Causal models for specifying requirements in industrial ML-based software: A case study

;

Available online 5 November 2025

Contents lists available at ScienceDirect

The Journal of Systems & Software

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

Causal models for specifying requirements in industrial ML-based software: A case study

Hans-Martin Heyn [a,*], Yufei Mao [b], Roland Weiß [b], Eric Knauss [a]

[a] Department of Computer Science and Engineering, University of Gothenburg and Chalmers, Gothenburg, 41756, Sweden
[b] Siemens AG, München, 80333, Germany

ARTICLE INFO

Editor: Prof Neil Ernst

Keywords:
Anomaly detection
Causal analysis
Causality
Industrial systems
Machine learning
Requirement engineering
Systems engineering

ABSTRACT

Unlike conventional software systems, where rules are explicitly defined to specify the desired behaviour, software components that incorporate machine learning (ML) infer such rules as associations from data. Requirements Engineering (RE) provides methods and tools for specifying the desired behaviour as structured natural language. However, the inherent ambiguity of natural language can make these specifications difficult to interpret. Moreover, it is challenging in RE to establish a clear link between the specified desired behaviour and data requirements necessary for training and validating ML models.

In this paper, we explore the use of causal models to address this gap in RE. Through an exploratory case study, we found that causal models, represented as directed acyclic graphs (DAGs), support the collaborative discovery of an ML system's operational context from a causal perspective. We also found that causal models can serve as part of the requirements specification for ML models because they encapsulate both data and model requirements needed to achieve the desired causal behaviour. We introduce a concept for *causality-driven development*, in which we show that data and model requirements, as well as a causal description of the operational context, can be discovered iteratively using graphical causal models. We demonstrate this approach using an industrial use case on anomaly detection with ML.

1. Introduction

Developing and deploying software systems that incorporate machine learning (ML) models are becoming routine processes in many different industries. Unlike the development of conventional software with a priori defined rules, developing software with ML models is a data driven process: Especially deep learning systems are "opaque learning machines" (Pearl, 2019) because they rely on statistical learning to discover associations among implicit variables from observational data (Peters et al., 2017). Many industrial applications require *robustness* of the employed ML models against changes in the input data distribution (Borg et al., 2019). A lack of robustness against changes in the input data distribution not only compromises reliability and eventual safety of a depending system, but also renders the system susceptible to *adversarial attacks* (Carlini and Wagner, 2017; Goodfellow et al., 2018). One reason for the lack of robustness in ML models can be found in the challenges to specify the models, because "if input and/or output data are high-dimensional, both defining preconditions and detailed function specifications are difficult" (Kuwajima et al., 2020). *Assumptions* about the operational context in which the ML model is deployed are often implicitly included during the design process (Mitchell et al., 2021), for example in the dataset used for training. However, robustness against (small) context changes can only be tested if the expected operational context has been made explicit, for example in the form of contextual requirements (Knauss et al., 2014, 2016).

Understanding the system's context, or problem domain, from a data perspective however is not solved in requirement engineering (RE) (Habiba et al., 2024). Current RE methods, such as behaviour-driven development (BDD) or goal modelling do not work well for systems with ML components because they cannot systematically approach the problem of specifying the necessary data based on the system's expected operational context (Ahmad et al., 2023). As a result, many ML development projects spend more than 80 % of the project time on the elicitation of data requirements and data preparation (Pei et al., 2022).

An important, and in RE not yet well explored aspect, of the operational context for systems with ML models are causal structures that determine the direction of cause and effect. Recent work suggests that causal probabilistic graph-based models are a powerful tool for capturing the context of a system from a causal perspective (Maier et al., 2024). In this paper, we propose a system specification concept called *causality-driven development* (CDD), in which graphical causal models play an important role in exploring and communicating assumptions about the operational context, desired functionality of ML models, and data needed to ensure correct causal behaviour of the system. We operationalised this approach through an exploratory case study in which we applied graphical causal models for system prototyping in an industrial

* Corresponding author.
E-mail addresses: hans-martin.heyn@gