286	0.333	0.333
Robot mov. impaired	0.286	0.364	0.600	0.600
Magnetic gripper impaired	0.400	**1**	**1**	**1**
Shelf Prox. Sensor impaired	0.111	0.125	0.333	0.333
Deposit position occupied	0.111	0.125	0.333	0.333

We can observe that the original algorithm, which considers only the variables to compute the JRCS, cannot distinguish between two root causes because they affect the same set of variables. In comparison, the extended algorithm rates the actual root causes as the likeliest. The temporal algorithm scores other root causes, which affect similar measurement variables and process steps as depicted in Table 1, as likelier. Root causes with identical $\mathcal{T}_G(r)$, like the last two root cause entries in the table, are rated as equally likely. This shows that the algorithm does not guarantee a clear distinction between root causes. Instead, it must result from the placement of sensors in the process.

6 Summary

In this work, we adapted a causal root cause analysis algorithm to consider temporal process step information. This allows us to draw temporal valid deductions about potential root causes for prior discovered anomalies.

For this purpose, the algorithm uses an adapted summary graph provided by an expert which includes annotations for root cause variables, measurable variables and latent variables. Additionally, the edge annotations contain knowledge about the temporal spread of anomalous information. This causes the algorithm to rely heavily on provided information by an expert but may contribute to preserving expert knowledge.

In our application on a robotic gripper setup, we demonstrate how the discovery of the actual root causes improves by considering coarse process step information. Future work may tackle the issue of adding detailed anomaly types and patterns to the matching process.

Acknowledgements

This work greatly benefited from the Dagstuhl seminar 24031 "Fusing Causality, Reasoning and Learning for Fault Management and Diagnosis".

References

[1] V. Agrawal, B. K. Panigrahi, and P. Subbarao. Intelligent decision support system for detection and root cause analysis of faults in coal mills. *IEEE Transactions on Fuzzy Systems*, 25(4):934–944, 2016.

[2] C. K. Assaad, E. Devijver, and E. Gaussier. Entropy-based discovery of summary causal graphs in time series. *Entropy*, 24(8):1156, 2022.

[3] C. K. Assaad, E. Devijver, and E. Gaussier. Survey and evaluation of causal discovery methods for time series. *Journal of Artificial Intelligence Research*, 73:767–819, 2022.

[4] C. K. Assaad, I. Ez-Zejjari, and L. Zan. Root cause identification for collective anomalies in time series given an acyclic summary causal graph with loops. In *International Conference on Artificial Intelligence and Statistics*, pages 8395–8404. PMLR, 2023.

[5] H. Beebee. *Hume and the Problem of Causation*. Oxford Handbooks, 1981.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[7] Z. Gao, C. Cecati, and S. X. Ding. A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767, 2015.

[8] X. Han, L. Zhang, Y. Wu, and S. Yuan. On root cause localization and anomaly mitigation through causal inference. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 699–708, 2023.

[9] K. Ishikawa and K. Ishikawa. *Guide to quality control*, volume 2. Asian Productivity Organization Tokyo, 1982.

[10] P. Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912.

[11] G. Koutroulis, B. Mutlu, and R. Kern. A causality-inspired approach for anomaly detection in a water treatment testbed. *Sensors*, 23(1):257, 2022.

[12] T. Ohno. *Toyota production system: beyond large-scale production*. Productivity press, 2019.

[13] J. Pearl. Causal inference. In I. Guyon, D. Janzing, and B. Schölkopf, editors, *Proceedings of Workshop on Causality: Objectives and Assessment at NIPS 2008*, volume 6 of *Proceedings of Machine Learning Research*, pages 39–58, Whistler, Canada, 12 Dec 2010. PMLR.

[14] J. Rehak, A. Sommer, M. Becker, J. Pfrommer, and J. Beyerer. Counterfactual root cause analysis via anomaly detection and causal graphs. In *2023 IEEE 21st International Conference on Industrial Informatics (INDIN)*, pages 1–7. IEEE, 2023.

[15] W. Yang, K. Zhang, and S. C. Hoi. Causality-based multivariate time series anomaly detection. *arXiv preprint arXiv:2206.15033*, 2022.

A Hybrid Model for Hot Rolling Pass Schedule Design Using Reinforcement Learning

Marco Kemmerling[1], **Nils Frahm**[1], **Christian Idzik**[2], **Anas Abdelrazeq**[1], **David Bailly**[2], **and Robert H. Schmitt**[1]
[1]Chair of Intelligence in Quality Sensing, RWTH Aachen University, Aachen, Germany
[2]Institute of Metal Forming, RWTH Aachen University, Aachen, Germany

Abstract

Hot rolling of metals requires the design of pass schedules to accommodate different products and arrive at the desired target properties. Pass schedule design is often still performed manually, although automated approaches are emerging in the literature. One such approach is the use of reinforcement learning, which requires extensive training on simulations. To allow for efficient development of suitable reinforcement learning agents, it is important to reduce the computational cost of each simulation step as much as possible. While some simulation models for hot rolling exist, they have not been developed explicitly for reinforcement learning purposes. Today, there is a lack of fast approximate models to allow for efficient (pre-)training of reinforcement learning agents. To address this gap, we propose a hybrid model consisting of a data-driven model augmented by physical approximations. Our hybrid model produces better predictions than either of its components and is efficient enough to allow for quick training of reinforcement learning agents. We demonstrate that reinforcement learning agents pre-trained on our hybrid model can be transferred to a more accurate simulation and achieve good results after a short additional training phase. The two-stage training approach significantly reduces the overall training while achieving similar agent performances. The reduction in training time enables faster iteration cycles in agent and environment design, thereby creating a foundation for the development of reinforcement learning approaches for a wider range of hot rolling scenarios in the future.

Keywords Hot Rolling · Reinforcement Learning · Hybrid Model · Pass Schedule

1 Introduction

In 2022, more than 1,855 million metric tons of crude steel were produced worldwide [1] and most of the produced steel is rolled at least once. Rolling is a well-established

and highly automated metal-forming process that produces finished or semi-finished products such as plates, sheets, and coils for various industries. In rolling, the metal workpiece is passed through a gap between two or more rotating cylindrical rolls to reduce its thickness. In most applications, the metal is reduced to the desired thickness in several steps, called passes. Each pass is defined by among others, the applied thickness reduction, rolling velocity, and the start time of each pass. The process parameters of all passes in a process are defined in a so-called pass schedule. These pass schedules significantly influence the final material properties and they are often designed using very specific heuristics and process knowledge [22].

In recent times, research has focused on automatic pass schedule design using reinforcement learning [18, 7, 8], where a reinforcement learning agent is coupled with a fast rolling model (FRM) to create and evaluate pass schedules. While the computational cost of such FRMs is appropriate for applications of trained agents in practice, the training of reinforcement learning agents requires significant numbers of interactions with such FRMs, which leads to lengthy training times and slow iteration cycles in the design of rolling agents.

To foster more rapid experimentation in the development of rolling agents, we propose the use of a faster approximate model to simulate hot rolling processes. This approximate model can be used to pre-train reinforcement learning agents, such that the majority of training time is spent on the much faster approximate model with a short subsequent training phase of the more realistic FRM. The model we propose is a hybrid model, consisting of a data-driven core trained by supervised learning which is further augmented by a physical approximation describing selected parts of the hot rolling process.

We demonstrate the accuracy of our model with respect to the FRM and further show that it can be used for the pre-training of agents which can then be successfully transferred to the FRM, arriving at a much reduced total training time.

In the remainder of this work, we describe our approach and result in detail, starting with a brief background on hot rolling in Section 2 and a discussion of related work in Section 3. In Section 4, we describe our modeling approach and demonstrate the performance of the hybrid model, before describing experiments on training and transferring reinforcement learning agents in Section 5.

2 Hot Rolling

Among the many subcategories of rolling, this paper focuses on the hot rolling of long slabs or sheets with rectangular cross-sections. In hot rolling, the temperature of the rolled metal is above its recrystallization temperature. For steel, this is usually between 720 and 1260°C [15]. On the one hand, the high temperature allows to deform the metal more easily due to its lower strength and higher ductility. On the other hand, it allows to influence the microstructure and thus the final mechanical properties which are relevant for the final usage. However, the relationships between the selected process parameters, the material behaviour and the final microstructure as well as the mechanical properties are highly complex.

To design hot rolling schedules so that the final material properties are achieved, a model of the process is needed. Initial efforts were made early in the last century to describe the rolling process and the material behaviour in simplified analytical and semi-empirical equations. These equations were integrated into fast-rolling models (FRM) which are often used in the industry as well as in academia. In the hot rolling process, numerous FRMs have been developed to simulate and predict material properties during and after

rolling based on the pass schedule and material parameters. More concretely, they predict the resulting forces, torques, temperature, and microstructure evolution, within seconds. One well-known rolling model called Slimmer (Sheffield Leicester Integrated Model) was developed by Caglayan and Seynon [3] to predict the microstructural evolution in rolling. Another example is the FRM presented by Schey in 2000 [19] where the goal of the FRM is to predict the rolling forces and torques for low carbon steels. Similar models from Bland and Ford in 1948 [2] and Roychoudhury and Lenard in 1984 [16] can be found in the literature.

In this paper, the classical analytical FRM called RoCaT (Rolling Calculation Tool) developed by Lohmar et al. [11] is used. It calculates the complete rolling process within a few seconds and allows the prediction of force, torque, crystallised fraction, and grain size, among others, including height resolution and the influence of shear.

However, when training reinforcement learning agents, a few seconds of pure calculation time for each interaction between agent and the FRM can lead to undesirably long training times, especially if typical deviations in material properties and process disturbances such as incorrect start temperature, longer pause times, etc. are also to be taken into account in the optimisation process.

3 Related Work

Currently, FRMs as described in the previous section are a typical choice to train reinforcement learning agents for hot rolling pass schedule design. While training of such agents using FRMs is possible, the incurred computational costs are significant and can lead to slow development cycles. For instance, Idzik et al. [7] report a training duration of roughly 24 hours when having direct access to a FRM and performing 20,000 training steps. In many cases, direct access to a FRM is not feasible due to intellectual property issues. In the absence of such direct access, training can be accomplished through a Software-as-a-Service (SaaS) architecture as proposed by Scheiderer et al. [18]. This allows for the use of the FRM while preserving the intellectual property of the domain experts, but incurs additional communication overhead. In our own experiments, the training duration increases to roughly one week when accessing a FRM model through a SaaS architecture. As multiple trainings are typically necessary to settle on a suitable reinforcement learning agent design, this impacts the development of reinforcement learning solutions significantly.

Instead of direct training on a FRM, a potential approach to speed up development time is to introduce a pre-training stage, in which a reinforcement learning agent is first trained on a faster, but less accurate data-driven model before being transferred onto the FRM for final adjustments. In recent years, data-driven models have increasingly replaced semi-empirical FRMs, as exemplified by Saravanakumar et al. [17] who use five inputs like coiling temperature and carbon equivalent for the final mechanical properties prediction. In a more advanced model developed by Xie et al. [21], a deep neural network is trained using 27 inputs to predict mechanical properties.

Data-driven models can additionally be augmented by mathematical or physical models to improve their accuracy and training speed. Such hybrid models can take a variety of different shapes, such as embedding physics into the input of data-driven models, using physics to inform the model selection and architecture, or embedding physics into loss functions [20].

In the context of hot rolling, hybrid models have previously been created to predict the roll force [13, 14, 6] by using a mixture of conventional features and features computed

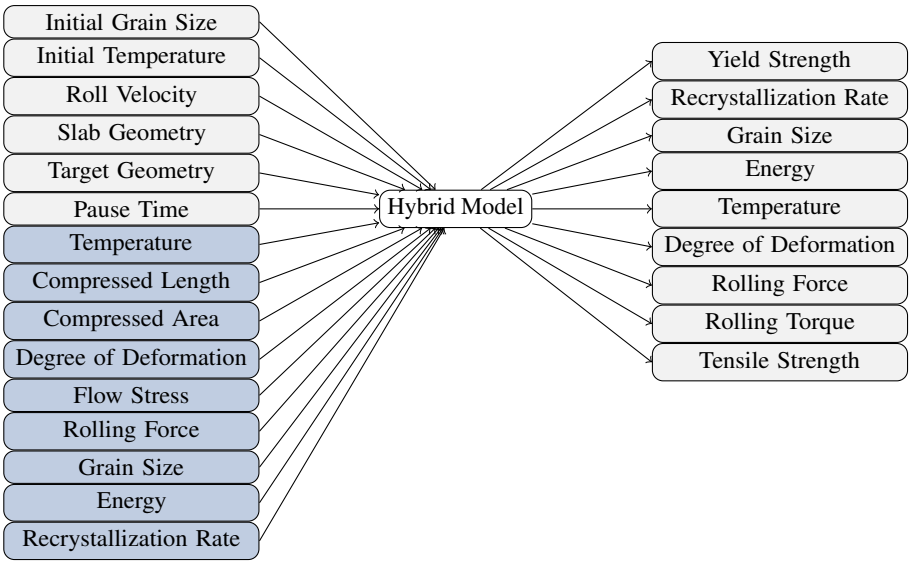


Figure 1: Overview of the proposed hybrid model. Inputs given in grey are regular features, while inputs given in blue are features that have been calculated by a physical approximation. For a more compact representation, several of the visualized individual inputs are actually groups of features.

by a mathematical model as input to a neural network or gradient boosting method. The resulting predictions are generally more accurate than either a purely data-driven or mathematical model. To the best of our knowledge, more comprehensive hybrid models that feature a set of target variables suitable for hot rolling pass schedule design have not been investigated before.

Although not considered in this work, explicit transfer learning techniques have been investigated before and may be helpful to facilitate a successful transfer of reinforcement learning agents from the pre-training stage to the FRM.

4 A Hybrid Hot Rolling Model

According to the classification by Wang et al. [20], our model is a physics-informed machine learning model, with physics embedded into the data-driven model as input in the form of simulation results. Our proposed model produces a set of outputs given a set of conventional features, as well as a set of features calculated by a physical approximation, as visualized in Figure 1. The overall model is actually a collection of individual models for each target variable, which may each be trained by different algorithms. Additionally, each pass is modelled by an individually trained model. As the pass number increases, the inputs for the corresponding model become more extensive, as all the inputs of previous passes are included as well.

The model is constructed along the procedure visualized in Figure 2, which starts with sampling the necessary training data from the RoCaT FRM through the SaaS architecture. We use a weighted random sampling strategy respecting the boundaries of RoCaT, which focuses on parameter values that are likely to be encountered during the design of pass schedules, while sampling more extreme values with reduced probability. An example

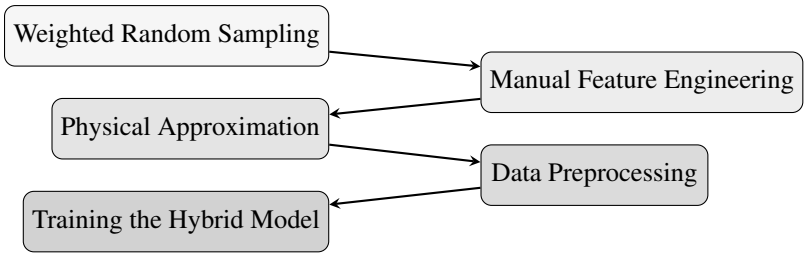


Figure 2: Overview of the construction procedure of the proposed hybrid model.

of the latter is a strong height reduction in a single pass, which leads to undesirable grain sizes. This effect only needs to be captured to the degree that a reinforcement learning agent trained on the model learns to avoid such scenarios and, consequently, very precise predictions of the effect are not required.

Different variables in the sampled data serve as features for the model (see Figure 1), with some being augmented by additional feature engineering to arrive at more suitable data-driven model inputs. Examples of this are (relative) height differences instead of absolute before and after height values, or the time between two passes instead of two separate before and after timestamps. Next to the conventional features, a separate set of features is then computed by a simple and efficient physical approximation of the process. This physical approximation part of the model consists of the equations used by Liebenberg [10, Section 4.1.2 & Appendix A], which are omitted here due to space constraints. These equations include procedures to estimate the temperature of the workpiece by considering its heat loss due to dissipation and contact with the roll, as well as formulae to calculate recrystallization rate, grain size, rolling force, and degree of deformation.

To prepare the data for the training, we standardize the features to have zero mean and unit variance. Furthermore, we perform features selection by using a tree-based approach to compute the impurity-based feature importances. The data is then split into training and test set to train a model for each target variable. For each target, three candidate models are trained using XGBoost [4], LightGBM [9], and CatBoost [12]. The final model for a target is then obtained by choosing the candidate model with the lowest average mean absolute error (MAE) on the test set.

The performance gain of the hybrid model in comparison to either of its two constituents is visualized in Figure 3 and displayed in Table 4. On average, the hybrid model yields far better predictions than the physical approximation alone, and achieves modestly better predictions than the data-driven model alone. The improvement over the data-driven model is especially pronounced in the prediction of the grain size, which is an important parameter due to it being one of the optimization targets in the pass schedule design.

While most predictions improve when incorporating the physical approximation into the data-driven model, the quality of some predictions actually decreases. This especially concerns the temperature predictions, which are almost always slightly worse in the hybrid model than in the purely data-driven model. This is not especially problematic, since the data-driven model can simply be used instead of the hybrid model for temperature predictions. Nevertheless, as Figure 5 shows, the quality of the temperature predictions of the hybrid model is not particularly low. Similarly, there is not much room for improvement for many of the prediction targets, with the exception of tensile and yield strength, recrystallization, and grain size.

Target Variable	Mean	MAE data-driven only	MAE hybrid	Performance Gain in %
Yield Strength	281	11.5	10.3	11
Recrystallization Rate	55.5	4.53	3.67	19
Grain Size	107	10.4	8.39	19
Energy	0.00281	0.000403	0.000318	21
Temperature	1061	2.75	3.20	-16
Degree of Deformation	0.107	0.00549	0.00521	5
Rolling Force	0.646	0.0257	0.0234	9
Rolling Torque	15.3	0.737	0.617	16
Tensile Strength	566	6.04	5.64	7

Table 4: All target variables with their mean values, the MAE of the predictions by the data-driven and hybrid machine learning approach, as well as the performance gain of the hybrid approach in %. The data has been averaged over all passes.

5 Pre-Training and Transfer of Reinforcement Learning Agents

The hybrid model described in the previous section is created to serve as a less computationally expensive replacement for the FRM. The intention is to use the hybrid model to quickly train reinforcement learning agents for pass schedule design and perform a subsequent transfer from the hybrid model to the FRM.

We use a similar agent and environment setup as described by Scheiderer et al. [18], where the environment offers an action space comprising the next workpiece height and the pause time after the next pass, as well as an observation space consisting of the current workpiece height, temperature, and grain size. The agent uses the soft actor-critic algorithm [5] and iteratively creates a pass schedule with up to 5 passes to transform a metal workpiece. To learn to design suitable pass schedules, the agent is rewarded after every t-th pass according to the reward function proposed by Scheiderer et al. [18]:

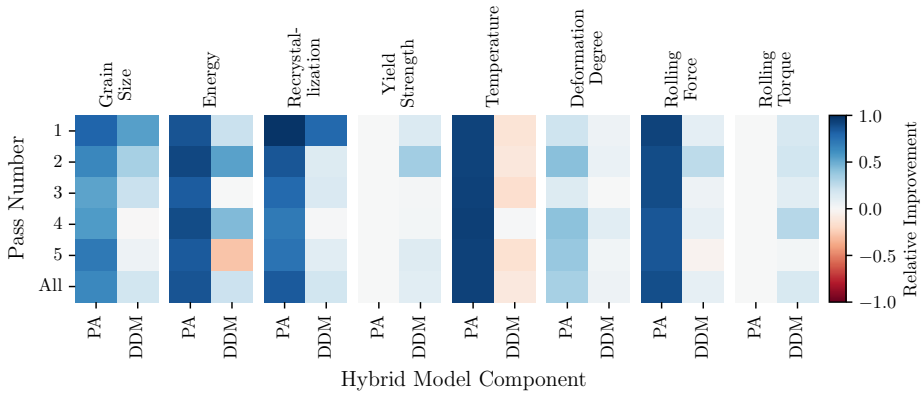


Figure 3: Relative improvement in MAE of hybrid model over each of its components: data-driven model (DDM) and physical approximation (PA). Yield strength and rolling torque are not computed by the physical approximation and the corresponding relative improvement of the hybrid model is hence denoted as 0.

$$R_t = 1 - \left(\frac{\alpha}{h_{max}} (\Delta h_t)^2 + \frac{\beta}{d_{max}} (\Delta d_t)^2 + \frac{\epsilon}{\epsilon_{max}} \sum_{k=1}^{t} e_k \right),$$

where h_t and d_t denote the distance to the goal height and grain size, respectively, while e_t corresponds to the used energy in the t-th pass. Each component of the equation is scaled by the associated maximal value h_{max}, d_{max} or e_{max}. The influence of each component can be controlled by adjusting α, β and ϵ. Note, that a reward of 1 is usually not possible, because the execution of a pass schedule requires energy.

As described in Section 3, current approaches to pass schedule design with reinforcement learning typically train agents directly on a FRM, which can be time-consuming and significantly slow down the development of new agent or environment designs. We replicate such an approach to serve as a baseline to which the pre-training and transfer approach is later compared. To facilitate this, we access the FRM RoCaT through a SaaS architecture and train an agent as described above for 2500 environment steps, which corresponds to a training time of 43.59 hours in our setup. As can be seen in in the learning curve displayed in Figure 6 (left), the agent shows good convergence behaviour in the achieved rewards, with some outliers due to exploration.

In contrast, training on the proposed hybrid model takes much less time. Again, we train an agent for 2500 environment steps, which corresponds to a training time of 0.17 hours on the hybrid model. As displayed in Figure 6 (middle), the agents converge to similar rewards on the hybrid model as on RoCaT. While the training time hence appears to be much reduced for similar results, the trained agent is expected to require some additional fine-tuning on RoCaT due to the approximate nature of the hybrid model.

To establish how much more efficient the overall approach consisting of pre-training and transfer phase is compared to sole training on RoCaT, the pre-trained agent is taken and trained on RoCaT for final adjustments. For this final phase, an additional 500 training steps are executed. As displayed in Figure 6 (right), the additional training

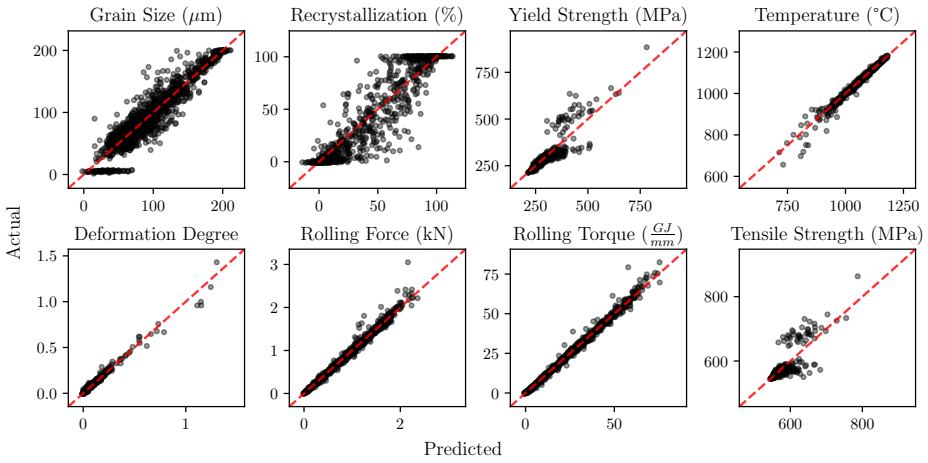


Figure 5: Scatter plots showing the discrepancies between predictions produced by our hybrid model and the actual values produced by the FRM for several different targets. Data points on the dashed red line indicate perfect predictions, while those above and below the line indicate under- and overestimation, respectively.

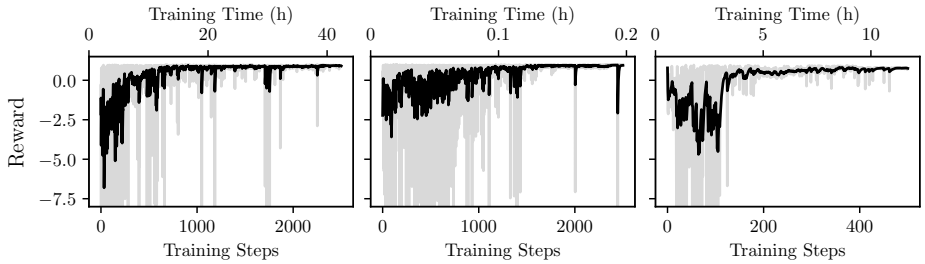


Figure 6: Learning curves of agents trained on RoCaT (left), on the hybrid model (middle), and transfer learning on RoCaT of an agent pre-trained on the hybrid model (right). The learning curves shown in black are exponentially weighted moving averages ($\alpha = 0.1$) with the original data being displayed in light grey. All curves are the average of three separately trained agents.

phase is needed, as the agent's rewards start significantly below those achieved on the hybrid model. The agent is nevertheless able to leverage its learned behaviour from the pre-training stage, which becomes apparent by the quickly rising rewards, which converge much more quickly compared to training from scratch. Overall, the combined training time of pre-training and transfer phase is 7.58 hours, while the training time of the baseline is 43.59 hours. It could be argued that the baseline training time is longer than necessary, as convergence behaviour can be observed starting at roughly half of the training duration. In any case, the two-stage training approach achieves a significant reduction in overall training time compared to the baseline.

6 Conclusion

We have proposed a hybrid model to approximate an FRM with the main purpose of using it to pre-train reinforcement learning agents for pass schedule design. Our model consists of a data-driven core, which is further augmented by a set of physical approximations of selected variables relevant for pass schedule design. The evaluation of the hybrid model shows that, on average, it outperforms each of its individual components.

While the predictions of our model are not as accurate as the FRM, its efficiency allows for quick training of reinforcement learning agents for pass schedule design. Although agents pre-trained on the hybrid model require some additional training on the FRM, the overall training time of this two-stage approach is significantly reduced while achieving similar results.

In the future, this reduction in training time will allow for more rapid development cycles of reinforcement learning agents. The agent used in our experiments only has access to a small, selected set of observations out of all the variables calculated by the FRM and our hybrid model. By incorporating additional observations, the performance of the agent can potentially be improved further in the future. Additionally, extending the observation and action spaces will particularly allow for the design of agents with more extensive generalization capabilities across hot rolling scenarios with varying initial geometries, materials, and material properties.

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612.

References

[1] World crude steel production from 2012 to 2022. `https://www.statista.com/statistics/267264/world-crude-steel-production/`. Accessed: 2024-01-03.

[2] D. Bland and H. Ford. The calculation of roll force and torque in cold strip rolling with tensions. *Proceedings of the Institution of Mechanical Engineers*, 159(1):144–163, 1948.

[3] H. Caglayan and J. Beynon. Slimmer sheffield leicester integrated model for microstructural evolution in rolling. In *Int. Conf. Modelling of Metal Rolling Processes. The Inst. of Materials, Imperial College, London*, pages 274–282, 1993.

[4] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[6] R. Hwang, H. Jo, K. S. Kim, and H. J. Hwang. Hybrid model of mathematical and neural network formulations for rolling force and temperature prediction in hot rolling processes. *Ieee Access*, 8:153123–153133, 2020. Publisher: IEEE.

[7] C. Idzik, J. Gerlach, J. Lohmar, D. Bailly, and G. Hirt. Application of Reinforcement Learning for the Design and Optimization of Pass Schedules in Hot Rolling. In *Congress of the German Academic Association for Production Technology*, pages 71–80. Springer, 2022.

[8] C. Idzik, D. Hilger, N. Hosters, M. Kemmerling, P. Niemietz, L. Ortjohann, J. Sasse, A. Serafeim, J. Wang, D. Wolff, et al. Decision support for the optimization of continuous processes using digital shadows. In *Internet of Production: Fundamentals, Applications and Proceedings*, pages 1–22. Springer, 2023.

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[10] M. R. Liebenberg. Autonomous agents for the world wide lab: artificial intelligence in the manufacturing industry, 2021.

[11] J. Lohmar, S. Seuren, M. Bambach, and G. Hirt. Design and application of an advanced fast rolling model with through thickness resolution for heavy plate rolling. In *2nd International Conference on Ingot Casting Rolling Forging*, 2014.

[12] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[13] S. Rath, P. Sengupta, A. Singh, A. Marik, and P. Talukdar. Mathematical-artificial neural network hybrid model to predict roll force during hot rolling of steel. *International Journal of Computational Materials Science and Engineering*, 2(01):1350004, 2013. Publisher: World Scientific.

[14] S. Rath, P. Talukdar, and A. Singh. Methodology of developing mathematical-ANN hybrid model based mill setup model for hot strip rolling. *International Journal for Computational Methods in Engineering Science and Mechanics*, 22(5):358–365, 2021. Publisher: Taylor & Francis.

[15] S. Ray. *Principles and applications of metal rolling*. Cambridge University Press, 2016.

[16] R. Roychoudhury and J. Lenard. A mathematical model of cold rolling-experimental substantiation. In *Proc. 1st Int. Conf. Techn. Plast*, pages 1138–1145, 1984.

[17] P. Saravanakumar, V. Jothimani, L. Sureshbabu, S. Ayyappan, D. Noorullah, and P. Venkatakrishnan. Prediction of mechanical properties of low carbon steel in hot rolling process using artificial neural network model. *Procedia Engineering*, 38:3418–3425, 12 2012.

[18] C. Scheiderer, T. Thun, C. Idzik, A. F. Posada-Moreno, A. Krämer, J. Lohmar, G. Hirt, and T. Meisen. Simulation-as-a-service for reinforcement learning applications by example of heavy plate rolling processes. *Procedia Manufacturing*, 51:897–903, 2020.

[19] J. A. Schey. Introduction to manufacturing processes, 2000. *Mc Craw Hill*, 2000.

[20] J. Wang, Y. Li, R. X. Gao, and F. Zhang. Hybrid physics-based and data-driven models for smart manufacturing: Modelling, simulation, and explainability. *Journal of Manufacturing Systems*, 63:381–391, 2022. Publisher: Elsevier.

[21] Q. Xie, M. Suvarna, J. Li, X. Zhu, J. Cai, and X. Wang. Online prediction of mechanical properties of hot rolled steel plate using machine learning. *Materials Design*, 197:109201, 2021.

[22] A. Özgür, Y. Uygun, and M.-T. Hütt. A review of planning and scheduling methods for hot rolling mills in steel production. *Computers Industrial Engineering*, 151:106606, 2021.

INTEGRATING CONTINUOUS-TIME NEURAL NETWORKS IN ENGINEERING: BRIDGING MACHINE LEARNING AND DYNAMICAL SYSTEM MODELING

Bernd Zimmering
Helmut-Schmidt-University
Holstenhofweg 85, 22043 Hamburg
bernd.zimmering@hsu-hh.de

Oliver Niggemann
Helmut-Schmidt-University
Holstenhofweg 85, 22043 Hamburg
oliver.niggemann@hsu-hh.de

24-01-2024

ABSTRACT

This paper examines the integration of Continuous-Time Neural Networks (CTNNs), including Neural ODEs, CDEs, Neural Laplace, and Neural Flows, into engineering practices, particularly in dynamical system modeling. We provide a detailed introduction to CTNNs, highlighting their underutilization in engineering despite similarities with traditional Ordinary Differential Equation (ODE) models. Through a comparative analysis with conventional engineering approaches, using a spring-mass-damper system as an example, we demonstrate both theoretical and practical aspects of CTNNs in engineering contexts. Our work underscores the potential of CTNNs to harmonize with traditional engineering methods, exploring their applications in Cyber-Physical Systems (CPS). Additionally, we review key open-source software tools for implementing CTNNs, aiming to facilitate their broader integration into engineering practices.

Keywords Cyber-Physical Systems · Dynamical Systems Modeling · Neural Ordinary Differential Equations

1 Introduction

In recent years, Machine Learning (ML) has emerged as a hyped area in scientific research as well as general public, demonstrating its prowess in fields like image and voice recognition, and text processing. The latest iterations of Chat GPT exemplify these advances. Despite its disruptive influence in these areas, ML's potential in the engineering domain has been comparatively untapped.

In the landscape of ML, recent popular models have been developed to address the unpredictable and varied nature of data in fields like image, text, and speech processing,

contrasting sharply with the structured and often deterministic data encountered in engineering disciplines, particularly those dealing with dynamical systems. ML Models such as the Variational Autoencoder (VAE) [7], Generative Adversarial Network (GAN) [9], and Normalizing Flow (NF) [23] are primarily designed for scenarios where the underlying data generation process is non-deterministic. VAEs leverage stochastic latent spaces, GANs rely on adversarial dynamics to generate data, and NFs use probabilistic transformations to model complex distributions. These characteristics make them particularly suited for domains like image and text processing, where data generation is inherently uncertain and diverse. Similarly, recent advancements in stable diffusion models [25], which iteratively refine a random noise input towards generating realistic images, have marked a significant breakthrough in the field of image synthesis.

However, machine learning in Cyber-Physical Systems (CPS) faces unique challenges such as integrating time dynamics, uncertainty estimation, and learning beneficial representations [20]. CPS, often modeled as *dynamical systems* (first defined by George D. Birkhoff [26]), evolve deterministically over time, typically represented by Ordinary Differential Equations (ODEs). This deterministic and continuous nature contrasts with the stochastic foundations of many ML models and is addressed in control engineering using models like ODEs and frequency domain representations [17].

In response to this, the past five years have seen the development of ML models that align more closely with the paradigms used in engineering for dynamical systems. One such breakthrough is the realization that the ResNet [14], a leading model for image classification, closely resembles the numerical approximation of an ODE using an Euler solver [27]. This insight has given rise to Neural Ordinary Differential Equations (Neural ODEs or NODE) [24], which are able to learn the underlying ODE of a sequence or time series.

While the ML research community explored the realms ODEs, concurrently, the integration of physical principles—captured by Partial Differential Equations (PDEs)—into machine learning frameworks was revolutionized by the introduction of Physics-Informed Neural Networks (PINNs) [22]. These networks adeptly embed known physical laws, such as the Navier-Stokes equations for fluid dynamics, into the predictive machinery of neural networks, offering a robust approach especially in contexts where data may be sparse. Nonetheless, our research primarily focuses on Neural ODEs, given their unique advantage of not necessitating a-priori physical knowledge for modeling time series of dynamical systems.

Although Neural ODEs closely mirror control engineering modeling practices, Recurrent Neural Networks (RNNs) such as Long Short-Term Memory (LSTM) [11] and the Gated Recurrent Unit (GRU) [6] remain predominant learning from CPS that contain dynamical systems [13,19,30]. This preference may stem from their analogy to nonlinear Delayed Difference Equations (DDEs) [29], a concept deeply rooted in engineering, coupled with their established presence (extending over 18 years as of 2024). Despite the extensive research and recognition of Neural ODEs as well as related models [15] in the ML domain – as highlighted in the Workshop "The Symbiosis of Deep Learning and Differential Equations III" at NeurIPS 2023[1] – their integration into ML applications within Cyber-Physical Systems (CPS) remains limited. This paper seeks to elevate the visibility of these models, collectively referred to as Continuous Time Neural Networks (CTNNs), and aims to broaden their application beyond conventional ML domains. The key contributions of this work are threefold:

[1] https://dlde-2023.github.io

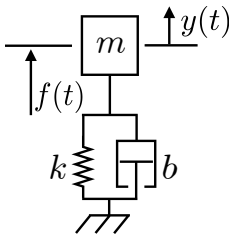


Figure 1: Spring-Mass-Damper System.

1. **Concise Review of Advanced Continuous ML Models:** This paper provides a detailed review of the latest continuous ML models, particularly those related to ODEs often employed in control engineering. We illustrate these concepts using a Spring-Mass-Damper system, which we model symbolically as an ODE, Transfer Function, and in closed form. This approach highlights the parallels with CTNNs.
2. **Comparison of CTNN and Engineering Models:** We present a concise comparison between CTNN approaches like Neural ODEs and modeling of dynamical systems engineering, focusing on their theoretical foundations.
3. **Overview of Open-Source Implementations:** We offer overview and guide of open-source tools for implementing these ML models in engineering applications.

The paper is organized as follows: We start with an analysis of dynamical system modeling, using a spring-mass-damper system as a case study. This is followed by an introduction to the mechanics of four CTNN models. Next, we explore the specifics of these ML models and compare their methodologies with conventional engineering approaches. Additionally, a detailed overview of essential open-source software tools for CTNNs is provided. The paper concludes with a summary and final reflections.

2 State of the Art and Background

2.1 Modeling in Controls Engineering

Modeling in control engineering often revolves around dynamical systems that adhere to conservation laws like energy conservation of sum of forces. In the case of a mechanical system the internal energy remains constant unless external forces are applied or dissipative forces, such as friction, are present. These dynamic interactions, are commonly described by sets of ODEs. Alongside energy conservation principles, which are typically well-understood, a single ODE can be formulated to describe the movement of a dynamical system under specific conditions. In engineering, modeling of dynamical systems is traditionally grounded in first principles (physical laws). Utilizing a Spring-Mass-Damper (SMD) system, depicted in Figure 1, serves as a practical example to illustrate the derivation of symbolic models across various mathematical domains, including ODEs, the Laplace domain, and the time domain. Please note that the SMD system is chosen for its illustrative value, simplifying the explanation of symbolic model derivation. Alternative examples, such as RLC circuits, could similarly demonstrate these concepts. In subsequent chapters, we will see how CTNNs model systems in a similar fashion, but relying on data rather than prior knowledge.

In the Spring-Mass-Damper (SMD) system, the mass experiences several forces. The force exerted by the spring is $-ky(t)$, where k is the spring constant and $y(t)$ is the displacement from the equilibrium position. The damping force, due to the damper, is $-b\dot{y}(t)$, with b being the damping coefficient and $\dot{y}(t)$ the velocity. Additionally, the system might be subject to an external force, represented by $f(t)$.

To derive the motion equation for the SMD system, we apply Newton's second law, which is $f = m \cdot \ddot{y}$. In this system, the forces include the spring force, the damping force, and any external force. By summing these forces, the equation of motion is obtained as follows:

$$m\ddot{y}(t) + b\dot{y}(t) + ky(t) = f(t) \tag{1}$$

Here, $\ddot{y}(t)$ is the acceleration of the mass. This equation describes the dynamic behavior of the SMD system, incorporating the effects of spring force, damping force, and external force.

While this form of a system can be challenging to analyze in-depth, the **State Space Model (SSM)** approach is often used, which describes a system as a combination of first-order ODEs. This is analogous to the latent space $\mathbf{z} \in \mathbb{R}^M$ used in Machine Learning, such as in an Autoencoder. However, in many cases, the state space is not necessarily lower-dimensional than the observation space. To apply SSM to our SMD system, we define the state variables as $z_1(t) = y(t)$ and $z_2(t) = \dot{y}(t)$. This allows us to rewrite Equation (1) as follows:

$$\dot{z}_1(t) = z_2(t) \tag{2a}$$

$$\dot{z}_2(t) = \frac{1}{m}\left[f(t) - bz_2(t) - kz_1(t)\right] \tag{2b}$$

Here, $\dot{z}_1(t)$ corresponds to velocity, and $\dot{z}_2(t)$ to acceleration. The SSM effectively encapsulates the system's dynamics in a two-dimensional state space, representing the primary energy storage components: the kinetic energy of the mass (in z_2) and the potential energy in the spring (in z_1).

Another method to analyze Equation (1) is the Laplace transform [28]. It enables the analysis of system properties and behavior without the necessity for time-domain simulations. This powerful tool transforms differential equations into algebraic equations, simplifying the analysis significantly. Applying the Laplace transform to Equation (1), we denote the Laplace transform of a function $g(t)$ as $\mathcal{L}\{g(t)\} = G(s)$, where $s \in \mathbb{C}$ is the complex frequency variable. For the SMD system, the transformed equations become:

$$\begin{aligned} \mathcal{L}\{m\ddot{y}(t)\} + \mathcal{L}\{b\dot{y}(t)\} + \mathcal{L}\{ky(t)\} &= \mathcal{L}\{f(t)\} \\ ms^2Y(s) - msy(0) - m\dot{y}(0) & \\ +bsY(s) - by(0) + kY(s) &= F(s) \end{aligned} \tag{3}$$

Here, $Y(s)$ and $F(s)$ are the Laplace transforms of the displacement $y(t)$ and the external force $f(t)$, respectively. The terms $sy(0)$ and $\dot{y}(0)$ represent the initial conditions in the Laplace domain. We can further rearrange Equation (3) to get insights representing the system's response in the frequency domain:

$$s^2Y(s) + \frac{b}{m}sY(s) + \frac{k}{m}Y(s) = \frac{1}{m}F(s) + sy(0) + \dot{y}(0) + \frac{b}{m}y(0) \tag{4}$$

$$H(s) = \frac{Y(s)}{F(s)} = \frac{1}{s^2 + \frac{b}{m}s + \frac{k}{m}} + \frac{sy(0) + \dot{y}(0) + \frac{b}{m}y(0)}{s^2 + \frac{b}{m}s + \frac{k}{m}} \tag{5}$$

In control engineering, when initial states $y(0)$ and $\dot{y}(0)$ are set to zero, the focus shifts to the Transfer Function (TF) $H(s)$. The system's behavior is largely determined by the nature of its poles, derived from the quadratic formula:

$$s = \frac{-\frac{b}{m} \pm \sqrt{\left(\frac{b}{m}\right)^2 - 4\frac{k}{m}}}{2}. \tag{6}$$

These poles, particularly in the underdamped case ($(\frac{b}{m})^2 - 4\frac{k}{m} < 0$), lead to complex conjugate roots and oscillatory system behavior. The solution for an underdamped system without external force ($f(t) = 0$) is:

$$y(t) = e^{-\frac{b}{2m}t}\left(y_0 \cos(\omega t) + \frac{v_0 + \frac{b}{2m}y_0}{\omega}\sin(\omega t)\right), \tag{7}$$

where $\omega = \sqrt{\frac{k}{m} - \left(\frac{b}{2m}\right)^2}$ represents the damped natural frequency. This formulation succinctly captures the characteristic damped oscillatory motion of the system.

2.2 Continuous Time Neural Networks

Following our exploration of the SMD system, which showcased equation derivation in the ODE, Time, and Laplace domains, we now delve into CTNN models. These models mirror the mathematical frameworks used to derive the symbolic models for the SMD system.

Neural Ordinary Differential Equations (NODE), as introduced by Chen et al. [24], merge the realms of machine learning and differential equations in a novel way. The essence of NODE lies in solving an Initial Value Problem (IVP) through the use of neural networks. Here, the IVP is characterized by finding a function that concurrently satisfies a *learned* differential equation and an initial condition.

In the NODE framework, system dynamics are encapsulated within a latent space $\mathbf{z} \in \mathbb{R}^M$ (similar to the SSM). This space is a transformation of the input sequence $\{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$ where $\mathbf{x} \in \mathbb{R}^N$. The evolution of this system through time is governed by a differential equation defined by a neural network:

$$\dot{\mathbf{z}}(t) = f_{\text{ODE}}(\mathbf{z}(t), t; \theta_{\text{ODE}}). \tag{8}$$

Here, $\dot{\mathbf{z}}(t)$ signifies $\frac{d\mathbf{z}}{dt}$, the rate of change of the latent state, while $f_{\text{ODE}} : \mathbb{R}^M \to \mathbb{R}^M$ is a neural network function parameterized by θ_{ODE}. An interesting aspect of NODE is its use of a VAE approach to sample the initial state $\mathbf{z}_0$. This approach ensures the latent space remains efficiently dimensioned, employing just the necessary states for representation, and additionally permits the use of the trained model as a generative tool in inference phases when needed. The VAE-like step involves an RNN running backwards through the observations $\mathbf{x} \in \mathbb{R}^N$, which outputs the parameters of a normal distribution $\mathcal{N}(\mu_{\mathbf{z}_{t_0}}, \sigma_{\mathbf{z}_0})$. The initial state $\mathbf{z}_0$ is then sampled from this distribution:

$$\mu_{\mathbf{z}_0}, \sigma_{\mathbf{z}_0} = \text{RNN-Encoder}(\{\mathbf{x}_T, \ldots, \mathbf{x}_0\}), \tag{9}$$

$$\mathbf{z}_0 \sim \mathcal{N}(\mu_{\mathbf{z}_0}, \sigma_{\mathbf{z}_0}). \tag{10}$$

The development of $\mathbf{z}(t)$ from this initial state is computed over K discrete time intervals, where K represents the number of desired steps in a forecast scenario:

$$\mathbf{z}(t_{i+1}) = \mathbf{z}(t_i) + \int_{t_i}^{t_{i+1}} f_{\text{ODE}}(\mathbf{z}(\tau), \tau; \theta_{\text{ODE}})d\tau,$$
$$\text{for } i = 0, 1, \ldots, T + K - 1. \tag{11}$$

Each latent state in the sequence $\{z_0, \ldots, z_T\}$ is decoded pointwise by a feedforward neural network to generate the output $\hat{\mathbf{y}}$:

$$\hat{\mathbf{y}}_i = \text{Decoder}(\mathbf{z}_i). \tag{12}$$

Chen et al.'s incorporation of ODE solvers (e.g. adaptive Runge-Kutta [3]) into the PyTorch framework was key in enabling backpropagation through these solvers, a crucial aspect for training NODE models. The primary challenge here was the substantial memory cost associated with backpropagation in complex or extended dynamical systems. The *adjoint method* proposed by Chen et al. provides a solution to this problem, improving memory efficiency while maintaining gradient accuracy.

Neural Controlled Differential Equations (CDE), as proposed by Kidger et al. [16], extend the NODE concept to handle irregularly observed time series by directly integrating observations into the differential equations. The CDE framework modifies the NODE model by introducing a multiplicative term to the differential equation:

$$\dot{\mathbf{z}}(t) = f_{\text{CDE}}(\mathbf{z}(t); \theta_{\text{CDE}}) \cdot \tilde{\dot{\mathbf{x}}}(t), \tag{13}$$

where $f_{\text{CDE}} : \mathbb{R}^M \to \mathbb{R}^{M \times N}$ is a neural network. To handle the typically unknown derivatives of observations, natural cubic splines are used to smoothly interpolate the data $\mathbf{x}$, allowing the computation of $\tilde{\dot{\mathbf{x}}}(t)$.

CDEs are particularly adept at processing time series with irregular observations but are not designed for forecasting scenarios where future inputs are unknown. Instead, they are more suitably employed as a replacement for the RNN encoder in Equation (9), enhancing the processing of past observed data.

Neural Flows by Biloš et al. [2] innovatively learn the flow of states, which is essentially the solution of an IVP, without the need to solve differential equations. Among various models, they introduce the ResNet-Flow, which satisfies key properties of an ODE solution: I) $F(0, x_0) = x_0$, ensuring that the flow aligns with the initial state at $t = 0$, and II) $F(t, \cdot)$ is invertible for all t, guaranteeing the uniqueness of the solution with respect to the initial condition.

The Resnet-Flow as used in their continous-time latent variable model (Encoder-Decoder approach similar to NODE) is defined as:

$$F(t, \mathbf{z}) = \mathbf{z}_0 + \varphi(t) g(t, \mathbf{z}_0; \theta_{\text{flow}}), \tag{14}$$

where $\mathbf{z}_0$ denotes the initial state. In this model, $\varphi : \mathbb{R} \to \mathbb{R}^N$ is a predetermined function (i.e. a hyperparameter) and $g : \mathbb{R}^{N+1} \to \mathbb{R}^N$ is a neural network designed to learn the flow dynamics, parameterized by θ_{flow}. To satisfy condition I) and ensure that the flow aligns with the initial state at $t = 0$, the function $\varphi(t)$ is chosen as $\varphi(t) = \tanh(t)$, which evaluates to zero when $t = 0$. For addressing condition II), which guarantees the invertibility of $F(t, \cdot)$ for all t, the network g is designed as a contractive neural network. This design ensures that the Lipschitz constant of g is less than 1, facilitating the maintenance of invertibility in the flow dynamics.

Neural Laplace, by Holt et al. [12], leverages a neural network for approximating a TF in the Laplace domain to predict time series dynamics. This approach parallels the capabilities of Neural ODEs but does not require an ODE solver, thereby simplifying the process and enhancing computational efficiency.

Central to the Neural Laplace model is the learning of the transfer function $H(s)$ for $\mathbf{s} \in \mathbb{C}^D$. This is achieved using a neural network $g(\mathbf{p}, \mathbf{s}; \theta_L)$, where $\mathbf{p} \in \mathbb{R}^D$ represents the initial conditions similar to those in Equation (5).

The key components of the Neural Laplace model include:

- An **Encoder** $f(\mathbf{x}_0, \ldots, \mathbf{x}_T - k; \theta_{\text{enc}})$ that encodes time series data into initial conditions $\mathbf{p}$ for the Laplace function $H(\mathbf{s})$.
- A **Neural Network** $g(\mathbf{p}, \mathbf{s}; \theta_L)$ designed to model the Laplace function $H(\mathbf{s})$.
- The **Inverse Laplace Transform (ILT)** algorithm that uses $g(\mathbf{p}, \mathbf{s}; \theta_L)$ to numerically construct a time series for given timesteps $\{t_0, \ldots, t_T + K\}$ where K is the number of forecast time steps. Please note that similar to ODE solvers there exist a few methods, that perform task depended. [12] uses Fourier Series Inverse which uses bi-linear approximation to solve the ILT integral [10]

The Neural Laplace model's notable advantage is its ability to operate without the need for an ODE solver. This feature, along with its domain-centric approach, allows it to efficiently handle a diverse range of differential equations, including forced ODEs and ODEs with deadbands, making it a robust tool for complex system analysis and modeling.

3 Comparative Analysis of Continuous-Time Neural Networks and Engineering Modeling Approaches

As demonstrated in the previous section, CTNN bear similarities to the engineering modeling paradigms for dynamical systems, particularly in their shared approach to solving an IVP. The NODE model, utilizing a system of first-order ODEs, closely mirrors the SSM in engineering. This similarity is especially evident in the concept of modeling system dynamics within a latent space governed by first-order ODEs, akin to the SSM approach. However, notable differences exist:

1. The initial state in NODE, and consequently the properties of the latent space, are determined by a VAE approach. This involves sampling and regularizing the initial values to adhere closely to the predefined prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This method contrasts with SSM, where the initial state typically corresponds to a tangible physical state, like velocities or movements.
2. NODE models typically represent the dynamics of a homogeneous ODE, lacking an explicit input function $f(t)$ that is commonly present in SSM. This absence means that NODE's dynamics are derived from a homogeneous system, differing from the SSM approach, where external inputs or forces (represented by $f(t)$) are often integral to the system's behavior and analysis.

Considering the second difference, Neural CDEs align more closely with SSM in engineering, as they allow the incorporation of excitations beyond just observations $\mathbf{x}$ in the machine learning context. However, the definition of f_{CDE} in CDEs, transforming a vector to a matrix, diverges from the typical SSM model. Additionally, while the excitation force in the SSM model, as per Equation (2), is additive, in CDEs (Equation (13)) it is multiplicative, a derivation from the field of rough analysis in mathematics. Despite the nomenclature similarity to controlled differential equations in control theory, which is $\dot{x}(t) = f(x(t), u(t), t)$, Neural CDEs are based on a different theoretical foundation, limiting their direct applicability to model forced dynamics in the same manner as engineering models.

Both Neural CDEs and NODEs utilize numerical ODE solvers frequently used in engineering, such as adaptive Runge-Kutta methods [3]. However, these solvers are often avoided in engineering due to their computational intensity.

In engineering, direct modeling of system responses to specific signals is uncommon, as it its usefullness for unknown excitation signals is limited. However, Neural Flows follow this paradigm, as they use a neural net that is satisfying the conditions of an IVP. Regarding the relevance and generalizability of Neural Flows to other types of signal remain a question in engineering contexts, where it is not always possible to assume the presence of such excitations in the dataset.

Neural Laplace models are closely related to transfer functions used in control engineering. Their capability to extend ODE models by easily modeling transport delay times — a common feature in control engineering—positions them as superior to SSM, particularly when the system's ODE is assumed to be linear. Unlike typical engineering practices where initial conditions are often neglected or assumed to be zero, Neural Laplace models utilize initial conditions determined by an encoder. Moreover, the use of the ILT algorithm is atypical for control engineering, where solutions to IVPs are usually obtained through partial fraction decomposition and laplace correspondence tables. Numerical ILT is not widely known or commonly used in control engineering. Nonetheless, as the neural network in Neural Laplace models learns a transfer function, it opens up possibilities for deriving system properties like poles and zeros from data.

In the context of applying these CTNN models to a specific engineering case, such as the SMD system, their distinct algorithmic approaches and the resulting learned equations offer a clear comparative perspective. Table 1 summarizes how each CTNN model would approach the SMD system, highlighting their unique methodologies and outcomes.

For instance, the NODE model would parameterize the ODE of the SMD system and operate under the assumption of a homogeneous system, omitting explicit input functions. In contrast, CDEs would modify this approach by incorporating observations in a multiplicative manner. This is different from the additive nature of excitations commonly seen in engineering SSMs. Neural Laplace models, aligning with control engineering's transfer functions, bring a novel perspective to modeling SMD systems by focusing on the system's Laplace domain properties. Lastly, Neural Flows, deviating from the traditional direct modeling of system responses in engineering, offer a unique approach by learning the state flow without directly solving differential equations.

Table 1: Comparison of Continuous-Time Neural Network Models Applied to the Spring-Mass-Damper System

CTNN Model	**Algorithmic Approach**	**Learned Equation for SMD System**
NODE	Utilizes a neural network to parameterize an ODE.	SSM of Eq (2) with $f(t) = 0$ including $z_1(0) = y(0)$; $z_2(0) = \dot{y}(0)$
CDE	Enhances NODE by incorporating observations into the ODE.	SSM similar to Eq. (2), but with multiplicative input of $\dot{f}(t)$
Neural Laplace	Uses the Laplace domain for solving ODEs and learning the transfer function.	Transfer Function $H(s)$ of Eq. (5) including initial conditions $\mathbf{p} = [y(0), \dot{y}(0)]^T$
Neural Flows	Learns the flow of states without directly solving differential equations.	Solution of IVP for $f(t) = 0$ like Eq.(7)

4 Software Libraries

The convergence of CTNNs with engineering paradigms necessitates quantitative evaluations of their performance and limitations. As these networks rely on complex theories and algorithms, existing software libraries are indispensable for practical applications. Table 2 presents a summary of prominent open-source software for CTNN studies. It's important to note that 'GitHub Stars' should be seen merely as an indicator of a library's popularity and recognition within the developer community, rather than a definitive metric for its usage or quality.

Table 2: Summary of Software Libraries for Neural ODEs, Neural Laplace, and Neural Flows

Library	**Reference**	**Functionality**	**Language& ML Framework**	**GitHub Stars**
`torchdiffeq`	[5]	ODE Solvers	Python Pytorch	4900
`torchdyn`	[21]	ODE Solvers	Python Pytorch	1200
`torchode`	[18]	ODE Solvers	Python Pytorch	<200
`neurodiffeq`	[4]	ODE Solvers PDE Solvers	Python Pytorch	<600
`diffrax`	[15]	CDE Solvers, ODE Solvers	Python Jax	1100
`torchlaplace`	[12]	Neural Laplace, ILT Solvers	Python Pytorch	<100
Neural Flows	[2]	Neural Flows	Python Pytorch	<100

`torchdiffeq`, `torchdyn`, and `torchode`, all Python libraries utilizing the PyTorch framework [1], offer similar functionalities in solving ODEs. `torchdiffeq` is notable for its integration with Chen et al.'s work [5], while `torchdyn` focuses on applying NODEs in neural network depths akin to ResNet. `torchode` enhances solver efficiency through JIT-Compilation and parallel processing.

`neurodiffeq` caters to both ODE and Partial Differential Equation (PDE) solutions, emphasizing boundary conditions integration. `diffrax`, leveraging the JAX framework [8], offers speed-optimized solutions for CDEs and ODEs and is maintained by Kidger [15].

`torchlaplace` specializes in the Neural Laplace algorithm, offering advanced ILT solvers [12]. *Neural Flows*, though not a formal package, provides practical code for implementing Neural Flow models [2].

5 Conclusion

In this paper, we have effectively bridged the divide between CTNNs and traditional engineering methodologies for modeling dynamical systems. By examining the mechanics of CTNNs, especially Neural ODEs, CDEs, Neural Laplace, and Neural Flows, against systems like the spring-mass-damper model, we have underscored the parallels between these ML models and advanced dynamical system modeling in engineering. CTNNs offer significant potential in scenarios where symbolic modeling of dynamical systems is intricate, providing a robust avenue for developing reliable ML-based models. These models, grounded in theoretical principles akin to those in engineering, enhance trust in their modeling processes. In addressing the limited utilization of CTNNs in engineering, our comprehensive guide and recommendations on open-source tools aim to encourage the engineering community to explore and confirm the practicality of these ML algorithms in real-world engineering tasks.

Acknowledgments

We would like to acknowledge the assistance of AI tools, including ChatGPT, for their support in refining the readability of this manuscript. Their contributions complemented our efforts, helping to make the paper more accessible.

References

[1] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[2] M. Biloš, J. Sommer, S. S. Rangapuram, T. Januschowski, and S. Günnemann. Neural flows: Efficient alternative to neural odes. In *Neural Information Processing Systems*, 2021.

[3] J. C. Butcher. A history of runge-kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.

[4] F. Chen, D. Sondak, P. Protopapas, M. Mattheakis, S. Liu, D. Agarwal, and M. Di Giovanni. Neurodiffeq: A python package for solving differential equations with neural networks. *Journal of Open Source Software*, 5(46):1931, 2020.

[5] R. T. Q. Chen. torchdiffeq, 2018.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. 2014.

[7] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[8] R. Frostig, M. Johnson, and C. Leary. Compiling machine learning programs via high-level tracing. 2018.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc, 2014.

[10] Harvey Dubner and Joseph Abate. Numerical inversion of laplace transforms by relating them to the finite fourier cosine transform. *J. ACM*, 15:115–123, 1968.

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] S. I. Holt, Z. Qian, and M. van der Schaar. Neural laplace: Learning diverse classes of differential equations in the laplace domain. In K. Chaudhuri, S. Jegelka, Le Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8811–8832. PMLR, 2022.

[13] M. Jafari, A. Kavousi-Fard, M. Dabbaghjamanesh, and M. Karimi. A survey on deep learning role in distribution automation system: A new collaborative learning-to-learning (l2l) concept. *IEEE Access*, 10:81220–81238, 2022.

[14] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[15] P. Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[16] P. Kidger, J. Morrill, J. Foster, and T. Lyons. Neural controlled differential equations for irregular time series. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6696–6707. Curran Associates, Inc, 2020.

[17] J. Lunze. *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen*. Springer Vieweg, Berlin, Heidelberg, 12th ed. 2020 edition, 2020.

[18] Marten Lienen and Stephan Günnemann. torchode: A parallel ode solver for pytorch.

[19] B. C. Mateus, M. Mendes, J. T. Farinha, R. Assis, and A. M. Cardoso. Comparing lstm and gru models to predict the condition of a pulp paper press. *Energies*, 14(21):6958, 2021.

[20] O. Niggemann, B. Zimmering, H. Steude, J. L. Augustin, A. Windmann, and S. Multaheb. Machine learning for cyber-physical systems. In B. Vogel-Heuser and M. Wimmer, editors, *Digital Transformation: Core Technologies and Emerging Topics from a Computer Science Perspective*, pages 415–446. Springer Berlin Heidelberg, Berlin, Heidelberg, 2023.

[21] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, and S. Ermon. Torchdyn: Implicit models and neural numerical methods in pytorch.

[22] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[23] D. Rezende and S. Mohamed. Variational inference with normalizing flows. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France, 2015. PMLR.

[24] C. Ricky T. Q., R. Yulia, B. Jesse, and D. David Kristjanson. Neural ordinary differential equations. In *NeurIPS*, 2018.

[25] R. Robin, B. Andreas, L. Dominik, E. Patrick, and O. Björn, editors. *High-Resolution Image Synthesis with Latent Diffusion Models*, 2022.

[26] T. Roque. Stability of trajectories from poincaré to birkhoff: approaching a qualitative definition. *Archive for History of Exact Sciences*, 65(3):295–342, 2011.

[27] L. Ruthotto and E. Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62(3):352–364, 2020.

[28] J. L. Schiff. *The Laplace Transform: Theory and Applications*. Springer eBook Collection Mathematics and Statistics. Springer, New York, NY, 1999.

[29] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404(8):132306, 2020.

[30] K. Zarzycki and M. Ławryńczuk. Lstm and gru neural networks as models of dynamical processes used in predictive control: A comparison of models developed for two chemical reactors. *Sensors (Basel, Switzerland)*, 21(16), 2021.

TOWARDS THE GENERATION OF MODELS FOR FAULT DIAGNOSIS OF CPS USING VQA MODELS

Silke Merkelbach[1], Alexander Diedrich[2], Sebastian von Enzberg[3], Oliver Niggemann[2], and Roman Dumitrescu[1]
[1]Fraunhofer Institute for Mechatronic Systems Design IEM, Paderborn, Germany
{firstname.lastname}@iem.fraunhofer.de
[2]Helmut-Schmidt-University, Hamburg, Germany
{firstname.lastname}@hsu-hh.de
[3]Hochschule Magdeburg-Stendal, Magdeburg, Germany
sebastian.von.enzberg@h2.de

ABSTRACT

In many use cases cyber-physical systems are employed to produce products of small batch sizes as efficiently as possible. From an engineering standpoint, a major drawback of this flexibility is that the architecture of the cyber-physical system may change multiple times over its lifetime to accommodate new product variants. To keep a cyber-physical system working normally it has become common to employ fault diagnosis algorithms. These algorithms partly rely on physical first-principles models that need to be updated when the architecture of the system changes which usually has to be done manually. In this article we present a practical approach to obtain such a first-principles model through evaluating piping and instrumentation diagrams (P&IDs) with visual questions answering (VQA) models. We demonstrate that it is possible to leverage VQA models to construct physical equations which are a preliminary stage for the creation of models suitable for fault diagnosis. We evaluate our approach on OpenAIs GPT-4 Vision Preview model using a P&ID we created for a benchmark water tank system. Our results show that VQA models can be used to create physical first-principles models.

Keywords Visual Question Answering · Large Language Models · Fault Diagnosis · First-Principles Models · Application of LLMs

1 Introduction

Cyber-physical systems aim to perform some function towards achieving a certain goal. In the area of cyber-physical production systems this goal is often the efficient production of goods at high volume and low costs. Unfortunately, some of the components of a

cyber-physical system may break down over time. For example, through wear and tear valves may get stuck, pipes may get leaks, and electronic components may get damaged. In many of these cases production has to be stopped until the defect can be mitigated [11]. Such mitigation is done through the early detection of faults using automated fault diagnosis methods. The goal of the research field of fault diagnosis is to provide algorithms and automated methods to find faults as soon as they occur and optimally to suggest mitigation strategies [3].

The challenge is that usually faults are diagnosed using a model-based approach. But obtaining the correct models that are also adaptive to potential changes to the cyber-physical system over its lifetime is often very expensive. In the past, these models had to be created by experts and needed to be adapted to different operating modes (i.e. ramp-up, production of different products etc.), a multitude of different product variants, and to changes in architecture [25]. Such manual adaptations were usually prohibitively expensive and limited adoption of diagnosis methods. Nowadays, several approaches exist that attempt to learn those models automatically [12, 10, 34, 33, 16] through statistical, control theoretic, and machine learning approaches. But learning models automatically from data always has the drawback that already known facts that can easily be described by experts, may be unaccounted for. In addition, some connections and rare events may not be covered by the learned model if they do not occur in the training data. It is thus highly relevant to develop approaches that take expert knowledge into account in order to automatically create suitable models for diagnosis [35].

Usually, the models needed for diagnosis tasks are referred to as system descriptions (SD). System descriptions can be specified through propositional logic, predicate logic, or even structural equations [13]. They usually express functional dependencies between components, signals and/or variables. In classical consistency-based diagnosis [8] SDs are described using propositional logic. But previous work found that obtaining propositional logic SDs automatically from data has several drawbacks [12, 10]. Others create system descriptions using structural equation models [13, 15], but those require a significant amount of expert knowledge. The challenge we address in this article is to create SDs that capture the physical relationships and dependencies within a cyber-physical system out of piping and instrumentation diagrams (P&IDs). P&IDs are used in the process industry to visualize processes with fluids and gases. We want to use an image with a P&ID as basis to obtain a model that tells us that if a valve is stuck, we will see that the flow through some pipe may not change as expected.

In this article we present a method that uses large language models with vision capabilities, also known as visual question answering (VQA) models, to create physical models from given images containing P&IDs. We propose a two-step solution where we first create a table with connections between the elements in the image and afterwards derive physical equations from it. The resulting model can in turn be used to create analytical redundancy relations (ARRs) or structural models to compare the predicted system behaviour to actual behaviour. Discrepancies between predictions and observations can then be used for diagnosis.

With our presented solution it is possible to automatically create physical first-princples models from P&IDs as basis for system descriptions and, together with actual data, perform diagnosis of cyber-physical production systems from process industry. Overall, our approach shows how to leverage large-language models capable of processing image data to diagnose adaptive and constantly changing system architectures of cyber-physical production systems.

Our contribution is the following: We present a novel approach to generate physical first-principles models of cyber-physical production systems from P&IDs by leveraging a VQA model, namely ChatGPT 4 Vision Preview [32].

We validate our approach in experiments on a P&ID we created for the four-tank system from the benchmark presented by Balzereit et al. [2]. The P&ID uses the symbols of the standard DIN EN ISO 10628-2 [9].

2 Related Work

The idea of identifying causal dependencies for physical modelling was introduced by the works of Forbus [14] and de Kleer et al. [7]. Recently, Nielsen et al. [30] have described a causality detection approach for Multilevel Flow Modelling, although compared to our approach they do not deal with fault diagnosis. Jaber et al. [19] have presented an approach to improve causal reasoning and modelling, and investigated the use of partial ancestral graphs. Chao et al. [6] have tried to mitigate the problem of little available data by presenting a framework which uses either physics-based models grounded on first principles, or a convolutional deep neural network approach. Cao et al. [5] and also Vokuvić [36] provide a review of causal discovery for industrial processes. They identify that, among several other methods, Granger Causality is a useful method to obtain causal relationships. But these do not correspond to physical models. Frisk et al. [16] have presented work to extract analytical redundancy relations automatically, without using large-language models. But these do not work with image data. An active research field between causality research and fault diagnosis are bondgraph approaches. Gao et al. [17] provide a good overview over bondgraphs and other modelling and model-based diagnosis methods. Recently, Borutzky [4] and Khan et al. [24] have introduced new residual-based approaches for fault diagnosis. In consistency-based diagnosis Matei et al. [28] have published articles about diagnosing physical systems using differential equations. Physical system diagnosis has also recently been addressed by Muškardin et al. [29] and by Yucesan et al. [38]. Kolb et al. [26] have presented a method to learn satifiability modulo theory (SMT) expressions as system models. The closest approach to our method was introduced by Krysander and Nyberg [27] and Gelso et al. [18] who presented the structural analysis of tank systems for fault diagnosis. We conclude that, while many works are available that require first-principles models, few works express the need to learn models automatically. And none do so in the context of fault diagnosis through the use of large-language models.

Large language models have not been broadly adopted by the fault diagnosis community yet. Kang et al. [20] have presented the usage of large language models for software fault localization. The same is true for the work of Wu et al. [37]. Balhorn et al. [1] have used large language models to correct possibly faulty P&ID diagrams, but have not attempted to generate any kinds of models from the diagrams. Ogundare et al. [31] have analysed the resilience of large language models to create system models. But their models are limited to single equations that have no automatic dependencies between each other. Kato et al. [23, 22, 21] extracted equations from a large number of scientific documents and created physical models out of them by judging their equivalence using a pre-trained large language model and defining requirements the physical model has to meet. But they neither use P&IDs nor do they create models for fault diagnosis.

3 Creating Physical First-principles Models using VQA Models

Our method to generate physical models out of P&IDs using VQA models consists of two steps: i) Extracting the structure of the system from the P&ID image data and ii) Creating physical first-principles models. We choose a two-step solution because it proved to be helpful for large language models if the problem is decomposed into smaller pieces [39]. In the first step, we input an image with a P&ID and a prompt including context and task into a VQA model, getting the structure of the system in tabular form as output. In the second step, we input the generated table with the system structure, again the P&ID, and a prompt including assumptions for and context about the resulting model. An overview of our approach in shown in Figure 1.

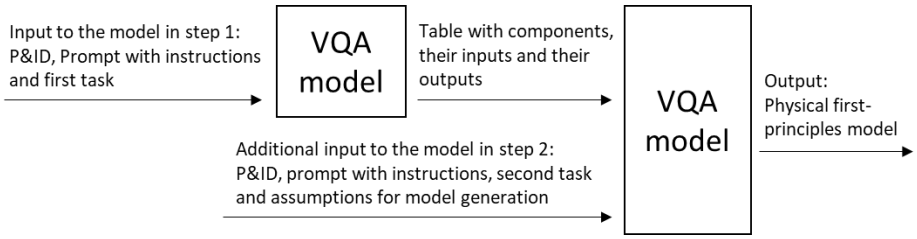


Figure 1: Two-step method to generate physical first-principles models for diagnosis out of P&IDs and additional, textual information

3.1 Extract the structure of the system from the P&ID (step 1)

In this step we generate the structure of the system with its components, their inputs, and their outputs from a P&ID. The P&ID image shows symbols according to DIN EN ISO 10628 [9]. The generated structure is a table that describes the adjacency of the components within the P&ID in electronically usable form. I.e. each row within the table shows a specific component and how it is connected to other components. By creating the table with the components and their connections first, the model has to focus on only one task instead of doing multiple tasks at a time which proved to create better results in the past when using large language models [39]. Our input to the VQA model, besides the P&ID image, are instructions describing the context, and a user message with the concrete task.

The instructions are optional parameters to the GPT-4 model and describe the role the model should take. We specified that the model should take the role of an engineer and included information about the structure of the image and how it should be read (i.e. that it should follow arrows and read from left to right, for example). We also specified the output format in the form of a comma separated values (.csv) file. The user message contains the concrete task that should be fulfilled by the model and is the actual prompt that most people know from using large-language models. It specifies the design of the table, to output the term 'unclear' if components could not be recognized, and to output only the table without any further explanations.

3.2 Create physical first-principles model from the connection table generated in the first step (step 2)

The second step of our method generates the physical model consisting of modelling equations for the identified components out of the following inputs: the connection table which was created in step 1, again the P&ID, the instructions, and the user message

with the task and assumptions for model generation. The connection table from step 1 contains the connections between the components and is used to support the model building process. The P&ID is given to the VQA model as reference in case that relevant information is missing in the textual description. The instructions again contain the role the VQA model should take. However, we changed the instructions to include the different input that is used in this step. The user message contains the concrete task that should be fulfilled by the VQA model and specifies the type of model that is needed for the given use case, such as a dynamical model. In addition, the user message includes assumptions that should be made for the creation of the physical first-principles model. The assumptions involve information about the geometry of the system and other information regarding the system, such as insulation or energy losses. They also involve some equations which should be used for modeling. The equations are given to the VQA model to be less dependent on the training data the model has seen so far.

4 Evaluation

To evaluate our approach we did one experiment for the first step and one for the second step. In the following we first describe the design of experiment, then present the results and provide a brief discussion about the limitations of our approach.

4.1 Experiment Design

We evaluate our approach using OpenAI's gpt-4-1106-vision-preview model [32] as VQA model which was accessed via the API interface using Python. As an application example, we took the four-tank-system S3 as presented by Balzereit et al. [2]. To make sure that ChatGPT has not seen the image before, we created a P&ID describing the system according to DIN EN ISO 10628 as shown in Figure 2. The system contains four tanks, seven valves, and sensors measuring flow and level while the tanks and valves have unique names. For simplicity, we ignore influences of temperature and assume it is an ideal system with no energy losses. The system was chosen since the connections are challenging through the separation of fluids after Tank 0, the bypass leading through Valve 3, and the three volume flows which lead to Tank 3.

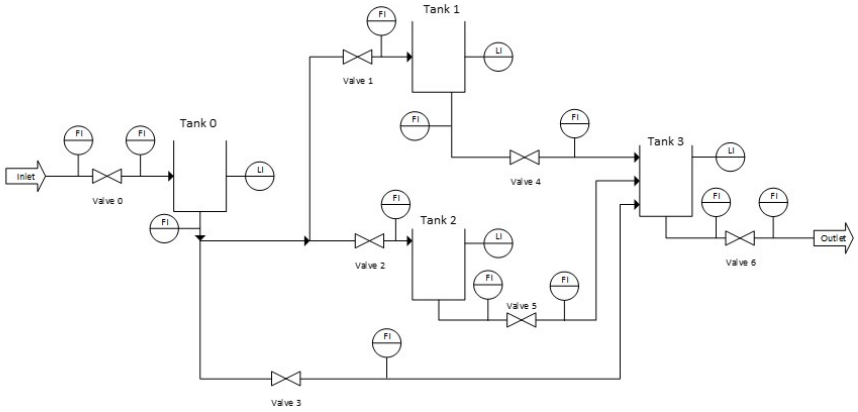


Figure 2: The P&ID of the four-tank system which was used in our experiments

We evaluated both steps of our approach in a separate experiment to better understand where potential problems might occur. This means we took a correct version of the table from step one as input for step 2 instead of using the real output from step 1. To keep the VQA model as deterministic as possible, we used the following parameters provided by OpenAI for prompting via the API interface in Python: seed=42, temperature=0, top_p=0.1, frequency_penalty=0, and presence_penalty=0. We repeated every experiment 100 times to get an idea of possible results and to make sure that valid ones occur more often than wrong ones. To test the reliability of the model, we included the order to write 'unclear' if it is not sure of the result. We used the following prompts in our experiments:

Step 1:
System instructions: "You are an engineer who analyzes piping and instrumentation diagrams. Your task is to identify all elements in the image and describe their input and output connection in tabular form. In the image, there is a piping and instrumentation diagram according to ISO 10628. In the image, there can be symbols for tanks, valves, pumps and other equipment. They are connected by pipes which are shown as lines with arrows indicating the flow direction. Pipes can have edges and do not always lead from left to right but also vertically. Inputs to the system are marked as arrows on the left side of the image and outputs of the system are arrows that lead towards the right side of the image."

User message: "Describe the elements and their connections in tabular form with the following columns: Element, Inputs from, Outputs to. If you are not sure that the answer is correct, write 'unclear'. Give only the table and no further explanations. Provide the output in csv format."

Step 2:
System Instructions: "You are an engineer who has the job to create physical models for process systems. As input, you get a table with elements of a system and which other elements are connected to them via input and output. In addition, you get an image of the system which shows the elements according to DIN EN ISO 10628. The elements are named according to their function. Unique symbols which show if it is an input, an output, or a state variable are usually used in the equations for variables and parameters."

User Message: "Create a dynamical model with physical equations for the volume flow through the valves and the fluid level in the system described in the following table: \"Element,Inputs from,Outputs to\n Valve 0,Inlet,Tank 0\n Tank 0,Valve 0,Valve 1; Valve 2; Valve 3\n Valve 1,Tank 0,Tank 1\n Tank 1,Valve 1,Valve 4\n Valve 4,Tank 1,Tank 3\n Valve 2,Tank 0,Tank 2\n Tank 2,Valve 2,Valve 5\n Valve 5,Tank 2,Tank 3\n Valve 3,Tank 0,Tank 3\n Tank 3,Valve 3; Valve 4; Valve 5,Valve 6\n Valve 6,Tank 3,Outlet\".\n\nUse the following Assumptions:\n\"The fluid in the system is water\nThe fluid is incompressible\nThere are no energy losses\nThe process is adiabatic\nThe tanks are open\nThe valves are at the bottom level of the tank they get feed from\nThe inlet of the tanks is at the top at the same height as the valve the tank gets its input from\nThe inlet to the system comes from an infinite water reservoir with 1m height above the first valve\nAll tanks have the same diameter at all heights\nPipes have the same cross-sectional area everywhere\nthe cross-sectional area of the outlet of tank 0 is three times larger than the cross-sectional area of the pipes\nThe equation for the fluid levels in the tanks i is \frac{dH_{T_i}}{dt} = \frac{1}{A_{T_i}}(\sum\nolimits Q_in-\sum\nolimits Q_{out})\nThe equation for the flow through the valves j is \dot{Q}_{V_j} = C_{V_j} \cdot a_{V_j} \cdot \sqrt{2gH_{T_k}} with T_k being the tank before the valve\"\n\nIf you are not sure that the answer is correct, write 'unclear'. Create the output in Latex.

Give only the equations in executable latex code and no further explanations. Add a brief description of the used symbols in latex."

Element	Inputs from	Outputs to
Valve 0	Inlet	Tank 0
Tank 0	Valve 0	Valve 1, Valve 2, Valve 3
Valve 1	Tank 0	Tank 1
Tank 1	Valve 1	Valve 4
Valve 4	Tank 1	Tank 3
Valve 2	Tank 0	Tank 2
Tank 2	Valve 2	Valve 5
Valve 5	Tank 2	Tank 3
Valve 3	Tank 0	Tank 3
Tank 3	Valve 3, Valve 4, Valve 5	Valve 6
Valve 6	Tank 3	Outlet

Table 1: Sample solution of step 1 and input to step 2.

4.2 Results of the Experiments

The goal for the large-language model in step 1 was to create a table with eleven rows which equals to the number of components in the system. Table 1 shows the sample solution. In all cases, the format was generated correctly. In two repetitions, the inlet and the outlet were recognized as components. Since they were correctly inserted to the table, we did not consider them in the evaluation. The result was considered correct if the complete content of the cell was correct. If something was missing or if a wrong term was included, we counted it as wrong. In total, there were 16 unique results over all 100 runs. The unique results occurred with the following frequencies: 35, 19, 15, 10, 5, 3, 3x2, and 7x1. The results of the experiment are shown in Table 2. We calculated mean, minimum, maximum, median, and the occurrence of 'unclear' in the wrong predictions over all runs. All of the tanks and valves in the image were recognised correctly in each run. The sensors did not occur in any of the results which means the VQA model was able to extract the important parts of the image. More inputs than outputs were identified correctly but the model added 'unclear' more often for the outputs. This means that the trustworthiness of the model to generate outputs is higher than for inputs since the model told us there was an error instead of predicting a wrong value. Unfortunately, the model was never able to recognise the connection between valve 3 and tank 3 correctly, but at least marked the output of valve 3 as 'unclear' in all 100 runs.

The goal of step 2 was to create an equation for every component in the system out of the eleven components in total. The output of the VQA model consisted of four parts in the majority of the runs, namely equations for the valves, equations for the tanks,

	Mean	Min	Max	Median	'unclear' of wrong prediction
Components	1.00	1.00	1.00	1.00	n.a.
Inputs	0.87	0.64	0.91	0.82	0.08
Outputs	0.76	0.36	0.91	0.68	0.83

Table 2: Results of the experiments for step 1. Mean: part of correctly identified components over all runs; Min: Minimum over all runs; Max: Maximum over all runs; Median: Median over all runs; "'unclear' of wrong prediction": Part of the wrong predictions that were marked as unclear (over all runs).

description of the parameters, and, in some cases, additional hints to take into account when using the model. We focused on the validation of the equations, since they would be used further to create models suitable for diagnosis. An equation was considered correct, if all the variables and parameters were included in the correct way. The results showed minor deviations in notation, such as writing $\dot{Q}$ or Q, different words for the reservoir at the inlet of the system (T_{inlet}, T_{res} or $T_{reservoir}$), and using Q_{in} instead of Q_{V_0}. Since these deviations do not affect the performance of the physical model as long as they are consistent over the complete model, they were accepted as correct versions of the model. There were 19 unique results within the 100 runs which occurred at the following frequencies: 52, 13, 8, 2x4, 3, 3x2, and 10x1. The mentioned minor deviations were counted as different unique results. In this step, the word 'unclear' did not occur at all. In total, 99% of the equations were correct while 91 runs generated the physical model completely correct.

4.3 Discussion of the Results

The experiments show the potential of VQA models for the creation of physical first-principles models out of P&IDs. While the second step indicates good performance, the first step holds some potential for improvement. In general, the result is strongly influenced by the data the VQA model has seen during training which makes the results less predictable when applying the proposed method to other systems. Especially since the training data for GPT-4 is not published. The fact that the VQA model was not able to recognize the connection between valve 3 and tank 3 shows that the VQA model might have seen a similar P&ID before, even though we created it from scratch. We did not validate the whole chain by using the output from step 1 as input to step 2. Instead, we used a known, correct version, such that we could look at the behaviour of the two steps on their own and identify the weaknesses of the approach. Thus, the output of step 2 is independent of the actual output of step 1. So far, we cannot judge to which system size our method scales. We expect that it might become necessary to split larger systems into smaller parts and develop strategies to combine the results. In addition, we only tested the approach with ChatGPT so far. Other systems and VQA models should be tested to achieve a more reliable assessment of the applicability of our method. Another point that might reduce the applicability is the fact, that suitable equations need to be given to the model in advance. An automated method to choose and input the equations is desirable. Before our method can be applied in practice, further investigations for the behaviour with other systems and VQA models need to be made.

5 Conclusion

We presented a method to generate physical first-principles models as basis for models that might be applied in fault diagnosis of cyber-physical systems. Our method consists of two steps and relies on VQA models. The first step is to identify components with their inputs and outputs on a given P&ID and to generate the result in tabular form. The second step is to use this table together with further information about the desired model and to generate the physical first-principles model out of it. The experiments, which were done with a P&ID we created for a benchmark system, showed that our method is able to create a model with some limitations. In the first step, 87% of the components and connections were recognized correctly and in the second step, 99% of the equations were created correctly. Future research should focus on improving the detection of components and their connections in P&IDs, testing the approach with more systems and VQA models, and adding steps to generate models that can directly be used for fault diagnosis.

References

[1] L. S. Balhorn, M. Caballero, and A. M. Schweidtmann. Toward autocorrection of chemical process flowsheets using large language models. *arXiv preprint arXiv:2312.02873*, 2023.

[2] K. Balzereit, A. Diedrich, J. Ginster, S. Windmann, and O. Niggemann. An ensemble of benchmarks for the evaluation of ai methods for fault handling in cpps. In *19th IEEE International Conference on Industrial Informatics*, 11 2021.

[3] K. Balzereit and O. Niggemann. Autoconf: A new algorithm for reconfiguration of cyber-physical production systems. *IEEE Transactions on Industrial Informatics*, 2022.

[4] W. Borutzky. A hybrid bond graph model-based-data driven method for failure prognostic. *Procedia Manufacturing*, 42:188–196, 2020.

[5] L. Cao, J. Su, Y. Wang, Y. Cao, L. C. Siang, J. Li, J. N. Saddler, and B. Gopaluni. Causal discovery based on observational data and process knowledge in industrial processes. *Industrial & Engineering Chemistry Research*, 61(38):14272–14283, 2022.

[6] M. A. Chao, C. Kulkarni, K. Goebel, and O. Fink. Fusing physics-based and deep learning models for prognostics. *Reliability Engineering & System Safety*, 217:107961, 2022.

[7] J. De Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial intelligence*, 24(1-3):7–83, 1984.

[8] J. De Kleer and J. Kurien. Fundamentals of model-based diagnosis. *IFAC Proceedings Volumes*, 36(5):25–36, 2003.

[9] Deutsches Institut für Normung e.V. (DIN). Din en iso 10628-2: Flow diagrams for process plants - part 2: Graphical symbols. Din standard, 2012.

[10] A. Diedrich, F. Buchholz, and O. Niggemann. Learning a causal system description for diagnosing physical systems. In *Proceedings of the 33rd International Workshop on Principles of Diagnosis, Toulouse, France.*, 2022.

[11] A. Diedrich, P. Deutschmann, and C. Junker. Servicenavigator - a bayesian assistance system for diagnosing industrial production systems. In *2022 5th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS)[submitted]*. IEEE, 2022.

[12] A. Diedrich and O. Niggemann. Diagnosing systems through approximated information [submitted]. *13th Annual Conference of the Prognostics and Health Management Society*, 2021.

[13] T. Escobet, A. Bregon, B. Pulido, and V. Puig. *Fault Diagnosis of Dynamic Systems*. Springer, 2019.

[14] K. D. Forbus. Qualitative process theory. *Artificial intelligence*, 24(1-3):85–168, 1984.

[15] E. Frisk and M. Krysander. Residual selection for consistency based diagnosis using machine learning models. *IFAC-PapersOnLine*, 51(24):139–146, 2018.

[16] E. Frisk, M. Krysander, and D. Jung. A toolbox for analysis and design of model based diagnosis systems for large scale models. *IFAC-PapersOnLine*, 50(1):3287–3293, 2017.

[17] Z. Gao, C. Cecati, and S. X. Ding. A survey of fault diagnosis and fault-tolerant techniques—part i: Fault diagnosis with model-based and signal-based approaches. *IEEE Transactions on Industrial Electronics*, 62(6):3757–3767, 2015.

[18] E. R. Gelso, S. M. Castillo, and J. Armengol. An algorithm based on structural analysis for model-based fault diagnosis. In *CCIA*, pages 138–147, 2008.

[19] A. Jaber, J. Zhang, and E. Bareinboim. Causal identification under markov equivalence. In *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

[20] S. Kang, G. An, and S. Yoo. A preliminary evaluation of llm-based fault localization. *arXiv preprint arXiv:2308.05487*, 2023.

[21] S. Kato, K. Kanegami, and M. Kano. Processbert: A pre-trained language model for judging equivalence of variable definitions in process models. *IFAC-PapersOnLine*, 55(7):957–962, 2022.

[22] S. Kato and M. Kano. Efficient physical model building algorithm using equations extracted from documents. In *Computer Aided Chemical Engineering*, volume 52, pages 151–156. Elsevier, 2023.

[23] S. Kato, C. Zhang, and M. Kano. Simple algorithm for judging equivalence of differential-algebraic equation systems. *Scientific reports*, 13(1):11534, 2023.

[24] A. S. Khan, A. Q. Khan, N. Iqbal, M. Sarwar, A. Mahmood, and M. A. Shoaib. Distributed fault detection and isolation in second order networked systems in a cyber–physical environment. *ISA transactions*, 103:131–142, 2020.

[25] H. Khorasgani and G. Biswas. Structural fault detection and isolation in hybrid systems. *IEEE Transactions on Automation Science and Engineering*, 2017.

[26] S. Kolb, S. Teso, A. Passerini, and L. De Raedt. Learning smt (lra) constraints using smt solvers. In *IJCAI International Joint Conference on Artificial Intelligence*, volume 2018, pages 2333–2340. ijcai. org, 2018.

[27] M. Krysander and M. Nyberg. Fault diagnosis utilizing structural analysis. *CCSSE, Norrköping, Sweden*, 2002.

[28] I. Matei, M. Zhenirovskyy, J. de Kleer, and A. Feldman. A hybrid qualitative and quantitative diagnosis approach. In *Annual Conference of the PHM Society*, volume 11, 2019.

[29] E. Muškardin, I. Pill, and F. Wotawa. Catio-a framework for model-based diagnosis of cyber-physical systems. In *International Symposium on Methodologies for Intelligent Systems*, pages 267–276. Springer, 2020.

[30] E. K. Nielsen, A. Gofuku, X. Zhang, O. Ravn, and M. Lind. Causality validation of multilevel flow modelling. *Computers & Chemical Engineering*, 140:106944, 2020.

[31] O. Ogundare, G. Q. Araya, I. Akrotirianakis, and A. Shukla. Resiliency analysis of llm generated models for industrial automation. *arXiv preprint arXiv:2308.12129*, 2023.

[32] OpenAI. Chatgpt 4 vision preview. version gpt-4-1106-vision-preview. url: https://platform.openai.com/docs/api-reference, 2023.

[33] J. Runge, S. Bathiany, E. Bollt, G. Camps-Valls, D. Coumou, E. Deyle, C. Glymour, M. Kretschmer, M. D. Mahecha, J. Muñoz-Marí, et al. Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):2553, 2019.

[34] J. Runge, P. Nowack, M. Kretschmer, S. Flaxman, and D. Sejdinovic. Detecting and quantifying causal associations in large nonlinear time series datasets. *Science advances*, 5(11):eaau4996, 2019.

[35] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2021.

[36] M. Vuković and S. Thalmann. Causal discovery in manufacturing: A structured literature review. *Journal of Manufacturing and Materials Processing*, 6(1):10, 2022.

[37] Y. Wu, Z. Li, J. M. Zhang, M. Papadakis, M. Harman, and Y. Liu. Large language models in fault localisation. *arXiv preprint arXiv:2308.15276*, 2023.

[38] Y. A. Yucesan, A. Dourado, and F. A. Viana. A survey of modeling for prognosis and health management of industrial equipment. *Advanced Engineering Informatics*, 50:101404, 2021.

[39] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

LEVERAGING SELF-SUPERVISED LEARNING FOR VIBRATION DATA IN INDUSTRIAL SEPARATORS

Tim Heuwinkel[1], **Silke Merkelbach**[1], **Nils Janssen**[2], **Sebastian von Enzberg**[3], **and Roman Dumitrescu**[1]
[1]Fraunhofer Institute for Mechatronic Systems Design IEM, Paderborn, Germany
{firstname.lastname}@iem.fraunhofer.de
[2]University of Wuppertal, Wuppertal, Germany
njanssen@uni-wuppertal.de
[3]Hochschule Magdeburg-Stendal, Magdeburg, Germany
sebastian.von.enzberg@h2.de

ABSTRACT

Industrial separators play a pivotal role in production processes of various sectors such as chemical, pharmaceutical, biotechnology, oil extraction and food industries, with over 3000 distinct applications. Operating these separators involves managing several process parameters as well as discharge and cleaning cycles, which are hard to control mainly due to deficiencies of current physical sensor technology. Recent studies have shown that machine learning can be utilized to detect faults and particle presence in separators via vibration data. However, traditional machine learning methods require domain expertise or vast amounts of labeled data. We propose the use of self-supervised learning to resolve this issue by learning useful representations from unlabeled data, which is significantly easier and cheaper to obtain. An empirical validation on data from a disk stack separator shows that self-supervised learning can improve upon manual feature engineering and supervised approaches in terms of cost, accuracy and data efficiency.

Keywords machine learning · industrial separators · vibration data · self-supervised learning

1 Introduction

Industrial separators, crucial across diverse sectors such as chemical, pharmaceutical, biotechnology, and food, serve over 3000 distinct applications involving various fluids and solids [11]. Operating these separators involves managing multiple process parameters, including rotational speeds, temperatures, volumetric flow rates, discharge cycles, and cleaning cycles. The accumulation of solids during the separation process

necessitates discharge at specific times for optimal efficiency and product quality [28]. However, the absence of physical sensors capable of directly capturing the quality and quantities of the separation process makes accurate and intelligent control systems challenging [11]. Recent advancements demonstrate progress in using vibration data and machine learning (ML) approaches for intelligent control systems in separators [24, 11]. Traditionally, creating expressive representations for classification or regression tasks related to intelligent control involved manual feature engineering, requiring domain expertise and data science skills [26, 41]. Alternatively, fully data-driven representations in a supervised fashion required large quantities of labeled data, which can be expensive, especially in industrial settings [11].

We propose the use of self-supervised learning (SSL) to mend this trade-off. SSL utilizes raw, unlabeled data recorded from ongoing production with few sensors and a pretext task, which generates feedback signals from the data itself. This approach ensures that meaningful feature spaces are learned in the intermediate representations of the model, allowing for subsequent extraction. Simple models for downstream tasks, in this case related to intelligent control, can then be trained using these intermediate representations as input, combining the advantages of minimal domain expertise and a small number of labeled samples. To test the efficacy of SSL for complex systems like industrial separators systematically, a general and modular structure of SSL approaches is established and implemented for validation. The validation shows what prepeprocessing steps, pretext tasks and architectures are most suitable, how much data efficiency is gained in comparison to supervised baselines, and how well these SSL approaches can generalize concerning process parameters.

2 Prior Work

Traditionally, domain experts manually engineered algorithms for tasks using vibration data, a time-consuming process [26, 41]. Supervised learning enhances this by autonomously solving tasks with labeled data, but there is very limited literature on using supervised learning for industrial separators. The closest work is Merkelbach et al. [24] who employ manual feature engineering on short-time Fourier transform (STFT) representations, achieving 91.27% accuracy using supervised random forest for classifying yeast presence in industrial separators. Zamorano et al. [39] detect wear in disk stack separators using vibration data and an SVM classifier with wavelet transform preprocessing. Vekteris et al. [32] propose a correlation-based approach for predicting bearing failures in dairy industry separators. Although the assignment of manually engineered representations to labels can be done autonomously, experts of the domain and of machine learning are needed to create the representations, leading to high development cost and long innovation cycles [12, 26, 41], as well as labeling cost.

While there is limited related literature on industrial separators, the analysis of ball bearings is described in much more detail. Due to the similarity of the used measurement technology and the basic mechanical principle, insights from this domain can also be included in the investigation of separators. In this domain, Transformer models gain traction due to improved capabilities and larger datasets [9, 1, 30, 3]. Some focus on sequential signals with vanilla Transformers, while others use vision Transformers for time-frequency representations [9]. Spectral transforms, mainly STFT, are common in Vision Transformers for bearing fault analysis [1, 3, 14]. The general architecture involves creating time-frequency representations, splitting them into patches, linearly transforming, and feeding them into a vanilla Transformer encoder [1, 30, 3, 20].

A common weakness in all supervised approaches is their reliance on a large quantity of labeled data, which can be expensive [21]. As stated in the introduction, SSL promises to learn expressive and useful representations without any labeled data, by creating supervision signals from the data itself [21, 25, 18]. Similar to the supervised setting, there is no literature utilizing SSL for complex assemblies like separators and therefore no evidence of their efficacy as well as no information on optimal selection of SSL methodologies. Different pretext tasks have been proposed for SSL in bearing fault analysis though [35, 16, 43, 42]. Most SSL pretext tasks can be classified as "contrastive learning" and "generative learning" [21, 25].

Contrastive SSL aims to create a representation space where similar data points are close, achieved by contrasting positive and negative samples. The SimCLR framework, a prominent example, involves an augmentation module, an encoder, a projection head, and a contrastive loss function [6]. Positive and negative pairs are created through augmentation, with the loss function maximizing agreement between positive pairs while minimizing agreement with other views in the batch [6]. Various works in bearing fault analysis leverage the SimCLR framework, demonstrating its effectiveness in tasks such as cutting tooth fault diagnosis and bearing fault diagnosis [35]. Modifications, like introducing multiple heads and uncertainty-based dynamic weighting, can enhance the framework's performance [16].

Generative SSL aims to learn meaningful representations by encoding input into a latent space and reconstructing it using a decoder [25]. The information bottleneck is crucial for meaningful representation learning, as without it, the task becomes trivial [25]. One classic generative method is the autoencoder (AE), where the encoder maps input to a condensed internal representation, and the decoder reconstructs the input from it [15, 43, 42]. The dimensionality of the internal representation serves as the information bottleneck. In Transformer models, creating a bottleneck in the latent space is challenging due to equal input and output dimensions in encoder and decoder blocks. The bottleneck can also be applied at the input, masking portions of the sequence before passing it to the encoder [5]. This method, tested in bearing fault diagnosis, can improve slightly upon state-of-the-art supervised models [5].

3 Self-Supervised Learning Approach

To test related approaches from the literature in a well-structured manner on their efficacy for industrial separators, general SSL approaches for bearing data are differentiated by the different choices of algorithms at the following stages of the training process: preprocessing, pretext task and architectures. Viable combinations of algorithm choices at different stages of the training process are added to the list of models to test (see Figure 1). The subsection on preprocessing motivates viable choices for transforming the input data, while the second subsection explores considerations related to pretext tasks. Viable deep learning architectures are proposed in the last subsection.

3.1 Preprocessing

Spectral transformations like wavelet- and Fourier transform have shown to benefit self-supervised learning (SSL) and supervised models in bearing fault analysis. While SSL works often use no transform, state-of-the-art supervised models commonly employ continuous wavelet transform (CWT) and short-time Fourier transform (STFT) [16, 10, 1, 14, 3]. CWT reveals non-stationary signals and accentuates singularities, offering advantages for separator-related tasks [30, 9]. The Morlet wavelet is popular for CWT in bearing fault analysis [40, 30] and is therefore chosen as the mother wavelet. Similar

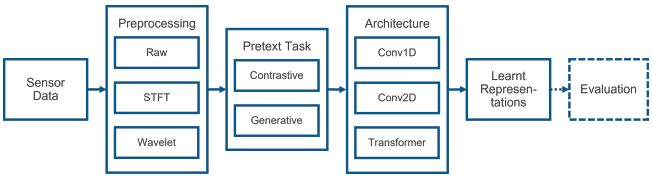


Figure 1: Stages of the learning process with viable algorithm choices in the domain of industrial separators.

to the CWT, STFT can capture time-varying characteristics in signals and highlights frequency components, which makes it successful in supervised bearing fault analysis, albeit less prevalent in SSL literature [1, 3, 14, 20]. To compare the effect of the wavelet transform and STFT to no transform on SSL with data from separators, all three are chosen for a comparison in the preprocessing stage.

Independent of the transform, standardization and data augmentation are employed to improve stability and generalization. Data augmentations help generalization by learning useful invariances [37, 17]. We chose jitter and scaling, since they have proven their effectiveness for vibration data [38, 44, 27].

3.2 Pretext Tasks

One of the most important aspects of an SSL framework is its pretext task. We incorporate generative and contrastive approaches as viable choices for the pretext task, since both types of tasks have been prevalent in SSL, not only in bearing fault diagnosis but also in state-of-the-art natural language processing (NLP) and computer vision (CV) benchmarks [21, 18].

For **contrastive learning**, a SimCLR-style pretext task is adopted, particularly suited for bearing fault analysis SSL [6, 35, 40, 33]. In this task, a recording is divided into short sequences, allowing the learning of representations invariant to variance within a recording. Beyond the typical two augmented views, a third view is created by randomly selecting and augmenting a different sequence from the same recording, enhancing performance over the vanilla SimCLR formulation [16]. The normalized temperature-scaled cross-entropy loss function (NT-Xent) is extended to handle three positive views, by adding all pair-wise losses of the views.

In the realm of **generative learning**, autoencoder models are chosen, given their prevalent use in SSL for bearing fault analysis [43, 42, 5]. For Convolutional Neural Network (CNN) models, the latent representation dimension is restricted to ensure effectiveness. The reconstruction loss utilized is the mean squared error, a standard for autoencoders. Transformer models, with the challenge of equal in- and output dimensions, adopt a modified masked autoencoder inspired by masked language modeling in NLP [8]. This involves masking a portion of the input data, training the encoder on unmasked patches, and padding the encoded patches to restore original dimensions. This generative pretext task offers advantages in reduced processing time and meaningful task creation [13].

3.3 Architectures

Convolutional Neural Networks (CNNs) dominate the self-supervised learning (SSL) landscape for bearing fault analysis, with ResNet being the preferred architecture due to its established performance on diverse datasets [10, 36, 38, 40] and is therefore

investigated for the architecture choice. For non-transformed data, the conventional choice is 1D CNNs, while 2D CNNs are used for three-dimensional time-frequency representations resulting from spectral transforms [16, 33, 35]. The 1D and 2D ResNet architecture can be employed for contrastive learning without modification, but a decoder is necessary for generative pretext tasks to map back to the input space. The decoder replicates the ResNet encoder but uses transposed convolutions for upsampling instead of regular convolutions for downsampling.

Transformer models have also demonstrated efficacy in supervised bearing fault analysis [9, 1, 30, 3] and are therefore chosen as an architecture to be tested. In the proposed approach, a single convolutional layer is added before the Transformer encoder to learn local features and adjust dimensions for three-dimensional spectral-transformed data [8, 13, 4]. To retain information about position, positional encoding based on geometric functions is applied, followed by the use of vanilla Transformer encoder layers. For generative pretext tasks, a vanilla Transformer decoder is added, while for contrastive pretext tasks, a small projector with average pooling and a linear layer is introduced. The average pooling operation is also utilized for fine-tuning on downstream tasks, reducing the output dimension of the decoder [31].

4 Validation

In the validation, all viable combinations of choices for each stage of the learning process are tested on their efficacy for solving downstream tasks related to industrial separators and compared to supervised baselines using vibration data from a separator in a laboratory environment.

4.1 Dataset and Preparation

The dataset for the proposed approach is generated using an experimental separator from the Chair of Fluid Mechanics at the University of Wuppertal. Three-axis accelerometers are attached at five positions to measure vibrations at different parts of the separator, namely at the bowl (2), top, bearing and base (see [24]). With a sampling frequency of 51,200 Hz, sequences of 4 seconds are recorded at regular intervals during the separation process, resulting in 204,800 values per recording, sensor and axis. A total of 1,337 recordings are used. To ensure unbiased results, the data is split into pre-training (935 recordings) and fine-tuning (402 recordings) sets. Each subset is further divided into training (841 and 253 recordings respectively) and validation sets (94 and 28 recordings respectively) for hyperparameter testing and early stopping. Additionally, the fine-tuning data requires a test set for final unbiased metrics calculation (121 recordings). Sequences for training are obtained by applying an overlapping sliding window, creating 200 sequences for Raw/Wavelet transformed data and 6 sequences for STFT transformed data per recording. Additionally, only the magnitude of the spherical coordinates per sensor was used to reduce input dimensionality [24]. No data selection or balancing strategies are employed, reflecting real-world scenarios where certain labels or conditions may be scarce and true distributions unknown.

4.2 Downstream Tasks

Adapting a pre-trained SSL model to downstream tasks can be achieved through fine-tuning on a small quantity of labeled data. In partial fine-tuning, only the classification or regression head and a few of the last layers are finetuned, while in full fine-tuning, every parameter of the model is adjusted [2]. We evaluate the SSL models and the baselines on

three tasks: (1) Determining if yeast particles are present in the separator. In our dataset 10.7% of recordings contain only water and 89.3% of recordings have varying yeast concentrations. (2) Estimating the yeast particle concentration of the input solution. The input yeast particle concentration in our dataset ranges from 0 to 2.5 g/l. (3) Assessing the amount of deposits on the disk stack. For this, the structural similarity index measure (SSIM) of a reference image before separation can be compared to an image during or after separation, to gauge the level of fouling [34]. The SSIM of the images ranges between 0.58 and 1.0 in our dataset.

4.3 Models and Baselines

Since almost all possible combinations are tested, it is easier to justify why certain combinations are not included. Data that was transformed by a spectral transform is two-dimensional in multiple channels, which limits the possibility to be processed by 1D convolutions and therefore only raw signals are processed with 1D CNNs. Similarly, processing one-dimensional data in multiple channels with 2D CNNs has no meaningful interpretation and is therefore excluded.

In addition to the SSL models, several baselines are implemented to represent general approaches in the literature and to isolate the effects of pre-training. The first baseline is the exact implementation of Merkelbach et al. [24], which was validated on the same data set as this paper and represents manual feature engineering with supervised learning. In the following, this approach will be called **STFT+BaselineManual**. The other baselines are based on supervised representation learning. One baseline model is trained for each type of architecture (**BaselineConv1D**, **BaselineConv2D** and **BaselineTransformer**) with the preprocessing strategy that led to the best results in the SSL model comparison.

4.4 Implementation Details

The most important hyperparameters for our approach are the latent dimension for contrastive and generative pretext tasks as well as masking rate, embedding dimension and number of decoder layers for (generative) Transformers. Since the chosen approach is heavily inspired by the SimCLR framework, we use a 128-dimensional latent space for contrastive tasks, like in the original implementation [6]. The latent dimension was set depending on the chosen transform for the generative learning tasks, since vastly more data needs to be compressed if a spectral transform was applied. For spectral transformed data a latent dimension of 1024 was used, while for combinations with no transform, a latent dimension of 256 was chosen. The choice of masking rate is the main adjustment for the difficulty of the generative pretext task for the Transformer models. Similar approaches for time series data report best results with a masking rate of 75%, hence a masking rate of 75% is also employed for our approach [29, 19]. Additionally the embedding dimension for the Transformer architectures was set at 64 (see [14, 19]) and a lightweight decoder with only one Transformer layer was implemented [7].

The training schedule for pre-training included a maximum of 100 epochs, and for fine-tuning 50 epochs, with early stopping based on validation loss. The binary cross-entropy loss was applied for the classification downstream task and a mean-squared error loss for the regression tasks. The base learning rate was set as $1e-4$ and the batch size was set as high as possible for the used GPU, since contrastive learning benefits from larger batch sizes to have more negative pairs [6]. In addition, a cosine decay learning rate scheduler was employed to improve convergence [22]. The AdamW optimizer was utilized to update the weights [23].

4.5 Empirical Results

Table 1 shows results for all selected SSL combinations as well as baselines, downstream tasks and metrics in the partial fine-tuning setting to highlight which SSL combinations work best. To show the effect of pre-training on labeled data efficiency and the learnt representations, the supervised baselines serve as a comparison. Since the best results for the Conv1D and Transformer architecture were achieved with no transform, the corresponding baselines are chosen as Raw+BaselineConv1D and Raw+BaselineTransformer (see Section 4.3). Applying the same criterion to Conv2D, the baseline STFT+BaselineConv2D is chosen. The hyperparameters of each baseline architecture is exactly the same as their self-supervised counterparts. The best overall combination according to the water/yeast classification and concentration regression task is STFT+Generative+Conv2D, with an F1-Score of 0.997 and an RMSE of 0.317. When partially fine-tuning, STFT+Generative+Conv2D outperforms the corresponding baseline, STFT+BaselineConv2D, by 4.7 percentage points in terms of F1 score for classifying water/yeast runs and 12.9% in terms of RMSE for predicting yeast concentration. However, two other combinations with no transform, a contrastive pretext task, Raw+Contrast+Conv1D and Raw+Contrast+TF perform similarly and could only be worse due to fluctuations in the test data. In general, contrastive pretext tasks seem to perform better than generative pretext tasks and convolutional architectures seem to perform better than transformer architectures.

Table 1: Results for partial fine-tuning of all selected SSL combinations as well as baselines and full training data sorted by Water/Yeast Accuracy/F1 score. Best results for each metric and overall model in **bold**. Wavelet transform is abbreviated with WT, Transformer with TF, (vanilla) neural network with NN and contrastive with Contrast.

Combination			Water/Yeast		Concentration		SSIM	
Pre	Pretext	Arch	Acc	F1	RMSE	MAE	RMSE	MAE
STFT	Generative	Conv2D	**0.995**	**0.997**	**0.317**	**0.243**	0.124	0.108
Raw	Contrast	Conv1D	0.985	0.990	0.348	0.278	0.105	0.082
Raw	Contrast	TF	0.980	0.987	0.377	0.308	0.121	0.105
Raw	Baseline	TF	0.972	0.981	0.393	0.332	0.106	0.090
STFT	Contrast	Conv2D	0.967	0.981	0.331	0.259	0.128	0.097
Raw	Baseline	Conv1D	0.954	0.972	0.421	0.348	0.116	0.100
WT	Contrast	Conv2D	0.928	0.957	0.380	0.298	0.106	0.088
STFT	Baseline	Conv2D	0.918	0.950	0.363	0.297	0.103	0.083
WT	Contrast	TF	0.893	0.935	0.393	0.314	0.119	0.104
Raw	Generative	TF	0.860	0.919	0.447	0.383	0.097	0.080
Raw	Generative	Conv1D	0.859	0.921	0.456	0.390	0.120	0.089
WT	Generative	TF	0.851	0.917	0.453	0.392	0.096	0.075
STFT	Contrast	TF	0.851	0.917	0.334	0.264	0.115	0.098
STFT	Generative	TF	0.851	0.916	0.468	0.393	0.105	0.092
Manual	Baseline	NN	0.851	0.915	0.427	0.359	0.116	0.089
WT	Generative	Conv2D	0.851	0.914	0.462	0.391	**0.093**	**0.074**

In essence, labeled data efficiency can be understood as maximizing accuracy for a given number of labeled data points. Partially fine-tuning the SSL models as well as the supervised baselines isolates the effect of pre-training on labeled data efficiency. This is because SSL models with pre-trained feature extractors are compared to randomly initialized feature extractors in the supervised models. While the partial fine-tuning might be optimal for SSL models from an data efficiency standpoint [13, 2], it is more realistic to finetune the full supervised model to harness the full representation learning

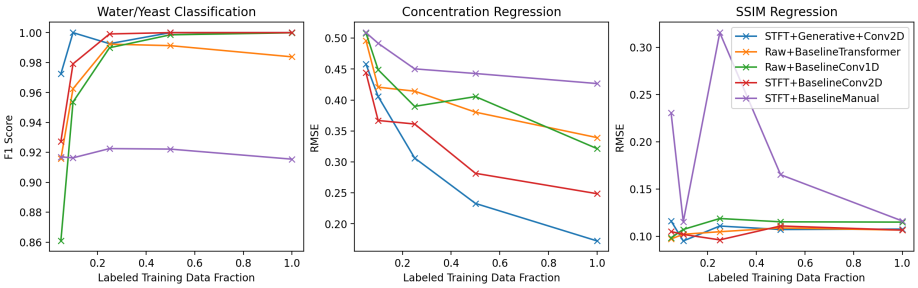


Figure 2: Results for the best SSL combination and baselines in all tasks over increasing fractions of training data with full fine-tuning. A labeled data fraction of 1.0 equates to 402 recordings.

power of deep models. In the full fine-tuning case, pre-training can be seen more as a sophisticated weight initialization scheme, since all pre-trained weights are adjusted.

Figure 2 hence visualizes the labeled data efficiency in the case of full fine-tuning. Similar to the partial fine-tuning case, one might expect that the supervised models would catch up to the pre-trained model, since the pre-trained weights are only used as a starting point for initialization. Because this is not the case, one might conclude that the initial weights set by the pre-training not only create immediately useful representations, but also weights that help convergence for settings with more training data. Apart from a dip in performance at 25% of labeled data for the first task, the SSL model seems to perform very well in comparison to the baselines. With only 5% of labeled data it already reaches an F1-score of 0.97 and an F1-score of 1.0 with 10% of labeled data (40 recordings) on the first task. While the supervised counterpart STFT+BaselineConv2D slightly outperforms the SSL model for very low labeled data settings on the second task, the positive margin between them steadily increases after 25% of labeled data. RMSE and MAE for the SSIM task are quite high for all models (compare range of labels for particle concentration and SSIM), suggesting a poor fit. Moreover, performance on this task does, not seem to be correlated with labeled data fraction at all, for any model, possibly indicating poor label quality.

Table 2: Results for partial fine-tuning for the best selected SSL combination and different out-of-sample (OOS) test sets.

		Water/Yeast		Concentration		SSIM	
Model	OOS	Acc	F1	RMSE	MAE	RMSE	MAE
STFT+Gen+Conv2D	-	0.995	0.997	0.317	0.243	0.124	0.108
STFT+Gen+Conv2D	RPM	1.000	1.000	0.417	0.327	0.179	0.140
STFT+Gen+Conv2D	VF	1.000	1.000	0.442	0.339	0.207	0.161

To actually be useful in practice, the SSL model should generalize reasonably well over possible process parameter ranges, since adjustments to these parameters happens regularly in production. Generalizability can be tested by performing out-of-sample (OOS) validations for different process parameters to simulate inference on unseen production conditions. The two process parameters available in the given data set are the revolutions per minute (RPM) and the volumetric flow rate (VF). The model was trained on recordings which had a nominal range of RPM and VF respectively and then tested only on recordings which were outside the nominal range. Table 2 outlines the changes

in accuracy when testing on out-of-sample data in terms of process parameters. In this setting the first downstream task can still be solved accurately, while the performance for the second downstream task degrades by 0.1 on non-nominal RPM ranges and by 0.125 for non-nominal VF ranges in terms of RMSE. For the third downstream task, testing on non-nominal RPM ranges degrades performance by 0.055 and 0.083 for RPM and VF respectively. While this represents an overall large degradation in performance for the second and third downstream task, a complete shift in process parameters without any similar training data in terms of the tested process parameters should be considered an extreme test of generalization, almost akin to adaptation.

Apart from predictive performance, no domain knowledge was required at any point of the process, greatly reducing development costs and making it possible to adapt the learnt representations autonomously through new unlabeled data. As of current pricing, pre-training the best SSL model (STFT+Generative+Conv2D) can be achieved with under 5 USD using an AWS ml.p3.2xlarge instance and 1000 four second recordings. Furthermore The approach is real-time capable with a processing time of 66ms for one sequence of 600ms sensor data on an NVIDIA RTX 3090 GPU and a 51,200 Hz sensor.

5 Conclusion

We proposed the use of self-supervised learning to improve labeled data efficiency for downstream tasks in industrial separators by learning useful representations from unlabeled vibration data, which is significantly easier and cheaper to obtain. Different combinations of preprocessing, pretext tasks and architectures have been tested and compared to supervised baselines inspired by the related literature. The best SSL model (STFT+Generative+Conv2D) requires little domain expertise and money to develop, while being label efficient, well generalized and flexible as well as real-time capable on the given data set. The empirical validation data is constrained to a single machine in a laboratory, limiting generalizability to diverse production environments. Moreover, real-life production data may be less diverse and contain fewer instances from non-nominal process parameter ranges than the artificially diverse datasets commonly used in experiments. Less diverse training data could inhibit inference for rare occurences in practice.

Since effectiveness of self-supervised learning for numerous domains and now industrial separators has been shown, it should be more closely examined by researchers and practitioners for real-life production processes, potentially paving the way for more efficient and environmentally friendly industrial separation processes.

Acknowledgments

The work in this paper was supported by the German Federal Ministry for Economic Affairs and Climate Action under grant 03EN4004B.

References

[1] C. T. Alexakos, Y. L. Karnavas, M. Drakaki, and I. A. Tziafettas. A combined short time fourier transform and image classification transformer model for rolling element bearings fault diagnosis in electric motors. *Machine Learning and Knowledge Extraction*, 3(1):228–242, 2021.

[2] R. Balestriero, M. Ibrahim, V. Sobal, A. Morcos, S. Shekhar, T. Goldstein, F. Bordes, A. Bardes, G. Mialon, Y. Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[3] Z. Bao, J. Du, W. Zhang, J. Wang, T. Qiu, and Y. Cao. A transformer model-based approach to bearing fault diagnosis. In *Data Science: 7th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2021, Taiyuan, China, September 17–20, 2021, Proceedings, Part I 7*, pages 65–79. Springer, 2021.

[4] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[5] J. Cen, Z. Yang, Y. Wu, X. Hu, L. Jiang, H. Chen, and W. Si. A mask self-supervised learning-based transformer for bearing fault diagnosis with limited labeled samples. *IEEE Sensors Journal*, 2023.

[6] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[7] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Y. Ding, M. Jia, Q. Miao, and Y. Cao. A novel time–frequency transformer based on self–attention mechanism and its application in fault diagnosis of rolling bearings. *Mechanical Systems and Signal Processing*, 168:108616, 2022.

[10] Y. Ding, J. Zhuang, P. Ding, and M. Jia. Self-supervised pretraining via contrast learning for intelligent incipient fault detection of bearings. *Reliability Engineering & System Safety*, 218:108126, 2022.

[11] R. Dumitrescu and M. Fleuter. *Intelligenter Separator: Optimale Veredelung von Lebensmitteln*. Springer-Verlag, 2019.

[12] D. Goyal, A. Saini, S. Dhami, B. Pabla, et al. Intelligent predictive maintenance of dynamic systems using condition monitoring and signal processing techniques—a review. In *2016 international conference on advances in computing, communication, & automation (ICACCA)(Spring)*, pages 1–6. IEEE, 2016.

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[14] Q. He, S. Li, Q. Bai, A. Zhang, J. Yang, and M. Shen. A siamese vision transformer for bearings fault diagnosis. *Micromachines*, 13(10):1656, 2022.

[15] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[16] C. Hu, J. Wu, C. Sun, R. Yan, and X. Chen. Inter-instance and intra-temporal self-supervised learning with few labeled data for fault diagnosis. *IEEE Transactions on Industrial Informatics*, 2022.

[17] B. K. Iwana and S. Uchida. An empirical survey of data augmentation for time series classification with neural networks. *Plos one*, 16(7):e0254841, 2021.

[18] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

[19] Z. Li, Z. Rao, L. Pan, P. Wang, and Z. Xu. Ti-mae: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.

[20] W. Liu, Z. Zhang, J. Zhang, H. Huang, G. Zhang, and M. Peng. A novel fault diagnosis method of rolling bearings combining convolutional neural network and transformer. *Electronics*, 12(8):1838, 2023.

[21] X. Liu, F. Zhang, Z. Hou, L. Mian, Z. Wang, J. Zhang, and J. Tang. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876, 2021.

[22] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[23] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[24] S. Merkelbach, L. Afroze, N. Janssen, S. von Enzberg, A. Kühn, and R. Dumitrescu. Using vibration data to classify conditions in disk stack separators. *Vibroengineering Procedia*, 46:21–26, 2022.

[25] A. Mohamed, H.-y. Lee, L. Borgholt, J. D. Havtorn, J. Edin, C. Igel, K. Kirchhoff, S.-W. Li, K. Livescu, L. Maaløe, et al. Self-supervised speech representation learning: A review. *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[26] M. H. Mohd Ghazali and W. Rahiman. Vibration analysis for machine monitoring and diagnosis: a systematic review. *Shock and Vibration*, 2021:1–25, 2021.

[27] G. Nie, Z. Zhang, M. Shao, Z. Jiao, Y. Li, and L. Li. A novel study on a generalized model based on self-supervised learning and sparse filtering for intelligent bearing fault diagnosis. *Sensors*, 23(4):1858, 2023.

[28] W. H. Stahl. *Fest-Flüssig-Trennung. 2, Industrie-Zentrifugen. Maschinen-& Verfahrenstechnik*. DrM Press, 2004.

[29] P. Tang and X. Zhang. Mtsmae: Masked autoencoders for multivariate time-series forecasting. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 982–989. IEEE, 2022.

[30] X. Tang, Z. Xu, and Z. Wang. A novel fault diagnosis method of rolling bearing based on integrated vision transformer model. *Sensors*, 22(10):3878, 2022.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] V. Vekteris, A. Trumpa, V. Turla, N. Šešok, I. Iljin, V. Mokšin, A. Kilikevičius, A. Jakštas, and J. Kleiza. An investigation into fault diagnosis in a rotor-bearing system with dampers used in centrifugal milk separators. *Transactions of FAMENA*, 41(2):77–86, 2017.

[33] W. Wan, J. Chen, Z. Zhou, and Z. Shi. Self-supervised simple siamese framework for fault diagnosis of rotating machinery with unlabeled samples. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[34] S. Wang, A. Rehman, Z. Wang, S. Ma, and W. Gao. Ssim-motivated rate-distortion optimization for video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(4):516–529, 2011.

[35] M. Wei, Y. Liu, T. Zhang, Z. Wang, and J. Zhu. Fault diagnosis of rotating machinery based on improved self-supervised learning method and very few labeled samples. *Sensors*, 22(1):192, 2021.

[36] Y. Wei, X. Cai, J. Long, Z. Yang, and C. Li. Self-supervised contrastive representation learning for machinery fault diagnosis. In *Neural Computing for Advanced Applications: Second International Conference, NCAA 2021, Guangzhou, China, August 27-30, 2021, Proceedings 2*, pages 347–359. Springer, 2021.

[37] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu. Time series data augmentation for deep learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[38] Z. Yan and H. Liu. Smoco: A powerful and efficient method based on self-supervised learning for fault diagnosis of aero-engine bearing under limited data. *Mathematics*, 10(15):2796, 2022.

[39] M. Zamorano, D. Avila, G. N. Marichal, and C. Castejon. Data preprocessing for vibration analysis: Application in indirect monitoring of 'ship centrifuge lube oil separation systems'. *Journal of Marine Science and Engineering*, 10(9):1199, 2022.

[40] B. Zhang, Y. Mao, X. Chen, Y. Chai, and Z. Yang. Self-supervised learning advance fault diagnosis of rotating machinery. In *International Conference on Neural Computing for Advanced Applications*, pages 319–332. Springer, 2021.

[41] S. Zhang, S. Zhang, B. Wang, and T. G. Habetler. Deep learning algorithms for bearing fault diagnostics—a comprehensive review. *IEEE Access*, 8:29857–29881, 2020.

[42] T. Zhang, J. Chen, S. He, and Z. Zhou. Prior knowledge-augmented self-supervised feature learning for few-shot intelligent fault diagnosis of machines. *IEEE Transactions on Industrial Electronics*, 69(10):10573–10584, 2022.

[43] W. Zhang, D. Chen, and Y. Kong. Self-supervised joint learning fault diagnosis method based on three-channel vibration images. *Sensors*, 21(14):4774, 2021.

[44] X. Zhang, Z. Zhao, T. Tsiligkaridis, and M. Zitnik. Self-supervised contrastive pre-training for time series via time-frequency consistency. *Advances in Neural Information Processing Systems*, 35:3988–4003, 2022.

MACHINE LEARNING PIPELINE FOR APPLICATION IN MANUFACTURING

Antje Fitzner[1], Tom Hülsmann[1], Thomas Ackermann[1], Kevin Pouls[1], Jonathan Krauß[1]
[1]Fraunhofer Research Institution for Battery Cell Production FFB, Bergiusstraße 8,
48165 Münster, Germany
antje.fitzner@ffb.fraunhofer.de

Hendrik Mende[2], Lars Leyendecker[2], Robert H. Schmitt[2,3]
[2]Fraunhofer Institute for Production Technology IPT, Steinbachstraße 17,
52074 Aachen, Germany
[3]Laboratory of Machine Tools and Production Engineering WZL, RWTH Aachen University,
Campus-Boulevard 30, 52074, Aachen, Germany

ABSTRACT

The integration of machine learning (ML) into manufacturing processes is crucial for optimizing efficiency, reducing costs, and enhancing overall productivity. This paper proposes a comprehensive ML pipeline tailored for manufacturing applications, leveraging the widely recognized Cross-Industry Standard Process for Data Mining (CRISP-DM) as its foundational framework.
The proposed pipeline consists of key phases, namely business understanding, use case selection and specification, data integration, data preparation, modelling, deployment, and certification. These are designed to meet the unique requirements and challenges associated with ML implementation in manufacturing settings. Within each phase, sub-topics are defined to provide a granular understanding of the workflow. Responsibilities are clearly outlined to ensure a structured and efficient execution, promoting collaboration among stakeholders. Further, the input and output of each phase are defined.
The methodology outlined in this research not only enhances the applicability of CRISP-DM in the manufacturing domain but also serves as a guide for practitioners seeking to implement ML solutions in a systematic and well-defined manner. The proposed pipeline aims to streamline the integration of ML technologies into manufacturing processes, facilitating informed decision-making and fostering the development of intelligent and adaptive manufacturing systems.

Keywords Machine learning · ML pipeline · Manufacturing · Data mining

1 Introduction

In the domain of modern manufacturing and Industry 4.0, the integration of machine learning (ML) has emerged as a driver of efficiency and innovation in manufacturing and across industries [1].ML has proven effective across a spectrum of applications in manufacturing, including process optimization, monitoring, control, and predictive maintenance [2]. The integration of ML techniques into manufacturing processes comprises an direct impact on physical systems. This transformative shift extends beyond automation, since ML systems have the capability to adapt a dynamic manufacturing environment. For example, in adaptive production workers receive real-time insights and are enabled to optimize the manufacturing process. This not only optimizes processes, but also enhances product quality, and minimizes downtime. However, the integration of ML into manufacturing also comprehends unique challenges such as certification of systems, safety protocols, guarantees of performance and reliability. The integration of ML into existing, but also new, processes has its challenges [1] that can be eased by ML pipelines. They serve as a systematic approach to streamline the end-to-end process of building, training, and deploying ML models. By delineating a clear path for the development of ML models, pipelines not only streamline processes but also enhance scalability and reliability within manufacturing environments.

1.1 Machine Learning Pipeline: CRISP-DM and Other Approaches

The Cross-Industry Standard Process for Data Mining (CRISP-DM; [3]) has long been recognized as a robust framework for guiding the development of data mining and ML projects. The development of CRISP-DM involved collaboration funded by the EU in the 90s, including prominent companies. This ensures that the standard is shaped by diverse perspectives, best practices, and real-world experiences, making it a robust framework for various industries that is still widely used. CRISP-DM consists of the following phases: *Business understanding*, *data understanding*, *data preparation*, *modelling*, *evaluation*, and *deployment*.
While CRISP-DM is a widely used and recognized framework, there are several alternative frameworks and methodologies, each with its own strengths and focus areas. Some notable alternatives are:
ASUM-DM (Analytics Solutions Unified Method for Data Mining/Predictive Analytics, [4]) is a standard process model developed by IBM for the application of data mining and predictive analytics. It serves as a revised and expanded version of CRISP-DM. ASUM-DM comprises five phases (analyze, design, configure & build, deploy, and operate & optimize) along with a project management stream. **CRISP-ML(Q)** (Cross-Industry Standard Process for the development of ML applications with Quality assurance methodology; [5]) is an extension of the CRISP-DM process, that includes two more phases: quality assurance for ML applications and monitoring & maintenance. In focus here are the identification of risks and dealing the the quality of ML applications. **KDD** (Knowledge Discovery in Databases; [6]) is an foundational methodology from the 90s that encompasses the entire process of knowledge discovery from data, including data selection, preprocessing, data trans-formation, data mining and interpretation leading to knowledge. It serves as a broader umbrella under which CRISP-DM and other methodologies operate. **SEMMA** (Sample, Explore, Modify, Model, Assess; [7, 8]) developed by SAS (Statistical Analysis System) that shares similarities with CRISP-DM. It outlines the following sequence of steps: data sampling, data exploration, modification of features, model building, and assessing their value according to the selected metrics. **DMME** (Data Mining Methodology for Engineering Applications; [9]) stands as a comprehensive extension of the CRISP-DM model, offering enhancements on the technical

part of the pipeline. Future work here suggests a more detailed definition of sub-tasks, interconnections and responsibilities.
This is a selective overview, considering the manifold of pipelines and processes designed for specific applications. Notably, some remain unpublished, like the very comprehensive **DMIE** [10], often documented only in theses. [11] and [12] compare multiple ML pipelines, providing insights into their strengths, weaknesses, and overall performance. [11] critiques CRISP-DM for neglecting tasks in project management, organization, and quality in data mining engineering, proposing an own process model. A more recent initiative describes the Data Science Process Model (**DASC-PM**, [13]) with the main objective to organize existing knowledge. The newest version concentrates on the project initiation phase, emphasizing the early establishment of crucial decisions and framework conditions for data science activities.
In conclusion, the state of the art in ML pipelines reflects a holistic approach. As the field evolves, data scientists continue to adapt and enhance each phase of the pipeline to meet the challenges and opportunities presented by the dynamic area of ML and their specific application domain.

1.2 Challenges of Machine Learning Pipelines in Manufacturing

One of the most compelling aspects of CRISP-DM and other similar frameworks is the fact that is was designed to be industry agnostic. Its design allows it to be used in a large variety of different applications [14], which explains its popularity and wide adoption across industries. Conversely, peculiarities of specific industries are not considered and can therefore not be adequately served by existing process frameworks.
The manufacturing domain has significantly profited from advances in data science and ML technology in recent time [15]. The increasing volume of data collected during production processes is used in order to decrease machine downtime and optimize processing times while at the same time improving product quality [16]. However, data driven applications in manufacturing are characterized by complex interactions between the physical and virtual world [17]. This leads to high reliability and safety requirements, as well as a high potential risk. At the same time, data sources are extensive, multimodal and often require deep process understanding, involving numerous stakeholders, to make them accessible for ML applications [18].
KRAUSS ET AL. identified unclear application areas, the availability of comprehensive, open datasets, and the lack of manufacturing specific development frameworks as the key challenges slowing the spread of ML use cases in manufacturing today [18]. They introduce a CRISP-DM based framework, in the form of the *ML Pipeline in Production*, which aims to address some key requirements of manufacturing environments. For example, by introducing two additional pipeline phases, *use case selection* and *certification*, as well as introducing superficial roles, describing the stakeholders for different phases of the pipeline [18].
While guidance on the practical implementation is given, some crucial aspects, like responsibilities within the phases and their output, are not considered by the authors. Drawing from lessons learned in conducted ML projects in the manufacturing domain, the proposed pipeline builds on the existing framework and extends it by addressing the aforementioned restrictions, as well as further refining the definitions of the pipeline phases. The extended pipeline therefore aims to help bridge the gap between theory and practical application even further and therefore facilitate the use of ML in the domain.

2 Machine Learning Pipeline for Manufacturing

This section introduces the proposed ML Pipeline, emphasizing its role as a comprehensive framework for guiding data mining and ML projects within the manufacturing domain.

2.1 General Overview & Contributions

The proposed ML pipeline provides a structured and collaborative approach, integrating the expertise of various roles, to facilitate the successful development, deployment, and ongoing maintenance of ML models. The pipeline as depicted in Figure 1 spans from *use case selection & specification* to *data integration*, *data preparation*, *modeling*, *deployment* and *certification*. It provides clarity on responsibilities to promote cross-functional collaboration, and to deliver robust and effective ML solutions tailored to use cases from the production domain.

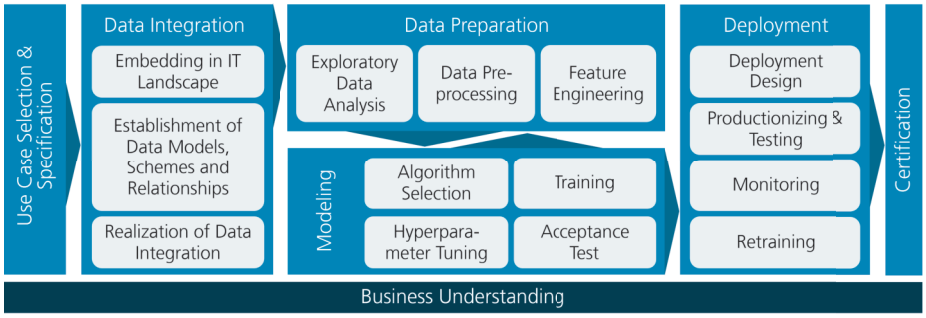


Figure 1: Overview over the proposed ML pipeline.

The pipeline therefore extends existing approaches used in the manufacturing sector in two key areas: Responsibilities and results. This is done by defining a responsible person and persons that are participating or have a consulting function for each of the pipeline sub-steps. This ensures that responsibilities are optimally allocated and provides accountability for all stakeholders and pipeline phases. These roles include persons directly involved in the pipeline, such as data scientists or engineers, as well as supporting roles such as machine operators and process experts.
Additionally, for each phase, the expected results, e.g., a document, concept or certain file, are defined. This improves the accessibility and transparency of the pipeline. As the goal of each sub-step is well defined, the transfer of the pipeline from concept into concrete ML projects is simplified. Furthermore, the expected results can be used as a reporting tool to track the progress of the active pipeline phase.

In both cases, special attention was given to the peculiarities of the manufacturing domain, such as the interplay of experts from different fields with different educational backgrounds and the strong feedback loop between the digital and physical world present in cyber physical systems (e.g. compared to recommender-systems in online shops).

2.2 Responsibilities and Results at each Sub-Step

The roles specified by the pipeline, as well as all sub-steps and their results are described in detail in the following.
The *data scientist* develops and applies machine learning models to extract insights from

data. The *data engineer* designs and maintains the infrastructure for collecting and storing data. The *IT professional* ensures the integration and security of the ML pipeline within the overall IT architecture. The *operator* oversees the operation and maintenance of the machines. The *process and production expert* oversees the production and provides insights into process and product data.

Use Case Selection & Specification: USE CASE SELECTION

Responsible	**Participating**
Depends on the use case. Typically data scientist, data engineer and experts.	All

Results
Prioritized list of use cases with an selection of promising use case to be implemented.

Use Case Selection & Specification: USE CASE SPECIFICATION

Responsible	**Participating**
Depends on the use case. Typically data scientist, data engineer and experts.	All

Results
Definition of the goal of the service to be developed and of requirements and of the (qualitative) metrics. Prerequisites for implementation (data basis) are set. The scope is planned (capacity and hardware resource planning) and definition of termination criteria is finalized.

Data Integration: EMBEDDING IN EXISTING IT LANDSCAPE

Responsible	**Participating**
Data Engineer	Data Scientist, IT professional

Results
Concept how to embed the ML system in the existing IT landscape considering interactions with e.g. MES & IIoT, other existing or planned services, and enable access to the relevant data bases.

Data Integration: ESTABLISHMENT OF DATA MODELS, SCHEMES AND RELATIONSHIPS

Responsible	**Participating**
Data Engineer	Data Scientist, Operator, Production Expert, Process Expert

Results
Definition of existing or new data models.

Data Integration: REALIZATION OF DATA INTEGRATION

Responsible	**Participating**
Data Engineer	None

Results
Data defined in data model are available, accessible and prepared (e.g., basic synchronisation (of time),standardized nomenclature and units according to parameter list).

Data Preparation: EXPLORATORY DATA ANALYSIS

Responsible
Data Scientist, Data Engineer

Participating
Operator, Production Expert and Process Expert

Results
Aggregated report about the data such as statistical analysis and visualizations. Assessment of the data quality and quantity. Definition of termination criteria based on to achieved data (e.g., quality or quantity).

Data Preparation: DATA PRE-PROCESSING

Responsible
Data Scientist and Data Engineer

Participating
Operator, Production Expert, Process Expert

Results
Pre-processed data and ready for model development. The structural pre-processing is done by the data engineer, while the case-related pre-processing based on prior knowledge of the features is done by the data scientist primarily.

Data Preparation: FEATURE ENGINEERING

Responsible
Data Scientist

Participating
Data Engineer, Operator, Production Expert, Process Expert

Results
New features developed for model development for this use case based on the domain knowledge of the use case. In addition, the feature extraction is performed.

Modeling: ALGORITHM SELECTION

Responsible
Data Scientist

Participating
None

Results
Determined set of suitable algorithms for this use case.

Modeling: HYPERPARAMETER TUNING

Responsible
Data Scientist

Participating
None

Results
Strategies for efficient and automated hyperparameter-tuning, suitable for the selected algorithm.

Modeling: TRAINING

Responsible
Data Scientist

Participating
Data Engineer, Operator, Production Expert, Process Expert

Results
Determine the conditions for the training. This covers the decision on the data split or the usage of cross validation and the definition of training parameters such as learning rates. The technical metrics to evaluate the model performance in accordance to the qualitative requirements for the use case are set.

Modeling: ACCEPTANCE TEST

Responsible
Data Scientist

Participating
Data Engineer, Operator, Production Expert, Process Expert

Results
Selection of the final algorithm, hyper-parameter and training parameters and evaluation of the model performance in alignment with use case requirements. All is validated with respect to the existing process.

Deployment: DEPLOYMENT DESIGN

Responsible
Data Engineer

Participating
Data Scientist, IT professional

Results
Selection of the deployment pipeline, e.g. the usage of Docker or other process managers, and the final embedding into the overall IT landscape as planned in the earlier phases.

Deployment: PRODUCTIONIZING & TESTING

Responsible
Data Engineer, Data Scientist

Participating
Operator, Production Expert, Process Expert

Results
Performed integration (including egde cases), penetration and performance test.

Deployment: MONITORING

Responsible
Data Scientist

Participating
Data Engineer, Operator, Production Expert, Process Expert

Results
Definition of the monitoring strategy and implementation (including retraining).

Deployment: RETRAINING

Responsible
Data Scientist

Participating
Data Engineer

Results
Definition of retraining strategy, e.g. implementation of automated retraining, based on the defined requirements.

Certification: CERTIFICATION

Responsible
Production Expert (May be third party)

Participating
All

Results
Certification of the system. System is ready to be used.

3 Discussion and Validation

In ML projects, the participation of different roles often leads to misaligned expectations and communication gaps [19], as well as decision-making complexities. To overcome this, it is essential to encourage collaboration, implement clear communication, and ensure a common project vision. CRISP-DM lacks clear definition of responsibilities among team members and falls short regarding project management compared to e.g., ASUM-DM. CRISP-DM also overlooks the crucial aspect of data acquisition and does not incorporate the concept of a *proof of concept* as an initial project phase.
The proposed ML pipeline ensures that every project phase aligns with the specific

needs and challenges of the manufacturing environment. One of the key advantages is the explicit definition of responsible roles throughout the project, by clearly outlining the roles of five key stakeholders. In this way, the pipeline promotes a collaborative and interdisciplinary approach. Furthermore, the pipeline provides clarity regarding the expected output at each phase of the project. This allows for better planning in advance, improved monitoring capabilities, and helps overcome communication challenges.
The number of iterations needed between the data preparation and modeling phases is not specified, as it can vary significantly based on the specific use case. Use cases involving extensive data exploration, cleaning, and feature engineering may require additional iterations compared to simpler tasks, again emphasizing the importance of adapting the pipeline to the requirements of each specific use case. The same is true for the weight given to different phases. In some situations, entire phases e.g., certification may be skipped, leading to a tailored approach based on given requirements.
Fraunhofer IPT and FFB used the ML pipeline as the structure for about 20 ML-projects, such as various work on glass molding and laser structuring or production in general [20, 21, 22]. The results from these projects validated the functionality of the presented ML pipeline. In MENDE ET AL. [20] feedback from the project members emphasized the effectiveness of the ML Pipeline through its proficient facilitation of model development and deployment within the manufacturing domain. A notable result was the creation of a clear communication that encouraged effective teamwork among various groups, such as machine operators, process planners, and data scientists. The pipeline showed its applicability to user-centred development of ML solutions by incorporating feedback from machine operators, tailored to their specific needs in daily operations.
Explicitly not addressed by the proposed pipeline are concrete timelines for the different phases. The timeline for any given ML project significantly depends on the complexity of the task at hand. However, flexibility is maintained by the pipeline to accommodate variations in project timelines based on varying task complexities.
In summary, the integration of ML into manufacturing processes has a significant impact on physical systems, going beyond automation to enable adaptive production. This results in real-time insights. Hence, the implementation of a tailored ML pipeline for manufacturing provides a structured framework facilitating the integration of ML technologies into manufacturing operations.

4 Conclusion

In ML projects, addressing the challenges arising from diverse roles in the manufacturing domain requires establishing a clear project management approach. The proposed pipeline does so by complementing existing pipelines through the addition of stakeholder responsibilities and anticipating the concrete output of the different phases. Some aspects, like timelines or number of iterations, which are mostly use case dependent, have purposefully been omitted. Validation across different use cases is yet to be completed, with a basic use case already having been addressed. Ongoing efforts are directed towards extending the validation to further scenarios, ensuring the robustness and applicability of the proposed method to a wide array of manufacturing use cases.

Acknowledgments

This work was supported under project “FoFeBat - Forschungsfertigung Batteriezelle Deutschland (03XP0256, 03XP0416, 03XP0501A) and “AI-NET-ANIARA” both funded by the German Federal Ministry of Education and Research (16KIS1275).

References

[1] Wuest et al.: Machine learning in manufacturing: advantages, challenges, and applications, Production & Manufacturing Research, 4:1, 23-45 (2016).

[2] Plathottam et al.: A review of artificial intelligence applicationsin manufacturing operations, J. Adv. Manuf. Process., 5(3), e10159 (2023).

[3] Shearer: The CRISP-DM model: the new blueprint for data mining. In Journal of data warehousing 5 (4), pp. 13–22 (2000).

[4] Angée et al.: Towards an Improved ASUM-DM Process Methodology for Cross-Disciplinary Multi-organization Big Data & Analytics Projects. In: Knowledge Management in Organizations. Cham: Springer International Publishing, pp. 613–624. (2018)

[5] Studer et al.: Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology (2020).

[6] Fayyad et al.: From Data Mining to Knowledge Discovery in Databases. In American Association for Artificial Intelligence (1996).

[7] SAS Institute: SEMMA data mining methodology (2008) http://www.sas.com

[8] Azevedo and Santos: KDD, SEMMA and CRISP-DM: A parallel overview (IADIS), (2008).

[9] Huber et al.: DMME: Data mining methodology for engineering applications – a holistic extension to the CRISP-DM model, Procedia CIRP, Volume 79, Pages 403-408, ISSN 2212-8271 (2019).

[10] Solarte: A proposed data mining methodology and its application to industrial engineering. Master's thesis, University of Tennessee (2002)

[11] Marbán et al.: Toward data mining engineering: A software engineering approach. In Information Systems 34 (1), pp. 87–107 (2009).

[12] Mariscal et al.: A survey of data mining and knowledge discovery process models and methodologies. The Knowledge Engineering Review.; 25(2):137-166 (2010)

[13] Schulz et al.: "DASC-PM v1.1 - A Process Model for Data Science Projects", NORDAKADEMIE gAG Hochschule der Wirtschaft, Hamburg 2022, ISBN: 978-3-9824465-1-6

[14] Mariscal et al.: A survey of data mining and knowledge discovery process models and methodologies, The Knowledge Engineering Review. Cambridge University Press, 25(2), pp. 137–166 (2010)

[15] Jourdan et al.: Machine Learning For Intelligent Maintenance And Quality Control: A Review Of Existing Datasets And Corresponding Use Cases. CPSL 2021. Hannover: publish-Ing., (2021), S. 499-513.

[16] Kim et al. Smart Machining Process Using Machine Learning: A Review and Perspective on Machining Industry. Int. J. of Precis. Eng. and Manuf.-Green Tech. 5, 555–568 (2018).

[17] Monostori et al., Cyber-physical systems in manufacturing, CIRP Annals, Volume 65, Issue 2, 2016, Pages 621-641, ISSN 0007-8506 (2016).

[18] Krauß et al.:Application Areas, Use Cases, and Data Sets for Machine Learning and Artificial Intelligence in Production (2023).

[19] Nahar et al.: Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process (2021).

[20] Mende et al.: Multi-target regression and cross-validation for non-isothermal glass molding experiments with small sample sizes, Proc. SPIE 12778, Optifab 2023, 127780S (2023).

[21] Motz et al.: Benchmarking of hyperparameter optimization techniques for machine learning applications in production, Advances in Industrial and Manufacturing Engineering, ISSN 2666-9129 (2022).

[22] Leyendecker et al.: Predictive Quality Modeling for Ultra-Short-Pulse Laser Structuring utilizing Machine Learning, Procedia CIRP, Volume 117, Pages 275-280, ISSN 2212-8271 (2023).

DATA ACQUISITION CHALLENGES IN AI-DRIVEN SURFACE INSPECTION: A PROVEN SOLUTION PROPOSAL ON COATED SHEET METAL PARTS

Sebastian Hunger
Miele & Cie. KG
Carl-Miele Straße 21, 33332 Gütersloh, DEU
sebastian.hunger@miele.com

Michael Breiter
Eyyes GmbH
Im Wirtschaftspark 4, 3494 Gedersdorf, AUT
michael.breiter@eyyes.com

Claudia Klein
Eyyes GmbH
Im Wirtschaftspark 4, 3494 Gedersdorf, AUT
claudia.klein@eyyes.com

ABSTRACT

This paper introduces an advanced AI-based automated surface inspection system for enhanced quality control in the manufacturing of white-coated sheet metal parts. The study emphasizes overcoming the challenges in AI-driven inspection, such as the need for large datasets corresponding to numerous physical components, which presents storage and logistical issues. By integrating Convolutional Neural Networks (CNNs) and a novel annotation process, the system can be trained effectively on various surface defects. The paper discusses three big data acquisition challenges and provides a solution approach, including large data volumes equivalent to numerous physical components, a novel division of the training process to reduce the workload for domain experts and the relevance of previously clearly defined defect classes.

Keywords AI Surface Inspection · Data acquisition challenges · AI-driven Quality Control · Quality Assurance in Manufacturing · AI in Industrial Process · Convolutional neural networks (CNN) · EfficientNet · VGG16

1 Introduction

The increasing complexity of industrial production processes requires innovative solutions such as automated surface inspection. This technology offers greater precision and repeatability than conventional methods [1]. In today's manufacturing industry, quality assurance is crucial, particularly for customer-relevant surfaces in the production of household appliances. With the aim to eliminate the strenuous, subjective and repetitive process of manual inspection of the components during production [2], this paper introduces an automated surface inspection system, representing a significant advancement in quality inspection. Utilizing artificial intelligence (AI) and advanced camera technology, this system facilitates fully automated assessments of white-coated sheet metal parts formed directly from the coil. The challenge of reliably identifying even the smallest defects with a thickness of a tenth of a millimeter on the approximately 0.5 square meter white-coated sheet metal components is paramount, as these directly impact customer satisfaction and brand perception.

1.1 Related work

Several studies have explored the domain of surface defect detection and classification through various machine learning models. The authors of [3] demonstrate the training of a ResNet model to recognize three different types of defects on steel raw surfaces, achieving a recall rate between 30% and 75% and precision between 72% and 90%, depending on the defect type. In [4] a convolutional auto encoder based anomaly detection method is shown to distinguish between defective and defect-free metal part regions. The authors show the whole process of model selection and parameter optimization but can't find a sufficiently well working result. In that work, no defect type classification is performed. In contrast, [5] outlines a method for training neural networks specialized in anomaly detection. They use a special approach to learn convolution layers for domains with low number of defect samples available. They show promising results on the public DAGM dataset. Furthermore, [6] introduces a new benchmark dataset for the classification of defects on raw steel surfaces. They further train a fast and efficient CNN model targeting inline-inspection, though the practical requirements for the deployment of the application are not discussed. There are many valuable insights into good camera/light settings in [7]. However information to automatic detection of defects and/or algorithmic classification is missing. To our knowledge there is a clear lack of scientific discourse and empirical studies in the field of inline surface inspection of pre-coated raw material with analogue dimensions and transport speeds.

1.2 The Initial situation

The pre-coated raw material is processed directly from the coil, starting with the uncoiling of the coated strip. Before the initial forming step, the material is straightened using leveling rolls and then cut into sheets approximately 600x700 mm using an automatic shear. Subsequently, the material undergoes six forming steps in a tool on a 1000-ton press, creating functional holes, cutouts, beads, and flanges in the component. The finished component is then placed on a conveyor belt and transported with a speed of 11.5 m/min to a manual inspection and handling station. Here, employees inspect the components for permissible defects and prepare them for assembly on the final product.

1.3 AI-driven surface inspection

The new AI-driven surface inspection system is placed on the conveyor belt between the outfeed conveyor of the press and the manual inspection and handling station. This is achieved by having the components pass through an inspection tunnel mounted on the conveyor belt (see fig. 2 no. 1.; movement from bottom right to top left). This makes it possible to inspect the front walls during the run without additional handling and to provide the result of the automated inspection when the respective component arrives at the manual handling station. Inside the test tunnel, 25 light strips and 30 high-resolution monochrome cameras are installed. The cameras are arranged in three rows and allow multiple recording directions at different angles of incidence to the inspected surface. They are mounted in a height-adjustable manner. The particular obstacle posed by the required overlap of the cameras is overcome through precise measurement of the overlap and dedicated calibration methods. The resolutions of up to 16.2 megapixels (5328 x 3040 pixels) ensures that the smallest unacceptable defect (a tenth of a millimeter in the highest quality zone) is displayed with at least 4 pixels. This value has proven to be effective in practical testing and can be used as a basis for the selection of cameras in further inspection systems. A complete raw data recording of one of the 600x700 mm components generates an image dataset of 2.7 GB (lossless compression as PNG). At full production capacity of the press, the live system thus needs to process approximately 1700 megapixels per second. For this reason, the computing and evaluation unit is located directly adjacent to the inspection tunnel (see fig. 2 no. 2.).

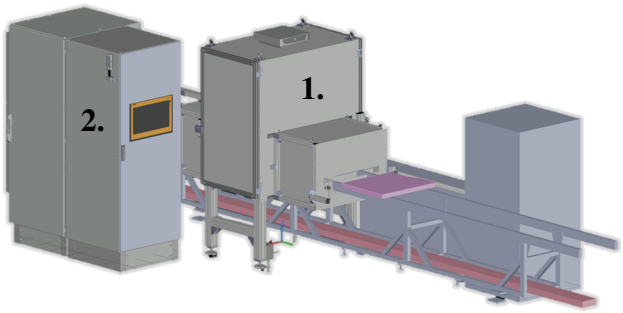


Figure 1: AI-driven surface inspection system

1.4 Why using AI for inspection ?

In the overall assessment of whether a component is acceptable, the type of surface defect (error or defect **class**) and its position (error or defect **localization**) play a crucial role. The localization of a detected anomaly is determined by an underlying CAD model of the component, light gate and a high-resolution encoder (5000 pulses/rotation) on the conveyor belt, which measures rotational motion and translates it to electrical signals. Since the components are always placed in the same orientation at the press outlet, the absolute position of a detected anomaly can be determined by rule-based algorithms. The detected anomaly is then classified by a convolutional neural network (CNN) into one of the following defect classes: scratch, impression, pore/prick, inclusion, bump/dent, contamination, streak, flitter, delamination from the base material and unknown defect. Whether the anomaly then becomes a defect depends on the combination of its position on the component and the defect class. This is why the defect classification capability of the CNN is of the greatest importance for the system. It is therefore not enough to

simply recognise an anomaly. For example, in the highest quality zone (direct visual area of the customer), defects such as scratches, pimples/inclusions, etc., are only permissible up to a size of a tenth of a millimeter. Shape-changing defects that alter the base material and coating, such as bumps and dents, are generally not permissible in this quality zone. In one of the lower quality zones, however, smaller bumps and dents with a diameter lying on the surface of five-tenths of a millimeter are allowed. Here, the robustness, accuracy, and classification capability of the CNN are crucial to avoid misclassifications. Otherwise, an anomaly will be classified as an impermissible defect due to incorrect classification, although the component should have been classified as acceptable (pseudo-error) or, worse, faulty components that should not be installed are rated as acceptable (slip).

1.5 CNN

Convolutional Neural Networks (CNNs) are known since 1989 as highlighted in LeCun´s work on "Backpropagation Applied to Handwritten Zip Code Recognition" [8]. Their ongoing success story was marked by a significant milestone in 2012, when they triumphed in the ImageNet Large Scale Visual Recognition Challenge, surpassing various other methods [9]. Since this breakthrough, CNNs have been instrumental in addressing numerous complex problems through Deep Learning. The most important property of CNNs is their ability to autonomously learn salient features directly from raw data during the training process. This characteristic distinguishes them from other machine learning approaches, which typically rely on pre-defined features. This given, CNNs can easily be scaled to use a higher volume of features and also more complex features. This adaptability and learning capability make CNNs a versatile and powerful tool in the realm of deep learning.

Convolutional Neural networks use convolution layers for deep feature learning. A convolution layer is the mathematical convolution operation, represented as neural network node. In computer vision, 2D spatial convolution over the image pixels with a third dimension for inter-channel influence is used to extract features in images. Several convolution layers as a feed-forward network are forming a deep neural network, where deep features gain a higher receptive field and can represent occurrence of more complex structures or shapes in the original image. CNNs are inspired by the early visual cortex of animals [6].

A single convolution operation consists of a square 2D spatial size N and the number of input channels C, giving N * N * C learnable parameters. In a convolution layer, M such convolutions are used to create M outputs, which are used as inputs for the next layer.

In formula 1 the computation of a single convolution layer output is shown, where $in(x', y', c)$ is the input layer with the 2D spatial index x' and y' and the channel index c, out is the output layer with channel index m and w_m is the m-th convolution filter with its convolution parameters indexed by x,y and c. In addition, $\alpha = N/2$ is an offset to center the convolution window around the input index position.

$$\mathrm{out}(x', y', m) = \sum_{c=1}^{C} \sum_{y=1}^{N} \sum_{x=1}^{N} w_m(x, y, c) * \mathrm{in}(x + x' - \alpha, y + y' - \alpha, c) \qquad (1)$$

In a depthwise convolution, the 3D convolution is reduced to pure 2D convolution ($C = 1$), without inter-channel influence, followed by a 1 x 1 x C convolution, bringing

back some inter-channel influence. This reduces the number of parameters and the number of necessary operations.

1.6 VGG16 vs. EfficentNet

The VGG-16 (Visual Geometry Group-16) network architecture is a highly efficient model for image classification, developed by the Visual Geometry Group at the University of Oxford. It consists of 16 layers, including 13 convolutional layers and three fully connected layers, interspersed with five max-pooling layers to reduce spatial dimensions and improve computational efficiency. The input layer is configured for 224x224 pixel images. Each convolutional layer uses 3x3 pixel filters, increasing in depth from 64 to 512. The model uses Rectified Linear Unit (ReLU) activation functions and ends with a softmax function for class probability outputs [10].

The EfficientNet B0 network architecture uses mobile inverted bottleneck layer with depth-wise separable convolution (mbconv) as building blocks, which are a more efficient approximation for general convolution filters. It consists of 16 such layers, two general convolution layers, a global average pooling layer and a fully connected layer. The channel depth is growing from 32 in the first layer to 320 in the last mbconv layer. The input resolution is 224x224. The architecture is designed to be scalable to reach best quality for any processing complexity target. The findings of Mingxing Tan and Quoc V. Le led to an alternative training approach utilizing the architecture of EfficientNet B0. The authors demonstrated that the EfficientNet architecture achieves comparable quality with reduced computational complexity, compared to other architectures [11]. However, a direct comparison with the VGG architecture was not explored in their study.

2 Challenges in Data acquisition

For the industrial application of AI-driven surface inspection three key challenges in the acquisition of training data can be identified:

1. **Uniform Anomaly Recording:** Anomalies identified by human inspectors must be uniformly and systematically recorded. These anomalies should be classifiable as either usable or defective, as much as possible independently of the inspector. Here, the personal perception of a defect plays a significant role.

2. **Unbalanced Dataset:** The defects detected on sheet metal parts measuring 600x700mm are predominantly less than one square millimeter in size. For this, a concept must be developed that allows for the rapid and targeted preparation of numerous small defects on a relatively larger defect-free surface for training the neural network.

3. **Data Management:** As mentioned in section 1.5 substantial data volumes are required for training purposes. These extensive data sets are equivalent to a considerable number of physical components, which present specific challenges in terms of storage and logistics.

The first challenge can be practically addressed by clearly defining error classes and incorporating additional well-defined meta information such as high- or low-contrast defects, their size or length (especially in the case of scratches) and position. This can be an excellent aid for an independent assessment of defects. It is advisable to clearly define in advance which anomalies (and which meta information) are to be classified as acceptable or unacceptable and not to leave this decision to the personal perception of surface quality. For this purpose, it is recommended that a small core team of quality

experts establish a clear distinction between the defect classes, e.g. a differentiation according to characteristics such as penetration depth, whether the coating penetrates completely or whether the defect is located on the substrate and the coating remains at least locally intact. This can also provide an additional decision-making aid in borderline cases. However, this is also not a trivial matter and is heavily dependent on individual circumstances, so that a detailed description is not provided here.

The second and third challenge was addressed in the project through the introduction of a specially developed "coarse" and "fine" annotation process. In the so-called coarse annotation process, the general position of the detected anomaly is marked using defect stickers, as shown in Figure 2. Each of these annotation stickers has a size of 15x10 mm (code 8x8 mm) and is assigned with a unique numerical identifier. After applying the sticker, the characteristics described under the first challenge, such as error class, size, length, and assessment regarding in-spec or out-of-spec (defective) anomalies, can be noted behind the sticker's identifier. The coarsely annotated component is then fed into the system in a special training mode. The system identifies the stickers and saves the image section with a recognized sticker for further processing. This allows only the raw images around an embroidered defect to be saved in a targeted manner (50-120 MB). Because of the extensive data associated with a part, manually searching for a marked defect area and extracting the images would be a time-consuming and resource-intensive task prone to errors. Although re-annotations are a natural part of the process in any AI project the time dedicated to them has been significantly reduced. Using the sticker number also allows for rapid location of noted defects at any time. From this point onwards, the physical component is no longer required. Subsequently the targeted fine annotation process can begin, in which the exact defect region is marked on the image series taken independently of the physical component and can be used for subsequent training.

A significant advantage arises from this training method, particularly crucial for the training process in industrial practice. Components can be immediately marked during the production process by quickly applying an annotation sticker. The second advantage is that no highly trained quality expert with domain knowledge is required to perform the coarse annotation process. This creates a decoupling in the data acquisition process between pure anomaly detection and the necessary classification of anomalies for training and eliminates the need for long-term storage, entry, and repositioning of physical components, which is particularly beneficial for larger sheet metal parts.

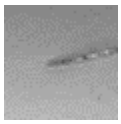
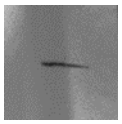
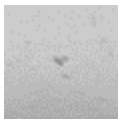
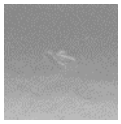
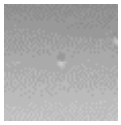
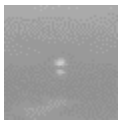
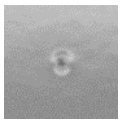
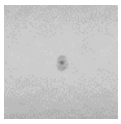
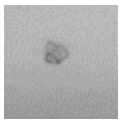
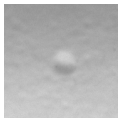
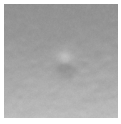
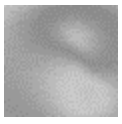
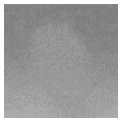
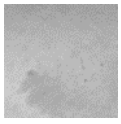
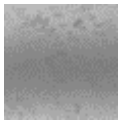
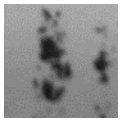
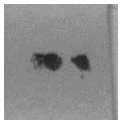
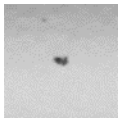
Figure 2: Positioning an annotation sticker near an anomaly

3 Data and Results

The initial data was collected as described above and annotated in 9 defect classes, both coarse and fine. Some examples of the most common and currently best represented error classes are shown in Table 1. In practice, it is recommended to provide a corresponding table with sample images during manual training data preparation, which enables a quick assignment to the in advance defined error classes.

3.1 Data

Table 1: defect class overview

Scratch			
Impression			
Pore/prick			
Inclusion			
Bump/dent			
Streak			
Contamination			

In addition, regions without any defects are collected by recording parts that don't have any defects and considering every region as "no defect" sample further grouped by "background" and "part". Collecting of the defects is done by the annotation process described above. Meta information like type and size is assigned to the defect as a

database entry per annotated defect. Manual corrections and data quality checking is done by machine learning experts with deep learning knowledge. After a few iterations of data collection, prototypical training and hard negative mining, a few defect classes were removed from training, because of low number of occurrences or ambiguity. Table 2 shows the current number of samples per class. It's essential to acknowledge that the research is still in the training phase and the noted imbalance in the number of test samples is expected to correct itself over time. In subsequent analyses the reasons for any deviations that occur will be discussed.

Table 2: samples per class

Class	Defects	Samples	Augmented	Training	Validation	test
No defect:						
Background	-	∼58500	∼58500	∼46800	∼12000	0
Part	-	∼196000	∼196000	∼156000	∼40000	∼7300
Defect:						
Scratch	176	639	5112	4368	744	13
Impression	155	488	4264	3580	684	31
Pore/prick	38	43	404	340	64	35
Inclusion	254	263	2104	1892	312	42
Bump/dent	81	117	936	792	144	33
Contamination	103	535	3026	2305	721	35

A single defect (column 2) on the part results in multiple samples (column 3) because an image is captured every 4.5 mm, so multiple images of the same defect are available. Data augmentation is used to create multiple additional samples (column 4). Only data augmentation that preserves the original image quality such as position jitter, mirroring and multiple of 90°rotations are used, so no interpolation is used which would introduce artificial features (or remove available features). Neither noise addition, nor blurring or resizing is used as augmentation by purpose, because specific cameras in their full pixel-resolution are used by design. Training set is used to learn the CNN weights. Validation set is used to observe the training and to select the best training stage result before overfitting begins.

3.2 Results

There is a strong class-imbalanceness in the data, which is handled by a special batch-balancing, where underrepresented classes are oversampled within a training epoch. Data was split into disjunct training and validation sets. No defect samples that belong to the same defect are put in separate sets. Training of VGG-16 and EfficientNet B0 were performed with an input resolution of 160x160[1] pixel grayscale images. Evaluation shows an overall accuracy of 99% (EfficientNet) and 98% (VGG), which is dominated by the high number of no-defect samples. The accuracy weighted by the number of samples per class is significantly lower with 53% (EfficientNet) and 58% (VGG). In detail (see Tabel 3), both classifiers perform well on the defect classes, scratch, inclusion and contamination and is weaker for impression, pore/prick and bump/dent. Due to the relatively small amount of data, a random effect can be assumed and no further analysis of the differences is carried out at this stage.

[1]Due to the generally very small defect size, the input layer was reduced from 224x244 to 160x160, resulting in improved processing time and lower storage volumes for the defect sections.

Defects	EfficientNet	VGG16
Scratch	0.69	0.85
Impression	0.39	0.32
Pore/prick	0.26	0.17
Inclusion	0.64	0.67
Bump/dent	0.18	0.42
Contamination	0.77	0.66

Table 3: Accuracy comparison of the detection performance of EfficientNet and VGG

From analyzing the confusion matrix and actual image content it looks like bump/dent and pore/brick share some typical patterns with other defect classes, which makes training harder. For these classes, a much higher number of samples will be needed, so that the classifier can learn how to distinguish fine details in the pattern, without overfitting to a few specific defect samples. It is important to mention that the strong imbalance and classifier focus on giving a very strong performance on the no-defect samples has some benefit in practice. In the production system, a very sensitive anomaly detection is used to generate defect candidates. After accepting only candidates that are detected at least 3 out of approx. 8 times at nearly the same part position (+/- 0.5 mm), while theoretically visible in the camera field of view, the classifier decides whether it is still a non-defect region like dust or textile fibers, or whether it is one of the defect classes. On an average real part, over 5000 candidates have to be classified, most of them no-defect regions. For that reason, the performance on no-defect regions has to be outstanding good, otherwise every single part would be overall classified as defective.

4 Conclusion

In conclusion, this paper introduces an novel AI-driven automated surface inspection system for the manufacturing of white-coated sheet metal parts, marking a substantial advancement in quality control. By integrating Convolutional Neural Networks (CNNs) and a new annotation process, the system adeptly addresses several key challenges in AI-based surface inspection, particularly the need for substantial datasets and the accompanying storage and logistical complexities.

The implementation of a structured and uniform anomaly recording process, coupled with a new approach to manage unbalanced datasets, underlines the system's capability to handle a wide range of surface defects with precision. The dual annotation method, integrating both coarse and fine processes, streamlines the training of the neural network, ensuring rapid preparation of large volumes of data with minimal human intervention.

However, it is crucial to acknowledge that this paper presents only the initial stages of the CNN training process. There remains a substantial need for additional training data to further refine the CNN algorithms and enhance their defect detection capabilities (probably around eight thousand more samples). Nevertheless, the system can already be used for initial support during surface inspection. The comparison between the EfficientNet and VGG16 models provides valuable insights into their respective strengths and areas for improvement in defect detection, which will guide future enhancements and adaptations of the AI-driven surface inspection system.

In summary, while this study offers a robust solution to current challenges in manufacturing quality control and sets a precedent for future research, it also highlights the ongoing journey in the realm of AI-driven surface inspection. The continual evolution of

training data and model refinement is essential for the advancement of quality assurance in manufacturing processes, promising a future of increasingly efficient and accurate automated inspection systems.

References

[1] Neogi, N., Mohanta, D.K. & Dutta, P.K. "Review of vision-based steel surface inspection systems". J Image Video Proc 2014, 50 (2014). https://doi.org/10.1186/1687-5281-2014-50

[2] Xie, Xianghua. “A Review of Recent Advances in Surface Defect Detection using Texture analysis Techniques.” Electronic Letters on Computer Vision and Image Analysis 7 (2008): 1-22. url: https://api.semanticscholar.org/CorpusID:8516210

[3] Konovalenko, Ihor, et al. "Steel surface defect classification using deep residual neural network." Metals 10.6 (2

[4] Kähler, Falko, Ole Schmedemann, and Thorsten Schüppstuhl. "Anomaly detection for industrial surface inspection: Application in maintenance of aircraft components." Procedia CIRP 107 (2022): 246-251.

[5] Staar, Benjamin, Michael Lütjen, and Michael Freitag. "Anomaly detection with convolutional neural networks for industrial surface inspection." Procedia CIRP 79 (2019): 484-489.

[6] Fu, Guizhong, et al. "A deep-learning-based approach for fast and robust steel surface defects classification." Optics and Lasers in Engineering 121 (2019): 397-405.

[7] Mike Muehlemann; "Standardizing Defect Detection for the Surface Inspection of Large Web Steel". Illumination Technologies, Inc. 2000

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel; "Backpropagation Applied to Handwritten Zip Code Recognition". Neural Comput 1989; 1 (4): 541–551. doi: https://doi.org/10.1162/neco.1989.1.4.541

[9] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E.; "ImageNet classification with deep convolutional neural networks". Commun. ACM 2017; 6 (60): 84-90. doi: https://doi.org/10.1145/3065386

[10] Karen Simonyan, Andrew Zisserman; "Very Deep Convolutional Networks for Large-Scale Image Recognition". Visual Geometry Group, Department of Engineering Science, University of Oxford; ICLR 2015 doi: https://doi.org/10.48550/arXiv.1409.1556

[11] Mingxing Tan, Quoc V. Le; "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". ICML 2019 doi: https://doi.org/10.48550/arXiv.1905.11946020): 846.

HYBRID ONLINE TIMED AUTOMATON LEARNING ALGORITHM FOR DISCRETE MANUFACTURING SYSTEMS

Simon Martens
Bielefeld University of Applied Sciences
simon.martens@hsbi.de

Alexander Maier
Bielefeld University of Applied Sciences
alexander.maier@hsbi.de

Alexander Wunn
Haver & Boecker OHG
a.wunn@haverboecker.com

ABSTRACT

This paper presents the Hybrid Online Timed Automaton Learning Algorithm (HyOTALA), a novel approach for anomaly detection in cyber-physical production systems (CPPS). It addresses the challenge of using hybrid data, combining sparse discrete and continuous signals, by learning a hybrid timed automaton model in an online setting. This model captures the dynamics of discrete manufacturing processes and provides significant advances in model identification for CPPS. The effectiveness of HyOTALA is demonstrated through a practical application, highlighting its potential to improve anomaly detection capabilities in industrial settings.

Keywords Cyber Physical Production Systems · Model Identification · Hybrid Timed Automata · Online · Unsupervised · Anomaly Detection

1 Introduction

Anomaly detection and predictive maintenance in cyber-physical production systems (CPPS) are important tasks in many industrial applications, as they can help identify and resolve problems in production processes. However, there are several challenges that need to be overcome to ensure a successful application [10].

When building models that can be used for anomaly detection, one of the biggest problems is that discrete PLC (programmable logic controller) data often goes unused.

Yet this data can provide valuable information about the health of a machine. Instead, the focus is on data from sensors, sometimes installed specifically for this purpose [25]. This may be due to a lack of awareness of methods tailored to anomaly detection in discrete production systems. Instead, classical time series algorithms are usually used, which do not always give the best results.

One reason why the use of discrete PLC data is crucial, is mode switching. Mode switching is responsible for continuous signal non-linearity and is highly correlated with discrete PLC data. An example is a gear change that results in an unstable motor speed. However, learning this dynamic is complicated without proper embedding of discrete signals. Even if discrete signals are embedded in a black box model, there is no guarantee that the characteristic will be learned correctly [11].

Another reason underpinning the importance of proper mode identification, is the temporal behaviour of a CPPS. In particular, explicit modelling of the transition time between modes is a powerful health indicator. A worn cutting tool, such as a blade, will take longer to cut through material. Or a worn conveyor belt will take longer to transport. Identifying these types of anomalies is low hanging fruit if discrete signals are properly processed.

A major challenge that we discuss in this paper is how to deal with hybrid data while ensuring that the above and other features are learned. A hybrid learning algorithm can handle sparse discrete data (e.g. digital valve states) and continuous data (e.g. analog engine speed).
To address this challenge, we've developed a new algorithm called the Hybrid Online Timed Automaton Learning Algorithm (HyOTALA). It learns a hybrid timed automaton model in an online setup.

In the paper, we'll first look at the current state of the art and explain the research gap in section 2. Then, in section 3, we'll take a deep dive into how HyOTALA works. After that, we'll evaluate the practical application of HyOTALA in section 4. Finally, we'll summarise our findings in section 5.

2 State of the Art

This section presents methods for identifying models that can be used for anomaly detection. While popular machine learning models may not always be the ideal choice, lesser-known explicit models offer a good alternative. This idea is further explored in the subsection 2.1. Further on, we will focus on (hybrid) timed automata as a formalism and explore methods for learning them. These methods will then be categorised according to the type of signals they can handle. Methods for identifying discrete models can be found in subsection 2.2, while methods for continuous and hybrid models, which are a combination of both types, are discussed in 2.3. Finally, the research gap of this paper is explained in 2.4.

2.1 Underutilization of Explicit Models

Modern machine learning methods show promising results on (research) data sets. Sequence-based models such as RNNs, LSTMs, or transformers are theoretically able to learn contextual, sequential, and temporal features of CPPS [17], [24].
However, there are practical limitations, such as the availability of large datasets for training complex models. It is also difficult to interpret and discuss black box models. In general, phenomenological models do not guarantee to learn system modes, sequential

and temporal behavior correctly. As a consequence, occurring point, context, and sequence anomalies can't be detected with certainty [31], [2].

We could do better by limiting the degrees of freedom of the models and providing a fixed and simple structure based on domain knowledge (explicit modeling). There are several types of explicit models that have been learned and then used for anomaly detection. In [12] Bayesian networks learn to discover dependencies between sensors and actuators. Another formalism is Hidden Markov Models (HMM), which are used to detect anomalies in a SCADA system, as [27] shows.
Of all the available formalisms, it is easy to argue that a discrete automaton or a Petri net is definitely a natural representation of a CPPS, since PLC programmers regularly use it as a programming paradigm [19], [5]. A timed automaton takes a discrete automaton one step further. Basically, it is a state machine that also keeps track of the timing of transitions between states. This is a huge advantage, because it models the interaction of the PLC program with the environment. In general it is a good model of a CPPS [15]. Anomaly detection using a timed automaton can be done by traversing through the automaton and comparing observations with the learned model. A significant deviation indicates an anomaly. Since a timed automaton is close to the practical experience of domain experts, it enables human-in-the-loop model improvement and validation [4]. Timed automata are used in practice to model the behavior in CPPS in [6] or [28].

A hybrid timed automaton is a timed automaton with state-specific continuous signals that don't correspond to a specific state. This allows for joint and context-aware analysis. In practice, the information provided by the continuous signals is stored in a statistical or machine learning context-aware model. This approach goes in a similar direction as the emerging field of informed machine learning, which deals with the integration of expert knowledge into machine learning models [21], [1].
But the formalism has its limitations. It doesn't take into account all the nuances and complexities of every possible PLC program, especially those with advanced process control and dynamic patterns. Before applying the methods presented in this paper, a domain expert should confirm the applicability. In general, this can be done quite intuitively with hybrid automata, since the identified states of the automaton correspond to the states of the system and can therefore be validated by an expert.

2.2 Discrete Methods

There exists a number of methods for identifying discrete timed models, which can be roughly divided into offline and online methods. BUTLA [16] and RTI+ [29] are two such methods that learn a timed automaton and differ in the methodology of state merging. Both methods have in common that they work offline, i.e. measurements have to be stored in advance and are permanently available. OTALA is a method that learns a timed automaton in an online manner [14]. Other possibilities are the learning of Petri nets, dynamic Bayesian networks or continuous-time Markov chains.

2.3 Continuous and Offline Hybrid Methods

One group of algorithms enables joint timing analysis of continuous and hybrid system behavior by creating a discrete representation of the overall system behavior. They convert continuous signals into discrete signals. Of course, this also works in the absence of discrete signals. Finally, an offline discrete algorithm like BUTLA can be used as if there were only discrete signals, or HyBUTLA for offline identification of hybrid timed automata using the initial discrete and continuous signals [21]. Methods for

intelligent discretization are e.g. Boltzmann machines ([8]), deep belief nets ([9]), or self-organizing maps ([30]).

2.4 Research Gap: Online Hybrid Methods

To the best of our knowledge, there is no online algorithm for learning hybrid timed automata. Existing algorithms for learning hybrid timed automata are offline, meaning that they require all input-output traces to be available in advance for learning. Online learning is a requirement for many industrial use cases. Often industrial systems have limited resources and need to process data in a stream. There is an entire research area dedicated to machine learning on edge devices [22].

There is the online algorithm OTALA for discrete data, but there seems to be no method that includes continuous data. It might be possible to train an intelligent discretization model online in advance and combine it with OTALA. This approach seems inconvenient because it's not certain that OTALA's assumption that each discretized state corresponds to a unique machine state holds. In this paper we want to introduce HyOTALA, an online algorithm for learning hybrid timed automaton models.

3 Algorithm

The previous section discussed the need for an online hybrid timed automaton learning method. In this section, we present our solution, called the Hybrid Online Timed Automaton Learning Algorithm (HyOTALA). HyOTALA builds on OTALA, originally introduced in [14], and is capable of learning timed automata without continuous signals.

The former algorithm OTALA is briefly summarized in the first subsection 3.1 and will be used to explain HyOTALA. The main difference between OTALA and HyOTALA is the concept of a hybrid state, which is explained in subsection 3.2. Finally, subsection 3.3 will explain how to integrate hybrid states into OTALA, resulting in HyOTALA.

3.1 OTALA

OTALA is an online algorithm that can identify a timed automaton model of a CPPS using discrete PLC IO signals. It avoids the need to collect labeled anomaly samples because only a model of normal operation needs to be learned, it is only required that the recorded data originate from the normal process. In addition, OTALA has the advantage of being able to autonomously detect when the learning process has reached convergence.

As shown in algorithm 1, OTALA can be implemented using three data structures: an automaton, a state, and a transition.
The concept of an automaton state is encapsulated by the state class. It is characterized by an IO vector and a visit counter. The IO vector consists of discrete inputs and outputs and is a primary key for a state, because OTALA assumes that an IO vector corresponds to exactly one machine state. In practice, a domain expert must verify this assumption by selecting unambiguous signals that reflect the machine state. In addition, there should be one model for each machine module that operates concurrently.
Transitions between states are modeled using transition objects. Each transition is defined by its previous and next state, a visit counter, and a timing distribution. The original implementation of OTALA only keeps track of the minimum and maximum timing. However, experience has shown that a more robust representation is needed. Therefore, we propose to learn a timing probability distribution that better reflects the temporal behavior. If a normal distribution can be assumed, online parameter estimation

Algorithm 1 OTALA: Data structures

```
1: class STATE
2:     discreteIO : IO VECTOR
3:     visitCounter : INT
4: end class

5: class TRANSITION
6:     prevState : STATE
7:     nextState : STATE
8:     timingDistribution : ONLINE-PROBABILITY-DISTRIBUTION
9:     visitCounter : INT
10: end class

11: class AUTOMATON
12:     States : DICT                          ▷ Key: Discrete IO
13:     Transitions : DICT                     ▷ Key: Previous and Next State
14:     NoChangeCounter : INT
15: end class
```

using Welford's method is a good option [3].
The heart of OTALA is the automaton class, which represents the entire automaton. It consists of a dictionary of states, indexed by their discrete IO vectors, and a dictionary of transitions, indexed by a combination of previous and next state IOs. This structure allows for efficient retrieval and updating of states and transitions during the learning process. In addition, the automaton also maintains a no-change counter, which is incremented if an observation doesn't change the automaton, or reset otherwise.

Algorithm 2 OTALA: Training loop

```
16: A ← initAutomaton()
17: while newObservationExists() and not isConverged(A) do
18:     t_t, d_t ← getNewObservation()                ▷ time, discrete IO
19:     s_t : State ← getOrCreateState(A, d_t)
20:     visitState(s_t)                                ▷ counter
21:     if s_t ≠ s_{t−1} then
22:         tr : Transition ← getOrCreateTransition(A, s_t, s_{t−1})
23:         visitTransition(tr, t_t, t_{t−1})          ▷ counter, timing distribution
24:     end if
25:     trackConvergence(A)
26: end while
```

OTALA's primary objective is to construct an automaton that accurately represents the dynamics of a system based on discrete IO observations. The algorithm iteratively refines the automaton until it converges, indicating that a stable representation of the system's dynamics has been learned. This learning process is described in algorithm 2.

The learning process ends when either there are no more samples or *isConverged* in line 17 evaluates to true. The stability of the representation correlates with the number of recent adjustments. Thus, *isConverged* essentially checks whether the no change counter of the automaton exceeds a threshold.
In line 18 *getNewObservation* parses the new observation into a timestamp and a discrete IO vector. In the following line, *getOrCreateState* either creates a new state object or

retrieves the state object corresponding to the IO.
The if condition in line 21 checks if the state has changed. If there is a state change, a transition object is created or retrieved. Besides incrementing the visit counter, *visitTransition* (line 23) is responsible for fitting the time distribution.
Finally, line 25 tracks the convergence by maintaining the no change counter. The counter is reset when a new state or transition is created, or when the transition timing distribution changes significantly.

3.2 Hybrid State

Because discrete PLC IOs carry information such as conveyor on/off, an automaton state is highly correlated with the CPPS mode or context. Since continuous signals are usually highly mode dependent, such as the vibration of a conveyor, it makes sense to embed a state-aware continuous model into an OTALA state. A model embedded into a state results in a hybrid state (HyState) as shown in algorithm 3.

Algorithm 3 Hybrid State

```
1: class HyState(State)
2:     contiuousModel : Online-ML-Model
3:     metric : Online-ML-Metric
4: end class
```

Since a *HyState* is to be integrated into OTALA, the online properties must be preserved and the model must be learnable online. Here the term model also includes an online preprocessing pipeline. A *HyState* also needs an online metric such as an R2 score for two main reasons. First, it helps to measure the quality of the learned model. Second, OTALA measures the convergence of the learning process, and to extend it to hybrid states, the online metric can be used to estimate the convergence of each continuous model.

Choosing a model and a metric requires domain expertise. For inspiration and implementations of online machine learning methods, we recommend taking a look at River, a library for machine learning with streaming data [20]. Using a library like River allows you to easily build complex online pipelines, including steps like feature selection, scaling, and model training. As explained in 2, always keep in mind the computational and memory constraints and the higher computational demands of complex pipelines. Online ML models such as autoencoder neural networks for embedded devices are actively being researched [23], [7].
However, for most tasks, you don't need really complex models. One thing we have learned is that simple models work (surprisingly) well in HyOTALA. This is because the context is given and mode switches that introduce nonlinearity, like a gear shift, are handled by the timed automaton model.

A good starting point is to estimate joint or single distributions of each continuous signal and use them as the continuous model. This has the advantage that these models are easy to interpret and human-in-the-loop feedback can be used for model optimization and quality control [4].
Another option is linear models, which for example take an aggregation of recent observations as input and predict the next observation, also used in edge computing [26]. In proof-of-concept experiments, we have used Bayesian linear regression, which has the advantage of handling uncertainty [18]. It has also proven effective in other online state estimation tasks of CPS [13]. The model naturally simplifies the pipeline by eliminating

the need for preprocessing. It is also capable of modeling uncertainty in a system. In our internal proof-of-concept case study, it was able to model the system reasonably well.

3.3 HyOTALA: A Learning Algorithm for Hybrid Systems

As mentioned before, the main difference between OTALA and HyOTALA is the replacement of states by hybrid states. Almost everything else is the same. The training loop must also be adapted. Most of the steps to train the system, as shown in 2, remain the same. However, when there is new data (*getNewObservation*, 18), there is also a continuous signal vector, noted as c_t. The other change concerns the function *visitState* (20). The new implementation is shown in the algorithm 4.

Algorithm 4 HyOTALA: visitState

```
1: function VISITSTATE(s_t, c_t)          ▷ Input: State + Continuous Data
2:     incrementCounter(s_t)
3:     f_t = preprocess(c_t, other)       ▷ Create a feature vector
4:     learnOne(s_t, f_t)                 ▷ Adapt model of state s
5:     evaluate(s_t, f_t)                 ▷ Adapt metric of state s
6: end function
```

When a state is visited, the former OTALA implementation just incremented the visit counter. HyOTALA also updates the continuous model of the current state with the new continuous data (c_t). This involves a preprocessing pipeline that creates a feature vector f_t that is used for model training. Note that the creation of the feature vector can also take into account an old feature vector f_{t-1} or a signal vector c_{t-1} stored in an internal state. Finally, the online metric is updated by evaluating the current model.

A key aspect that has not been thoroughly explored is the impact of the ratio of continuous to discrete signals on the algorithm's quality. We suggest to maintain a balanced ratio of continuous to discrete signals. A rule of thumb indicates that the ratio of continuous to discrete signals (conti/discrete) should not exceed 0.5.

4 Evaluation

The proposed learning algorithm is applied to real data extracted from a Haver & Boecker OHG packaging machine. Haver & Boecker OHG is a machine manufacturer that specializes in building packaging machines that pack bulk materials into bags. A packaging machine can consist of several filling modules. As an example, a filling module is shown in Figure 1. The main components of a filling module are the weighing unit and the impeller that feeds the product into a bag. The impeller is not visible in this illustration because it is located behind the filling tube. Each filling module follows a cyclical pattern of placing the bag on the filling tube, taring the weight, filling the bag, sealing the bag, and releasing the bag. For simplicity, only one filling module is considered in this section. This is also done to avoid possible parallel processes. The data set includes 71 binary and 2 continuous characteristics from each PLC cycle. Error messages from the machine are also included in the data set.

The training data was reviewed by domain experts for anomalies and errors. A time period was chosen where the machine was continuously packing and no errors occurred. Each continuous feature is modeled by a separate model. Initial modeling efforts revealed a major problem in learning the continuous models. This problem concerns the states during the filling process. In some cases, a binary feature may experience a rising edge

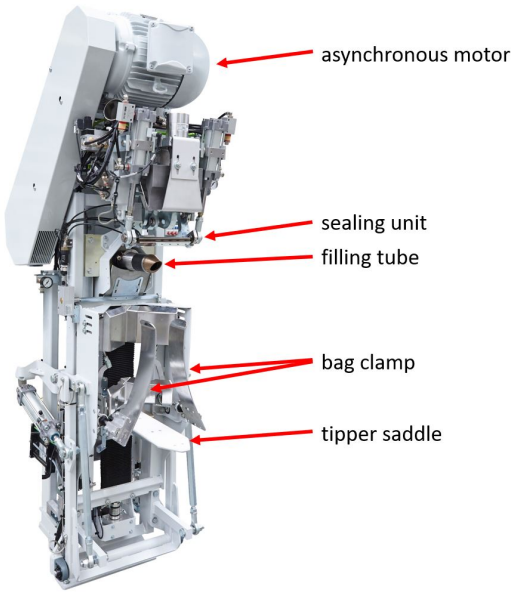


Figure 1: A Haver & Boecker OHG Filling Module

for a few PLC cycles, causing a brief exit from the current state during filling. This can make it difficult to model the continuous features as a function of time because the values of the continuous feature increase during the brief exit. The difficulty is that when the state is re-entered, the value of the continuous feature is higher than when the state was first entered. Figure 2 illustrates the problem. Fill 1 is a fill where this phenomenon did not occur. However, the phenomenon is observed in Fill 2. It is clear that the value of the signal is significantly larger when the state is re-entered. To solve this problem, domain experts have recommended removing certain binary features. Therefore, the state is no longer left for a short time.

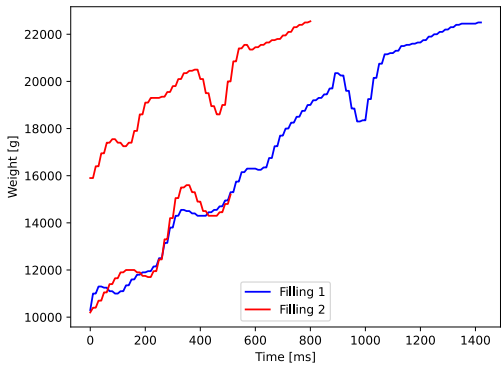


Figure 2: Continuous Signals in one Hybrid State

Using HyOTALA results in an automaton with about 45 states and 60 transitions, which is shown in Figure 3. The cyclic behavior mentioned above is easily recognizable. The results were shown to several domain experts, mostly PLC programmers. They were able to identify different stages of the packaging machine based on the provided automaton. Even on closer inspection by the domain experts, this automaton is considered an appropriate abstraction of the machine's behavior.

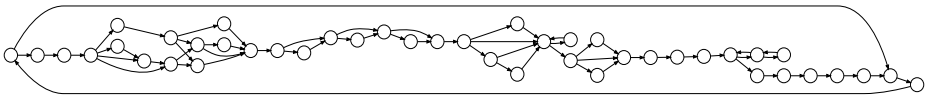


Figure 3: Anonymized Hybrid Timed Automaton for a Packaging Process

Figure 4 compares the total number of changes to the number of PLC cycles. In this figure, only the changes resulting from the addition of a new state or transition are counted. It is obvious that most states and transitions are learned after only one fill. This paper only discusses the convergence of the binary part. In addition, future considerations should include the convergence of the continuous models and the time distributions of the transitions.

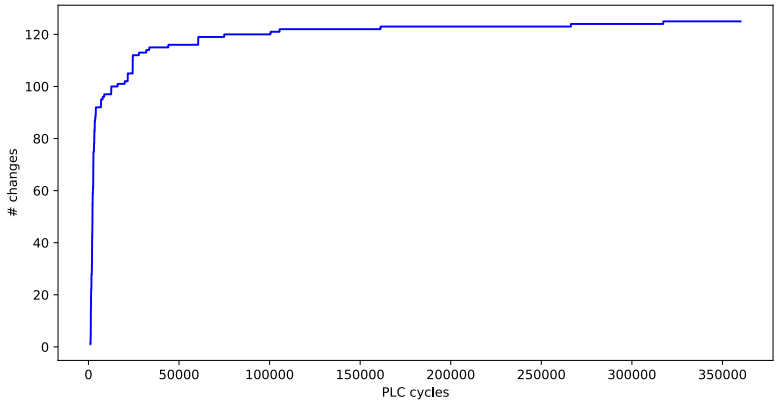


Figure 4: Convergence

Two anomalies are used to further determine the learning performance of the resulting automaton. The first case is a typical anomaly that occurs in packaging machines: Overweight or underweight of the filled bag. Underweight or overweight bags can occur for several reasons. For example, the product flow is not optimal and the product flows at different speeds during filling. Another reason could be an improperly parameterized packaging machine. The filling of an overweight bag is considered as an example. The set of binary states observed during filling is compared with the set of binary states of the machine. This comparison shows that 4 states were observed during the filling process that are not represented in the learned automaton. This kind of observation is called a binary anomaly. The same comparison was made with the filling of an underweight bag. This comparison also shows unknown states that are not contained in the learned automaton.

A second type of anomaly is considered that focuses on the learning performance of continuous models. Unlike the previous example, which considers binary models. When the bag is discharged after filling and sealing, it falls onto the conveyor belt below. Under unfortunate circumstances, the bag may tilt as it falls onto the conveyor belt. As a result, the following bags cannot be transported any further and a deadlock occurs. In the current configuration of the packaging machine, there are no sensors installed in this area to detect this error. In this case, the observed continuous signals of a particular state are compared with the corresponding learned continuous models. Immediately after the bag is tilted in the machine, unusual deviations are observed in some states before the following filling. This result is plausible and can be explained by domain experts. The reason for the unusually high weight measurement before filling is due to the tilted bag pressing on the machine's weighing unit. However, the weighing unit tares and no further anomalies are detected during the subsequent filling process. We can therefore show that, at least in this case, several continuous anomalies are detected, even though no sensors are explicitly installed for this case.

5 Conclusion

The main lessons learned from the practical application of HyOTALA are that the integration of domain knowledge during data preprocessing, training data selection and training of HyOTALA has a significant impact. It avoids the need to collect labeled anomaly samples because only a model of normal operation needs to be learned.
The transparency and interpretability of the algorithm was particularly helpful during the various modeling tests during training. In addition, the algorithm seems intuitive to humans, and most domain experts with no previous experience in machine learning were able to grasp the algorithm quickly. This aspect is particularly important because it promotes the acceptance of this algorithm within the company. The modeling results seem to be understandable so far. Another finding is that a short change from one state to another can be counterproductive in the case of continuously increasing or decreasing continuous values. Especially when the continuous processes are modeled as a function of time.

Going forward, the availability of more publicly available datasets with hybrid signals containing both continuous and discrete elements would be helpful. Such datasets are essential for rigorously testing and refining HyOTALA and similar algorithms, thereby improving their applicability to real-world hybrid scenarios. In addition, conducting more extensive experiments with HyOTALA could provide deeper insights into the effectiveness of online pipelines in different contexts. This would be particularly beneficial for newcomers to HyOTALA, providing them with a solid foundation and clear guidelines for implementation.
One of the most promising applications of HyOTALA is in anomaly detection, as suggested in the evaluation section 4. Further research should focus on developing a detailed framework and best practices for the use of hybrid timed automata in practical settings. In conclusion, while HyOTALA represents a significant advancement in the field of machine learning for CPPS, its full potential can only be realized through continued exploration, refinement, and application.

References

[1] V. Cobilean, H. S. Mavikumbure, C. S. Wickramasinghe, D. L. Marino, and M. Manic. Informed deep learning for anomaly detection in cyber-physical systems.

In *2023 IEEE International Conference on Industrial Technology (ICIT)*, pages 1–7. IEEE, 2023.

[2] A. A. Cook, G. Misirli, and Z. Fan. Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7:6481–6494, 2020.

[3] A. A. Efanov, S. A. Ivliev, and A. G. Shagraev. Welford's algorithm for weighted statistics. In *2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, pages 1–5. IEEE, 2021.

[4] C. Emmanouilidis, S. Waschull, J. A. Bokhorst, and J. C. Wortmann. Human in the ai loop in production environments. In *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems: IFIP WG 5.7 International Conference, APMS 2021, Nantes, France, September 5–9, 2021, Proceedings, Part IV*, pages 331–342. Springer, 2021.

[5] G. Frey and L. Litz. Formal methods in plc programming. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, volume 4, pages 2431–2436. IEEE, 2000.

[6] C. Gerking, D. Schubert, and E. Bodden. Model checking the information flow security of real-time systems. In *Engineering Secure Software and Systems: 10th International Symposium, ESSoS 2018, Paris, France, June 26-27, 2018, Proceedings 10*, pages 27–43. Springer, 2018.

[7] T. L. Hayes and C. Kanan. Online continual learning for embedded devices. *arXiv preprint arXiv:2203.10681*, 2022.

[8] N. Hranisavljevic, A. Maier, and O. Niggemann. Discretization of hybrid cpps data into timed automaton using restricted boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020.

[9] N. Hranisavljevic, O. Niggemann, and A. Maier. A novel anomaly detection algorithm for hybrid production systems based on deep learning and timed automata. *arXiv preprint arXiv:2010.15415*, 2020.

[10] P. Kamat and R. Sugandhi. Anomaly detection for predictive maintenance in industry 4.0-a survey. In *E3S web of conferences*, volume 170, page 02007. EDP Sciences, 2020.

[11] C. Krupitzer, T. Wagenhals, M. Züfle, V. Lesch, D. Schäfer, A. Mozaffarin, J. Edinger, C. Becker, and S. Kounev. A survey on predictive maintenance for industry 4.0. *arXiv preprint arXiv:2002.08224*, 2020.

[12] Q. Lin, S. Adepu, S. Verwer, and A. Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. In *Proceedings of the 2018 on asia conference on computer and communications security*, pages 525–536, 2018.

[13] Z. Lyu, G. Wang, and C. Tan. A novel bayesian multivariate linear regression model for online state-of-health estimation of lithium-ion battery using multiple health indicators. *Microelectronics Reliability*, 131:114500, 2022.

[14] A. Maier. Online passive learning of timed automata for cyber-physical production systems. In *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pages 60–66, 2014.

[15] A. Maier, O. Niggemann, and J. Eickmeyer. On the learning of timing behavior for anomaly detection in cyber-physical production systems. In *DX*, pages 217–224, 2015.

[16] A. Maier, O. Niggemann, R. Just, M. Jäger, and A. Vodenčarević. Anomaly detection in production plants using timed automata-automated learning of models from observations. In *International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 363–369. SciTePress, 2011.

[17] W. Marfo, D. K. Tosh, and S. V. Moore. Condition monitoring and anomaly detection in cyber-physical systems. In *2022 17th Annual System of Systems Engineering Conference (SOSE)*, pages 106–111. IEEE, 2022.

[18] A. P. Mathur and N. O. Tippenhauer. Swat: A water treatment testbed for research and training on ics security. In *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*, pages 31–36. IEEE, 2016.

[19] O. Modrlák and M. Sbieschni. Sequence control of a drilling machine by simatic s7-300. 2019.

[20] J. Montiel, M. Halford, S. M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H. M. Gomes, J. Read, T. Abdessalem, et al. River: machine learning for streaming data in python. 2021.

[21] O. Niggemann, B. Stein, A. Vodencarevic, A. Maier, and H. K. Büning. Learning behavior models for hybrid timed systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1083–1090, 2012.

[22] P. P. Ray. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, 34(4):1595–1623, 2022.

[23] H. Ren, D. Anicic, and T. A. Runkler. Tinyol: Tinyml with online-learning on microcontrollers. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[24] O. Serradilla, E. Zugasti, and U. Zurutuza. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospect. *ArXiv*, abs/2010.03207, 2020.

[25] D. Shetve, I. Raju, R. V. Prasad, R. Trestian, H. Nguyen, and H. Venkataraman. Adaptive n-step technique for real-time anomaly detection in smart manufacturing. *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 01–06, 2022.

[26] H. Sivan, M. Gabel, and A. Schuster. Online linear models for edge computing. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 645–661. Springer, 2020.

[27] K. Stefanidis and A. Voyiatzis. An hmm-based anomaly detection approach for scada systems. pages 85–99, 2016.

[28] G. Sugumar and A. Mathur. Assessment of a method for detecting process anomalies using digital-twinning. In *2019 15th European Dependable Computing Conference (EDCC)*, pages 119–126. IEEE, 2019.

[29] S. Verwer. Efficient identification of timed automata: Theory and practice. 2010.

[30] A. von Birgelen and O. Niggemann. Using self-organizing maps to learn hybrid timed automata in absence of discrete events. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2017.

[31] Y. Wu, H. Dai, and H. Tang. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 9:9214–9231, 2021.

REGRESSION VIA CAUSALLY INFORMED NEURAL NETWORKS

Shahenda Youssef[1], **Frank Doehner**[1], **and Jürgen Beyerer**[1,2]
[1]Karlsruhe Institute of Technology KIT, Karlsruhe, Germany
[2]Fraunhofer IOSB, Fraunhoferstraße 1, Karlsruhe

ABSTRACT

Neural Networks have been successful in solving complex problems across various fields. However, they require significant data to learn effectively, and their decision-making process is often not transparent. To overcome these limitations, causal prior knowledge can be incorporated into neural network models. This knowledge improves the learning process and enhances the robustness and generalizability of the models. We propose a novel framework RCINN that involves calculating the inverse probability of treatment weights given a causal graph model alongside the training dataset. These weights are then concatenated as additional features in the neural network model. Then incorporating the estimated conditional average treatment effect as a regularization term to the model loss function, the potential influence of confounding variables can be mitigated, leading to bias minimization and improving the neural network model. Experiments conducted on synthetic and benchmark datasets using the framework show promising results.

Keywords Neural Networks · Causal graphs · Prior knowledge · Causal inference · Propensity score weighting · Regression

1 Introduction

Despite the promising results of Neural Networks (NNs) across various fields, they still face unresolved challenges [17]. A key issue is their limited performance when there is a lack of training data, which affects their ability to generalize [27]. Additionally, the black-box nature of NNs prevents a precise explanation of their mechanism [27]. While they excel at uncovering concealed features and their co-occurrences within the input data, they struggle to uncover and clarify any underlying causal relationships between those features. To address these challenges, it's critical to incorporate causality into machine learning frameworks [26, 19].

Incorporating NN models with causality enhance their robustness and ability to generalize [27]. Furthermore, such frameworks can accurately model shifts in data distributions by concentrating on causal relationships, which are based on a sequence of cause and

effect rather than correlation, which simply observes patterns without implying direction. Therefore, causality provides NN with the capability to properly reason beyond its training data [26].

The adoption of deep learning in manufacturing systems is still in its early stages [15]. This could be attributed not only to its exclusive reliance on data-driven techniques but also to the lack of research conducted on incorporating domain experts' causal knowledge into deep learning models. Recent research is focused on improving model performance and explainability by integrating prior knowledge into the learning process [6, 1].

The methodologies to incorporate prior knowledge within NNs are varied and innovative [31, 5]. This involves embedding prior knowledge into NNs by transforming the input data [12, 9], imposing informed constraints on the loss function to direct the optimization process towards solutions that respect established relationships and theoretical frameworks [20, 8], and the integration into the architecture of NNs itself or constraining model parameters [3, 12]. The ongoing studies aim to combine different methods to ensure that the incorporated prior knowledge effectively guides the learning process, while still allowing the NNs to uncover new insights based on the data. However, the development of models that incorporate causal prior knowledge continues to be a challenge [27].

The subsequent sections of the paper are structured in the following manner: Section 2 provides an overview of causal inference, and section 3 addresses related work. In section 4 we present the proposed framework, followed by the experimental results in section 5. We conclude with section 6 outlining some major unsolved challenges and provide insight into potential directions for future research.

2 Causal Inference

Causal inference determines cause-and-effect relationships between variables based on observational data. It aims to estimate the causal effects of a specific treatment or intervention on an outcome of interest while considering potential factors that may introduce bias or influence the relationship [24].

2.1 Causal Graphs

Causal graphs, that do not contain any cycles between variables, can be represented as Directed Acyclic Graphs (DAGs). Such a DAG can be defined as $G = (V, E)$, where the causal graph G depicts the causal relationships between variables, nodes, V, with directed edges E between the nodes depicting the direction of cause and effect [32].

When estimating causal effects, it is important to consider two types of variables: Confounders and instrumental variables (IVs) [22]. As shown in figure 1, a confounder $C \in V$ is a variable that influences both the treatment $X \in V$ and the outcome $Y \in V$, which implies that any observed correlation between the treatment and the outcome might be due to the confounder's spurious correlations rather than a causal relationship. On the other hand, the instrumental variable $Z \in V$ is a variable that only affects the treatment and is not directly linked to the outcome.

2.2 Conditional Average Treatment Effect

Conditional Average Treatment Effect (CATE) measures the causal effect of the treatment on the outcome, conditioned on confounding variables or covariates [18]. The backdoor

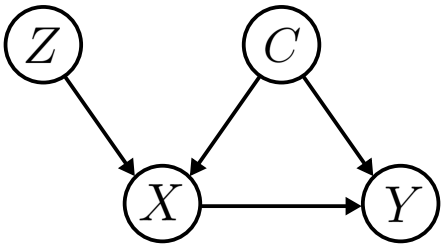


Figure 1: Causal Graph, $X \to Y$ indicates that the treatment node X is the cause of the outcome node Y. Node C is a confounding variable with an effect on X and Y, and Node Z is an instrumental variable with effect on X.

criterion introduced by Pearl [24] controls for confounding variables to obtain a more precise estimation of the causal effect. It identifies and blocks potential paths that might introduce bias between the treatment and outcome variables. The CATE is then defined by,

$$\tau_{\text{CATE}} = \mathbb{E}[Y \mid \text{do}(X = x)] = \mathbb{E}_C \mathbb{E}[Y \mid X = x, C = c], \tag{1}$$

where $\mathbb{E}[Y|\,\text{do}(X = x)]$ represents the expectation of the outcome Y under the intervention $\text{do}(X = x)$. The do-operator allows for the identification and estimation of causal effects from observational data under certain conditions. C represents the set of confounders of variables X and Y.

The instrumental variable method [10] estimates the effect of the instrument on the treatment and the outcome to estimate the effect of the treatment on the outcome. The CATE is then defined by,

$$\tau_{\text{CATE}} = \mathbb{E}[Y \mid \text{do}(X = x)] = \frac{\mathbb{E}[Y \mid Z = z]}{\mathbb{E}[X \mid Z = z]}\,, \tag{2}$$

where Z represents the set of IVs. It helps in identifying causal effects where the backdoor method fails [18].

2.3 Inverse Probability Weighting

Inverse Probability Weighting (IPW) [22] is a statistical technique that can reduce bias and provide more accurate estimates of causal effects. It involves assigning weights for each observation based on their estimated propensity scores $e(C)$, which represent the probability of receiving a particular treatment given a set of observed confounders, and it is represented as

$$e(C) = P(X = x \mid C = c). \tag{3}$$

IPW is a method for controlling the confounders by constructing pseudo-populations within the data and weighing them based on the inverse of their propensity scores α_{IPW} in order to decounfounded the data

$$\alpha_{\text{IPW}} = \frac{1}{P(X \mid C)}\,. \tag{4}$$

Pseudo-populations are created by upweighting the underrepresented and downweighted the uprepresented groups in the dataset [18]. This weighting aims to balance the distribution of covariates between the treatment and untreated groups, making them more comparable, and thereby reducing potential confounding effects.

3 Related Work

Deng et al. [7] present a deep learning framework for forecasting societal events that leverage causal inference. Their approach utilizes Individual Treatment Effects (ITE) to estimate the impact of different treatments or events on societal outcomes within spatiotemporal environments. The model integrates causal information into event predictions, by predicting potential outcomes for different treatment scenarios.

The work by Richens et al. [25] discusses the use of Structural Causal Models (SCM) as a means to encode the relationships between diseases, symptoms, and risk factors, enabling more accurate diagnostic reasoning. The authors develop algorithms that prioritize causal inference and highlight the importance of addressing confounding issues and the necessity of causal knowledge in the diagnostic process.

Kyono et al. [13] show how causal graphs can be used as prior knowledge to improve model selection and enhance the reliability of NN performances. They propose incorporating this knowledge into a Structural Causal Model to calculate a score that evaluates how well a model's predictions align with the SCM and input variables.

Teshima et al. [30] introduce a model-independent method for data augmentation that leverages the conditional independencies relations in the data distribution, encoded in causal graphs, to enhance supervised learning.

In a recent study Terziyan et al. [29] proposed a novel framework for enhancing Convolutional Neural Networks (CNNs) by incorporating causality-awareness into their architecture. An additional layer of neurons is introduced to the architecture that is specifically designed to estimate asymmetric causality in images by leveraging convolutional layers to extract features from images and then using these features to estimate conditional probabilities, effectively improving image classification and generation.

The expanding number of research emphasizes the importance of integrating causal regularization strategies into the framework of predictive modeling to address the issue of confounding variables in causal inference. By regularizing the model to consider causal relationships, it can provide more reliable and interpretable results [11, 14].

4 Method

We propose the novel framework RCINN that combines the strengths of causal inference and neural networks in regression tasks. The initial step involves encoding domain knowledge into a causal graph. If a causal graph is unavailable, causal discovery methods can be utilized to learn it from the data [16]. However, in this work, we are assuming that the causal graph is known.

Figure 2 shows the structure of our proposed, causally informed neural network framework RCINN for integrating causal prior knowledge into a neural network. Besides the usual training data $(\mathcal{D}, Y)$, where $\mathcal{D}$ are input features and Y are the true labels, the framework leverages additional prior knowledge from an independent source given by the DAG G.

In order to identify the CATE using either the backdoor criterion or the IV method discussed in 2.2, by using the Dowhy library for causal inference from Microsoft [28] to conduct an analysis of the given causal graph G regarding confounders and IVs.

Each observation in the training dataset is assigned a weight based on its corresponding inverse probability weight α_{IPW} as shown in equation 4. This weighting approach

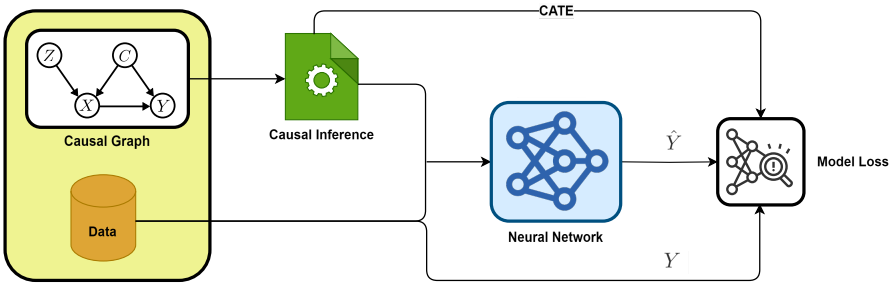


Figure 2: The proposed causally informed neural network framework RCINN.

prioritizes observations that are less likely, based on covariates, to receive the treatment they actually received. Adding these weighted feature values as an additional feature to the training dataset generates a new input features $\mathcal{D}_{\text{new}}$ that implies increasing the initial weights of the features with causal relationships when inputting them into the neural network model.

To estimate the causal effect τ_{CATE}, if the graph contains confounders, the CATE is estimated using the backdoor equation 1. On the other hand, for IVs, equation 2 is used. We scale the outcome by IPW to get an unbiased non-parametric CATE estimator

$$\tau_{\text{CATE}} = \mathbb{E}[\alpha_{\text{IPW}} \cdot \tau_{\text{CATE}}]. \tag{5}$$

To make sure that we have controlled for variables that contribute to bias. By using one of the refutation tests from Dowhy which is used to validate the causal estimates. It adds randomly generated covariates to the data, then reruns the estimator to return a tested causal estimate β and check if the causal estimate changes or not. The robust causal estimate should not change much with a small effect of the unobserved common cause. The neural network model is then trained with the new input features, by learning a function from the data $(\mathcal{D}_{\text{new}}, Y)$. The loss function of the regression neural network learning model is then computed as

$$\mathcal{L}_{\text{Pred}} = \mathcal{L}_{(Y,\hat{Y})} + \lambda \cdot W^2, \tag{6}$$

where $\mathcal{L}_{(Y,\hat{Y})}$ is the label-based loss that can be represented by the mean squared error, $Y, \hat{Y}$ are the actual labels and predicted values respectively. $\lambda \cdot W^2$ is the L2 regularization function used to control model complexity. It adds the squared values of the model weights W to the loss function and thereby encourages smaller weights and helps to prevent overfitting.

On top of the predictive loss, we added a regularization term causal loss $\mathcal{L}_{\text{Causal}}$ that penalizes the deviations from the causal estimate during neural network training. This is done by squaring the difference between the predicted outcomes and the estimated causal effect. The causal loss encourages the neural network to make predictions that are consistent with the causal relationships and robust to unobserved confounding, and it is defined as

$$\mathcal{L}_{\text{Causal}} = (\hat{Y} - \tau_{\text{CATE}})^2 + (\tau_{\text{CATE}} - \beta)^2, \tag{7}$$

By incorporating the causal loss term $\mathcal{L}_{\text{Causal}}$, we can mitigate the potential influence of confounding variables that may affect both the treatment and the outcome. We train the NN model by minimizing the following loss function

$$\mathcal{L}_{\text{Total}} = \mathcal{L}_{\text{Pred}} + \mu \cdot \mathcal{L}_{\text{Causal}}, \tag{8}$$

by multiplying $\mathcal{L}_{\text{Causal}}$ with the regularization hyperparameter μ, we control the strength of the regularization term. Higher values of μ increase the regularization effect, encouraging the model to respect the causal graph more strongly.

5 Experiments

We evaluate our method on both linear and non-linear regression datasets. The datasets include Linear, a linear synthetic dataset with three confounders and two IVs. Nonlinear, a non-linear synthetic dataset with three confounders. A benchmark Infant Health and Development Program (IHDP) dataset [28] with 25 confounders.

For a fair comparison, we adopt the same training setting and the same NN architecture for all models. For each task, we report the model loss with a standard deviation calculated over 5 different testing data. The data is divided into training, validation, and testing sets with proportions of 60%, 10%, and 30% respectively. The neural network architecture consists of two layers, with the first layer having 128 neurons and the second layer having 64 neurons. The output layer is the final layer of the neural network. During training, we adopt the Adam optimizer [2]. The training batch size is 32. We set λ as 0.1 and μ as 0.01 for all datasets.

Table 1: Performance of Regression tasks on three different datasets: Linear, Nonlinear, and IHDP. RF represents a Random Forest algorithm, Baseline_NN model is the neural network without causality and RCINN utilizes causal prior knowledge.

Datasets	RF	Baseline_NN	RCINN
Linear	0.227 ± 0.006	0.425 ± 0.006	$\mathbf{0.152 \pm 0.002}$
Nonlinear	34.853 ± 0.759	25.329 ± 0.593	$\mathbf{20.913 \pm 0.145}$
IHDP	1.95 ± 0.43	2.52 ± 0.11	$\mathbf{1.7 \pm 0.14}$

The results are reported in Table 1. We observe that our method demonstrates promising performance on both the synthetic datasets as well as on the benchmark dataset.

Overall, this approach allows us to leverage the strengths of both causal inference and neural networks, leading to a reduction in bias and an improvement in the performance of the neural network model.

6 Future Prospects

Much potential lies in exploring new combinations of approaches, that have not yet been investigated. One such example is merging causal prior knowledge with the architecture of NNs using the attention mechanism [23]. This allows the model to iteratively process knowledge by selecting only relevant content in each step. The inclusion of a knowledge-based attention layer improves prediction and overall model performance.

Another promising framework involves integrating a causal graph model in the form of an embedding graph layer, which can then be used as input for Bayesian Neural Networks (BNNs) [21]. This integration aims to enhance the incorporation of causal prior knowledge by refining the prior distribution during model training. Such a probabilistic approach takes the uncertainties in the model's predictions into account, recognizing that understanding the confidence level of a prediction is equally important as the prediction itself.

Recent research primarily focuses on data-driven approaches that assume independent and identically distributed (IID) data. However, when working with spatiotemporal data that deviates from this IID assumption, it becomes challenging [4]. Future work will be directed towards incorporating causal models capable of handling highly correlated values over time.

Acknowledgments

This work is funded by Research Group DFG FOR 5339.

References

[1] K. Beckh, S. Müller, M. Jakobs, V. Toborek, H. Tan, R. Fischer, P. Welke, S. Houben, and L. von Rueden. Explainable machine learning with prior knowledge: an overview. *arXiv preprint arXiv:2105.10172*, 2021.

[2] S. Bock and M. Weiß. A proof of local convergence for the adam optimizer. In *2019 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[3] A. Borghesi, F. Baldo, and M. Milano. Improving deep learning models via constraint-based domain knowledge: a brief survey. *arXiv preprint arXiv:2005.10691*, 2020.

[4] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[5] T. Dash, S. Chitlangia, A. Ahuja, and A. Srinivasan. A review of some techniques for inclusion of domain-knowledge into deep neural networks. *Scientific Reports*, 12(1):1040, 2022.

[6] A. Daw, A. Karpatne, W. D. Watkins, J. S. Read, and V. Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. In *Knowledge Guided Machine Learning*, pages 353–372. Chapman and Hall/CRC, 2022.

[7] S. Deng, H. Rangwala, and Y. Ning. Causal knowledge guided societal event forecasting. *arXiv preprint arXiv:2112.05695*, 2021.

[8] E. Gallup, T. Gallup, and K. Powell. Physics-guided neural networks with engineering domain knowledge for hybrid process modeling. *Computers & Chemical Engineering*, 170:108111, 2023.

[9] M. Gaur, K. Faldu, and A. Sheth. Semantics of the black-box: Can knowledge graphs help make deep learning systems more interpretable and explainable? *IEEE Internet Computing*, 25(1):51–59, 2021.

[10] M. A. Hernán and J. M. Robins. *Causal Inference: What If*. Boca Raton: Chapman Hall/CRC, 2020.

[11] L. Kania and E. Wit. Causal regularization: On the trade-off between in-sample risk and out-of-sample risk guarantees. *arXiv preprint arXiv:2205.01593*, 2022.

[12] S. W. Kim, I. Kim, J. Lee, and S. Lee. Knowledge integration into deep learning in dynamical systems: an overview and taxonomy. *Journal of Mechanical Science and Technology*, 35:1331–1342, 2021.

[13] T. Kyono and M. van der Schaar. Improving model robustness using causal knowledge. *arXiv preprint arXiv:1911.12441*, 2019.

[14] T. Kyono, Y. Zhang, and M. van der Schaar. Castle: Regularization via auxiliary causal graph discovery. *Advances in Neural Information Processing Systems*, 33:1501–1512, 2020.

[15] R. Malhan and S. K. Gupta. The role of deep learning in manufacturing applications: Challenges and opportunities. *Journal of Computing and Information Science in Engineering*, 23(6), 2023.

[16] D. Malinsky and D. Danks. Causal discovery algorithms: A practical guide. *Philosophy Compass*, 13(1):e12470, 2018.

[17] G. Marcus. Deep learning: A critical appraisal. *arXiv preprint arXiv:1801.00631*, 2018.

[18] A. Molak. Causal inference and discovery in python. *Small*, 344:344, 2023.

[19] R. Moraffah, M. Karami, R. Guo, A. Raglin, and H. Liu. Causal interpretability for machine learning-problems, methods and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(1):18–33, 2020.

[20] N. Muralidhar, M. R. Islam, M. Marwah, A. Karpatne, and N. Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pages 36–45. IEEE, 2018.

[21] K. P. Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

[22] B. Neal. Introduction to causal inference. *Course Lecture Notes*, 2020.

[23] Z. Niu, G. Zhong, and H. Yu. A review on the attention mechanism of deep learning. *Neurocomputing*, 452:48–62, 2021.

[24] J. Pearl and D. Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[25] J. G. Richens, C. M. Lee, and S. Johri. Improving the accuracy of medical diagnosis with causal machine learning. *Nature communications*, 11(1):3923, 2020.

[26] B. Schölkopf. Causality for machine learning. *Probabilistic and Causal Inference*, 2022.

[27] B. Schölkopf, F. Locatello, S. Bauer, N. R. Ke, N. Kalchbrenner, A. Goyal, and Y. Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[28] A. Sharma and E. Kiciman. Dowhy: An end-to-end library for causal inference. *arXiv preprint arXiv:2011.04216*, 2020.

[29] V. Terziyan and O. Vitko. Causality-aware convolutional neural networks for advanced image classification and generation. *Procedia Computer Science*, 217:495–506, 2023.

[30] T. Teshima and M. Sugiyama. Incorporating causal graphical prior knowledge into predictive modeling via simple data augmentation. In *Uncertainty in Artificial Intelligence*, pages 86–96. PMLR, 2021.

[31] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy, et al. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):614–633, 2021.

[32] D. B. West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

End-to-End MLOps Integration: A Case Study with ISS Telemetry Data

Henrik S. Steude [1], **Christian Geier** [2], **Lukas Moddemann** [1], **Martin Creutzenberg** [2], **Jann Pfeifer** [3], **Samo Turk** [4], **and Oliver Niggemann** [1]
[1]Helmut Schmidt University, Hamburg, Germany, firstname.lastname@hsu-hh.de
[2]Just Add AI, Bremen, Germany, firstname.lastname@justadd.ai
[3]Airbus Defence & Space, Bremen, Germany, firstname.lastname@airbus.com
[4] Lufthansa Technik, Hamburg, Germany, firstname.lastname@lht.dlh.de

Abstract

Kubeflow integrates a suite of powerful tools for Machine Learning (ML) software development and deployment, typically showcased independently. In this study, we integrate these tools within an end-to-end workflow, a perspective not extensively explored previously. Our case study on anomaly detection using telemetry data from the International Space Station (ISS) investigates the integration of various tools—Dask, Katib, PyTorch Operator, and KServe—into a single Kubeflow Pipelines (KFP) workflow. This investigation reveals both the strengths and limitations of such integration in a real-world context. The insights gained from our study provide a comprehensive blueprint for practitioners and contribute valuable feedback for the open source community developing Kubeflow.

Keywords MLOps · Kubeflow · ISS · Anomaly Detection

1 Introduction

MLOps, a fusion of Machine Learning (ML) and operations [10], has been an established concept for some years now. Especially since Sculley et al. [20] highlighted the complexity of operating ML applications in 2015. As a result, numerous new MLOps tools have been developed in recent years [14]. In parallel containerization of software gained traction as a popular way of ensuring compute workload resource allocation and user isolation. Kubernetes is one of the most popular container orchestration platforms providing abstraction to underlying infrastructure [2]. Among MLOps tools, Kubeflow emerged as a notable Kubernetes-based MLOps environment. It facilitates the integration of multiple ML tools and has gained attention in the community, with notable recognition including its incorporation in the CNCF Incubator [3].

Kubeflow is designed to support the entire lifecycle of ML projects, encompassing everything from data exploration to model development and retraining. Within this framework, Kubeflow integrates a variety of tools for distinct phases of the ML workflow.

However, existing documentations often focus more on the standalone functionalities of these tools, rather than exploring their combined potential in a cohesive workflow [7, 12, 11].

In response to this, our study aims to examine the practical application of these tools in an automated end-to-end workflow. Specifically, our research questions are as follows:

- Can Kubeflow's MLOps tools (e.g., Katib, PyTorch Operator, KServe) be effectively integrated into a pipeline for training, deploying, and re-training deep Neural Networks (NN) in an automated manner?
- What challenges exist in such pipelines and how can they be tackled?

To answer these questions, we conduct a case study based on a real-world application. The case study involves training and deploying an anomaly detection model for a subsystem of the International Space Station's (ISS) Columbus module. This use case presents an ideal scenario for our study due to the complexity of the data, which is more intricate than in most standard examples for ML tools. While MLOps spans a broad spectrum, including non-technical aspects such as Change Management and Teamwork, our study is primarily centered on the technical aspects.

Through this research, we contribute to the field of MLOps in two primary ways. Firstly, we have implemented an ML-workflow that showcases the integration of multiple tools within a single Kubeflow Pipelines (KFP) framework. The source code of our implementation is available on GitHub[1], offering a resource for practitioners who wish to explore similar applications or build upon our work. Secondly, we provide a detailed analysis of our case study, highlighting the functionalities that Kubeflow efficiently supports and identifies areas where further development could be beneficial. Our contributions to the open-source community, including reporting issues and submitting fixes, are part of our effort to refine these tools for enhanced usability and functionality.

2 Related Work

This section aims to situate our study in the broader context of current scientific research, offering an overview of the latest MLOps tools and reviewing recent studies that have employed case studies in MLOps.

State-of-the-art MLOps tools Recent publications compared Kubeflow with other tools for MLOps and evaluated them based on their ability to handle different phases of the MLOps process. E.g. Ruf et al. [19] divided the end-to-end MLOps process into five phases: *Data Preprocessing*, *Model Training*, *Model Management*, *Model Deployment*, and *Operations and Monitoring*. They evaluated 26 different tools for MLOps based on their ability to handle these phases. Among the evaluated tools, only three offered features for all five phases: Kubeflow, Polyaxon, and TFX. Two other tools offered features for four of the five phases: ZenML (excluding *Operations and Monitoring*) and Flyte (excluding *Data Preprocessing*). Köhler et al. [15] evaluated 30 different tools for MLOps based on three selection criteria: *(i)* the tool must be end-to-end, *(ii)* the tool must be marketed as Kubernetes native, and *(iii)* the tool must be open-source. Among the evaluated tools, Kubeflow, Polyaxon, and Pachyderm were the only ones that fulfill all three criteria.

Further, GitHub stars can be used as a measure of popularity for open-source software and the strength of their community [1]. See Table 1 for an overview of GitHub stars,

[1] https://github.com/hsteude/code-ml4cps-paper

as of January 2024, for the MLOps tools mentioned above. Considering this metric,

MLOps Tool	Number of GitHub Stars
Kubeflow	13.3k
Pachyderm	6k
Flyte	4.3k
Polyaxon	3.4k
ZenML	3.4k
TFX	2k

Table 1: Popularity of MLOps Tools based on GitHub Stars

Kubeflow appears to be the most popular among these tools and stands out as one of the few capable of handling the entire MLOps process end-to-end.

Case studies of MLOps in different disciplines Kubeflow has found application across diverse research domains, serving as a pivotal tool in bioinformatics for achieving rapid scaling through containerization [25]. Furthermore, researchers have utilized Kubeflow to establish automated ML workflows tailored for a service-aware 5G network model [24]. The adoption of a cloud-native approach emerges as a strategic choice, facilitating seamless deployment of workloads, even on public cloud resources. A Kubeflow platform was also successfully implemented in the context of CERN, where it was employed for expeditious simulation of electromagnetic showers using generative deep learning techniques [8]. In the realm of ML use case applications, various papers describe MLOps workflows designed to enhance collaboration and negotiation dynamics between surgeons and patients [9]. Additionally, there are instances where ML pipelines deployed on the cloud have been instrumental in drug discovery processes [22].

To our best knowledge, there are no existing studies that investigate the integration of Kubeflow's tools into an end-to-end workflow. Before presenting our case study in detail in Section 4, we will introduce the corresponding tools in the next section.

3 Method

As motivated in Section 1, our study aims to examine the integration of key MLOps tools within Kubeflow using the KFP framework. In this section, we describe the tools provided by Kubeflow that we have utilized in our case study. The specific application and purpose of these tools in the context of our case study will be detailed in Section 4.

To position these tools within the general ML workflow, we align them with the stages of the Cross-Industry Standard Process for Data Mining (CRISP-DM) [21] (see Figure 2), a process model that continues to be widely accepted, even over two decades after its introduction [16]. It is important to note that these tools are ML method agnostic and are applicable across a range of methodologies including supervised, unsupervised, and reinforcement learning.

Kubeflow Notebooks: For the initial stages of CRISP-DM, such as business understanding and data understanding, Kubeflow Notebooks offer an interactive environment for exploratory data analysis. Various web-based development environments, such as JupyterLab [18], can be utilized to perform interactive and visual data analyses. These environments run within Kubernetes, allowing for scalability in terms of resources (CPU/GPU and memory requests and limits) and enabling programmatic control of other Kubeflow tools described subsequently.

Dask: Although Dask [4] is not a part of Kubeflow, it complements the other tools described in this section well, especially as Kubernetes is one of the possible backends for Dask. Dask enables distributed processing of large datasets and offers a programming interface closely resembling those of Pandas [23]. Thus, it is frequently used in the 'Data Preparation' phase of the CRISP-DM cycle.

Katib: In the 'Modeling' phase, Katib [7] serves as a Kubernetes-native solution for automated ML (also called *AutoML* in the Kubeflow user interface), and especially hyperparameter tuning. Katib supports a wide range of popular ML frameworks, including PyTorch, TensorFlow, and scikit-learn, and implements various optimization algorithms for hyperparameter tuning. It enables the parallel execution of extensive hyperparameter tuning experiments, utilizing Kubernetes as its backend.

Training Operators: For better support of the training phase of ML models, Kubeflow includes Training Operators [2] [12]. They facilitate various training methodologies, including data- and model-parallel training, which are crucial for handling large NNs and extensive datasets, particularly important when dealing with foundation models. These Training Operators can also be integrated into Katib experiments.

KServe: In the 'Deployment' stage of CRISP-DM (not to confuse with the Kubernetes concept of *Deployments*), KServe [11] significantly streamlines the model deployment process. As a serverless[3] framework, it reduces deployment complexities for developers and provides optimized, high-performance inference capabilities. KServe includes features such as auto-scaling and built-in logging and monitoring, facilitating efficient model serving in production environments.

Kubeflow Pipelines (KFP): As a key component in Kubeflow, KFP [13] links various stages of the CRISP-DM cycle. It provides a framework for building, deploying, and managing scalable ML workflows. KFP executes distinct processing steps, known as pipeline components, arranged in a Directed Acyclic Graph (DAG) to ensure efficient data flow and adherence to workflow dependencies. Components within KFP can output data as either parameters for small datasets or as artifacts for larger files. KFP also tracks the data generated, orchestrating its flow between components and keeping records of each pipeline run and its artifacts.

In our case study, we focus on the practical application of a KFP setup, orchestrating the range of tools outlined in this chapter. The next section will delve into the specifics of this implementation, detailing the experiment and elucidating its results.

4 Results

Within this section we start with a description of the use case itself, followed by an outline of the specific steps of the ML pipeline developed for this purpose. The focus is on illustrating the integration and automation of the tools, as discussed in Section 3, within the framework of a Kubeflow pipeline.

[2] In Kubernetes, an 'Operator' is a method of packaging, deploying, and managing a Kubernetes application. Leveraging the built-in Kubernetes extension capacity, 'custom resources', Operators describe application-specific operational knowledge for managing the application's entire lifecycle. This encapsulation of operational procedures facilitates automation of routine tasks like deployment, scaling, failure recovery and configuration changes, thereby enhancing system reliability and efficiency.

[3] Serverless computing is an execution model where developers don't have to be concerned with capacity planning, management and scaling of their applications.

4.1 Use case

This case study is motivated by the need to identify anomalies in the telemetry data of the ISS, particularly within the Columbus module. Launched into orbit in February 2008 as a key contribution from the European Space Agency (ESA) with Airbus as prime contractor, the Columbus module offers a unique microgravity environment for a wide range of research activities [5]. It is continuously monitored and controlled by the Columbus Control Center (COL-CC) at the German Aerospace Center (DLR) in Oberpfaffenhofen, Germany.

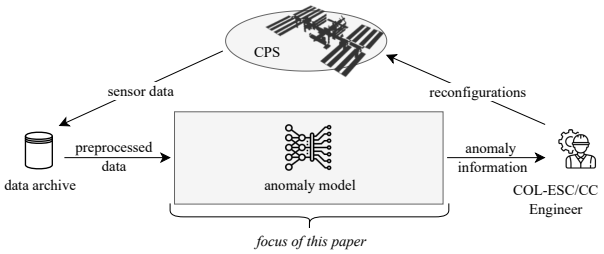


Figure 1: High-level overview of the information flow in the ISS diagnosis system.

From the business perspective, the primary focus of this case study is on the Environmental Control and Life Support System (ECLSS) of the Columbus module. ECLSS is crucial for maintaining a habitable environment within the module by regulating air circulation, temperature, and humidity. The goal of the project is to support the engineers at the Airbus-operated Columbus Engineering Support Center (COL-ESC) in detecting anomalies within the Columbus module's telemetry data stream using ML. This approach aims to enhance the efficiency of their daily operations and aid in the identification of hard-to-detect anomalous conditions. Examples of the varying nature and root causes of these anomalies can be found in the five-year operation report for the Columbus ECLSS subsystem [17].

Figure 1 depicts the information flow in the ISS anomaly detection use case. The telemetry data is transmitted to the ground stations, where the data is distributed, calibrated and archived. This preprocessed data is subsequently used for the anomaly detection model, both for training on historical data and inference on the incoming telemetry stream. For every anomaly detected by the system, information about time ranges and relevant subsystems are forwarded to the engineers in the COL-ESC. Following their investigation, they can decide to initiate countermeasures, such as suggesting reconfigurations of the system.

Columbus' telemetry data consist of thousands of distinct parameters, collected at a minimum frequency of one Hertz, streamed to Earth and archived over many years. A subset of this data has been labeled for the presence of anomalies by the responsible engineers, providing a crucial dataset for analysis.

From a technical perspective, this use case entails several typical ML project requirements: *(i)* exploration and preprocessing of large datasets, *(ii)* training of deep neural networks including hyperparameter tuning, *(iii)* deployment for real-time inference on streaming data, and *(iv)* regular retraining to accommodate data drift. In line with MLOps best practices, this project aims for a high degree of automation and full reproducibility.

4.2 ML-Pipeline

The pipeline implemented for this study encompasses several stages, structured to address distinct aspects of the ML workflow. These stages are: *(i)* initial data preparation, *(ii)* parallelized preprocessing, *(iii)* segmentation of telemetry data into training, development, and testing datasets, *(iv)* hyperparameter tuning, *(v)* model training, *(vi)* model evaluation, and *(vii)* model deployment. Figure 2 visualizes the pipeline as realized in KFP at a high level aligning the steps with the CRISP-DM cycle. The complete DAG and the associated source code[4] are available in our GitHub repository[5].

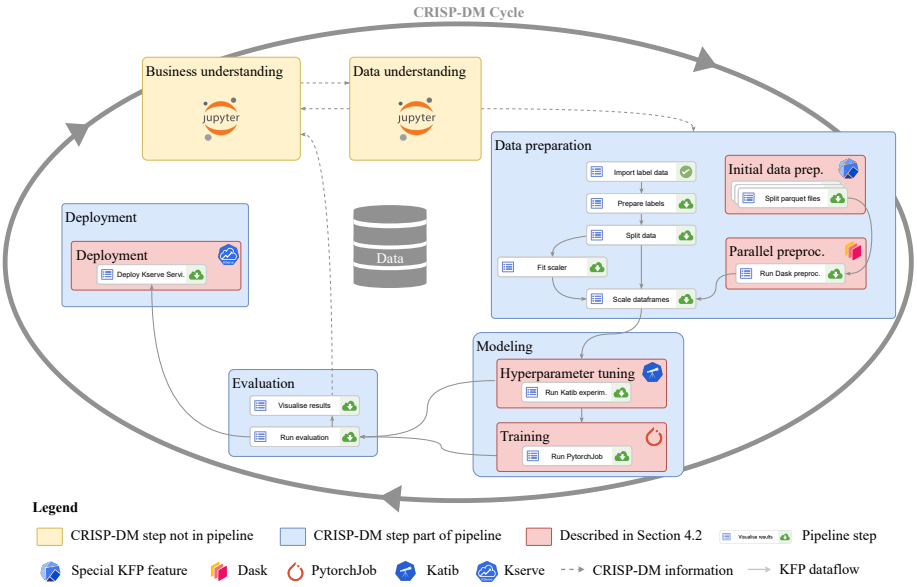


Figure 2: High-level overview of the ML pipeline implemented within KFP. This schematic illustrates the structured workflow aligned with the CRISP-DM process stages, integrating various tasks such as data preparation, modeling, hyperparameter tuning, and deployment. Key stages that incorporate additional Kubeflow tools are highlighted with red borders.

In the following, we will focus on the components highlighted in red in Figure 2, which are important to our research questions. These components demonstrate the integration of additional tools, namely Daks, Katib, PytorchJob and Kserve.

Initial Data Preparation The ECLSS subsystem's telemetry data were provided as annual Parquet files, each containing more than 1,000 time series sampled at 1Hz. Due to their large size, conventional data processing tools such as Pandas were inadequate. Additionally, the use of Dask for parallel processing necessitated the division of data into smaller segments. A KFP component was developed for this purpose, systematically dividing large files into smaller, processable units. Using this component, all the yearly archives files are split in a parallel for loop implemented using the KFP SDK.

[4]The source code uses the KFP SDK v2 and was tested on KFP backend version 2.0.0
[5]https://github.com/hsteude/code-ml4cps-paper

Parallel Preprocessing For preprocessing, Dask was employed to harness the full computational capacity of the Kubernetes cluster. The establishment of a Dask cluster, facilitated by the Dask Kubernetes operator and related Python package, enabled parallel data processing tasks within the pipeline. The cluster, composed of multiple worker pods[6], was dynamically assembled and subsequently dismantled after task completion, optimizing resource utilization.

Hyperparameter tuning For the hyperparameter tuning process, following the preparation of training, development, and testing datasets, we utilized Katib. This process involved utilizing the Kubernetes Python SDK for configuring and launching Katib training sessions within the pipeline. A key challenge encountered was the integration of Katib with KFP's data management system. Unlike other stages in the pipeline, where input/output operations are seamlessly managed by KFP as artifacts with data stored in object storage, Katib trials require a different approach for data access. To circumvent this limitation, we implemented custom code to handle data retrieval directly from object storage using the s3fs library for each trial.

Training Optimized parameters obtained in hyperparameter tuning process were passed to the PyTorch training job component. This stage required parallel training across multiple GPUs and nodes within the Kubernetes cluster, utilizing the Distributed Data Parallel (DDP) algorithm. Similar to the challenges faced during the hyperparameter tuning phase, the training job also encountered I/O challenges due to the worker pods not being managed by KFP. We tackled this as in the Katib job by reading from and writing data to object storage.

Deployment Model deployment, the final step in the pipeline, was facilitated by KServe. KServe utilizes model training artifacts (e.g., scaler, model, etc.) to establish a REST endpoint for inference. Similar to Katib and training job operators, KServe is not directly integrated into pipelines, thus it cannot leverage seamless artifact handling. Further, users are required to explicitly implement the configuration (i.e., manifests) and code to deploy the inference service from the pipeline. Finally, while permissions for most services/steps in the pipeline are pre-configured, an administrator must specifically grant users permission to deploy models using KServe.

Model evaluation Though the primary focus of this study is to demonstrate the integration of various Kubeflow tools, we briefly discuss the outcomes of the model evaluation to illustrate the applicability of the implemented pipeline here.

A KFP component was developed for model evaluation. This component loads the trained model and performs inference on the test set, which was neither used for training nor validation. To assess the model's performance, we adopted the *Composite F-Score*, as introduced in [6]. This score, designed for anomaly detection in multivariate time series, optimizes a balance between time-wise precision and event-wise recall, under the assumption that an ideal anomaly detector should identify at least one point per anomaly event while minimizing false positives.

Additionally, scatter plots were generated to visualize the temporal progression of the model's likelihood scores against the true anomaly labels (see Figure 3). KFP offers the capability to display these metrics and visualizations within its pipeline UI, enabling comparison across different pipeline runs with varying input parameters. For

[6] A pod is the smallest unit of computing on Kubernetes and is composed of a one or more containers sharing storage and network.

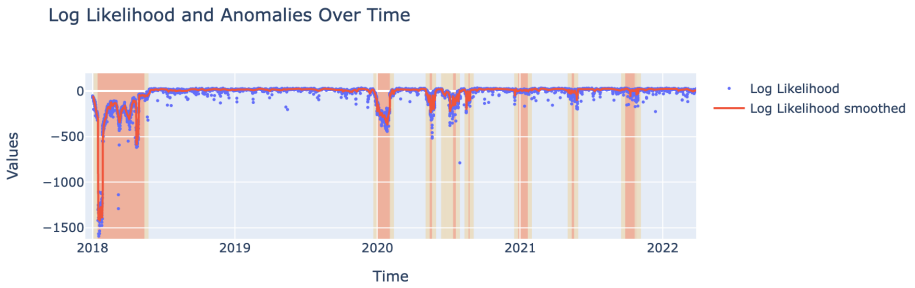


Figure 3: Log likelihood (blue dots) of anomaly detection over time, with a smoothened curve (red line) for trend visualization. Red shaded areas represent periods with labeled anomalies. The yellow shaded areas denote segments surrounding the anomalies that are exclusive to the test set and have been labeled as non-anomalous.

our experimentation, we deployed the pipeline on a Kubernetes cluster comprising three nodes, each equipped with 128 CPU cores and 500GB of memory, alongside a total of 4 NVIDIA A30 GPUs. The input parquet files amounted to 7.5GB. In our case study, the implemented anomaly detection model achieved a *Composite F-Score* of 0.77, which is considered satisfactory given the complexity of the task.

5 Discussion

In this paper, we have explored the orchestration of multiple tools within Kubeflow making use of the KFP framework for tying several tools and other frameworks together and demonstrated that such integration is indeed feasible. This section aims to discuss the aspects of our implementation that were particularly successful, as well as those that presented challenges, necessitating workarounds.

The points raised in this section are a curated selection, chosen to emphasize on those aspects we deem most significant for users considering Kubeflow and for contributors involved in its ongoing development. The highlighted strengths and challenges are based on our direct experiences, serving both as a guide for prospective adopters and as constructive feedback to inform continuous improvement of the platform.

In the course of our work with the Kubeflow framework, we identified several key strengths that facilitated the orchestration of various MLOps tools:

- **Compute Resource Sharing:** Kubeflow is well-suited for sharing Kubernetes cluster resources among teams, as pods release all resources upon job completion. This is also true for notebooks, which scale resources based on utilization between user-defined requests and limits, and also holds for Katib experiments and training jobs.
- **Compute Resource Merging** The training operators facilitate Neural Network training on multiple GPUs and compute nodes simultaneously, allowing not only efficient resource sharing but also the consolidation of these resources for intensive tasks.
- **Unified Environment for the Entire Process:** Theoretically, data scientists do not need to leave the Kubeflow environment to complete the entire described process, with the potential to work entirely within a browser-based interface.

- **Caching of Task Results:** KFP allows for caching, which is particularly useful for long-running processes at the pipeline's start. Changes towards the pipeline's end, like evaluation or result visualization, do not necessitate re-running these processes, ensuring reproducibility while managing artifacts from each step.
- **Effective Collaboration:** Inviting team members into one's Kubeflow namespace simplifies collaboration on common pipelines or notebooks.
- **High Level of Automation:** As the pipeline encapsulates the entire end-to-end process of model creation, it facilitates the fully automated generation and deployment of new model versions. Such retraining can be triggered by the availability of new data, detection of model drift, or after a predefined period of time.

Despite the successes, our implementation faced several challenges:

- **Need for More Comprehensive Documentation:** Compared to other ML frameworks, Kubeflow's documentation is less developed and would benefit significantly from more end-to-end examples.
- **Pipeline I/O Limitations:** While KFP simplifies pipeline development and usage, its I/O tools are not fully compatible with PyTorch jobs, Katib experiments or KServe. Integrating those tools into the KFP workflow required implementing custom workarounds. While writing custom code for data access is not necessarily a problem, it does require a deeper understanding of the underlying tools and their APIs.
- **Limitations of the Open-Source Server-Side Kubeflow Pipelines:** The server-side component of Kubeflow Pipelines is offered in both open-source and proprietary forms. The open-source variant has not yet fully implemented certain control flow features, such as advanced options for if-conditions and parallel for loops, which are available in the proprietary versions.
- **Ongoing Development and Emerging Issues:** During our case study implementation, we encountered features that were only available in SDK v1, which has since been replaced by SDK v2. This highlights Kubeflow's active development stage, indicating that users should be prepared for minor bugs and evolving features.

To summarize, our case study has underscored Kubeflow's potential as a robust platform for managing the complexities of MLOps tasks, thanks to its resource management capabilities, cohesive environment, and experiment tracking features. Nevertheless, the challenges we have outlined, particularly concerning documentation and tool integration, suggest that there is still room for refinement to fully realize this potential.

6 Conclusion

In this paper, we have explored the feasibility of integrating and automating a suite of MLOps tools within the Kubeflow framework, with a focus on a case study involving anomaly detection in telemetry data from the ISS Columbus module. Our research has successfully demonstrated that such integration is achievable, providing insights into the practical application of Kubeflow for complex MLOps tasks.

While we have identified KFP as an effective platform for managing MLOps tasks, we also encountered challenges, particularly in terms of documentation and tool integra-

tion. These challenges highlight that Kubeflow, although promising, requires further development and refinement to fully realize its potential.

In summary, our study shows that Kubeflow is a robust tool for MLOps, particularly distinguished by its resource management capabilities, provisioning of a cohesive environment for the entire ML process, and strong community support. Despite the challenges observed, our work contributes to the MLOps community by demonstrating a pathway for efficiently managing complex ML workflows.

For future work, we believe that focusing on improving Kubeflow's documentation, addressing its pipeline I/O limitations, and enhancing tool compatibility are among the most important aspects.

Acknowledgments

This research – as part of the *(K)ISS* project – is funded by *dtec.bw – Digitalization and Technology Research Center of the Bundeswehr*. dtec.bw is funded by the European Union through *NextGenerationEU*.

References

[1] H. S. Borges, A. Hora, and M. Valente. Understanding the factors that impact the popularity of GitHub repositories, 2016.

[2] E. Casalicchio and S. Iannucci. The state-of-the-art in container technologies: Application, orchestration and security. *Concurrency and Computation: Practice and Experience*, 32(17):e5668, 2020.

[3] Cloud Native Computing Foundation. Kubeflow brings MLOps to the CNCF incubator. `https://www.cncf.io/blog/2023/07/25/kubeflow-brings-mlops-to-the-cncf-incubator/`, 2023. Accessed: January 11, 2024.

[4] Dask Development Team. Dask: Parallel computing in python. `https://github.com/dask/dask`, 2023. Accessed: January 11, 2024.

[5] eoPortal Authors. ISS: Columbus module. `https://www.eoportal.org/satellite-missions/iss-columbus`, 2012. Accessed: January 20, 2024.

[6] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Trans Neural Netw Learn Syst*, PP, Aug. 2022.

[7] J. George, C. Gao, R. Liu, H. G. Liu, Y. Tang, R. Pydipaty, and A. K. Saha. A scalable and cloud-native hyperparameter tuning system, 2020.

[8] D. Golubovic and R. Rocha. Training and serving ml workloads with kubeflow at cern. In *EPJ Web of Conferences*, volume 251, page 02067. EDP Sciences, 2021.

[9] T. Granlund, V. Stirbu, and T. Mikkonen. Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product. *SN computer Science*, 2(5):342, 2021.

[10] D. Kreuzberger, N. Kühl, and S. Hirschl. Machine learning operations (MLOps)): Overview, definition, and architecture. *IEEE Access*, 11:31866–31879, 2023.

[11] KServe Authors. Kserve. `https://github.com/kserve/kserve`, 2023. Accessed: January 11, 2024.

[12] Kubeflow Authors. Training operator. `https://github.com/kubeflow/training-operator`, 2023. Accessed: January 11, 2024.

[13] Kubeflow Pipelines Authors. Kfp. `https://www.kubeflow.org/docs/components/pipelines/`, 2023. Accessed: January 11, 2024.

[14] I. Kumara, R. Arts, D. Di Nucci, W. J. Van Den Heuvel, and D. A. Tamburri. Requirements and reference architecture for MLOps:Insights from industry. *Authorea Preprints*, Nov. 2023.

[15] A. Köhler. Evaluation of MLOps tools for kubernetes: A rudimentary comparison between open source Kubeflow, Pachyderm, and Polyaxon. `https://www.diva-portal.org/smash/get/diva2:1711840/FULLTEXT01.pdf`, October 2022. IT 22 119.

[16] F. Martinez-Plumed, L. Contreras-Ochando, C. Ferri, J. Hernandez-Orallo, M. Kull, N. Lachiche, M. J. Ramirez-Quintana, and P. Flach. CRISP-DM twenty years later: From data mining processes to data science trajectories. *IEEE Trans. Knowl. Data Eng.*, 33(8):3048–3061, Aug. 2021.

[17] P. Parodi, Z. Szigetvari, R. Muller, and A. Quaglia. Columbus ECLSS: five years of operations and lessons learned. In *43rd International Conference on Environmental Systems*, 2013.

[18] J. M. Perkel. Why Jupyter is data scientists' computational notebook of choice. *Nature*, 563(7729):145–146, Nov. 2018.

[19] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam. Demystifying MLOps and presenting a recipe for the selection of open-source tools. *Applied Sciences*, 11(19):8861, 2021. Accessed: Januar 8, 2024.

[20] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison. Hidden technical debt in machine learning systems. *Adv. Neural Inf. Process. Syst.*, 2015.

[21] C. Shearer. The CRISP-DM model: The new blueprint for data mining. *Journal of Data Warehousing*, 5(4), 2000.

[22] O. Spjuth, J. Frid, and A. Hellander. The machine learning life cycle and the cloud: implications for drug discovery. *Expert opinion on drug discovery*, 16(9):1071–1079, 2021.

[23] The pandas development team. pandas-dev/pandas: Pandas, Feb. 2020.

[24] T. Tsourdinis, I. Chatzistefanidis, N. Makris, and T. Korakis. AI-driven service-aware real-time slicing for beyond 5G networks. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.

[25] D. Y. Yuan and T. Wildish. Bioinformatics application with kubeflow for batch processing in clouds. In *International Conference on High Performance Computing*, pages 355–367. Springer, 2020.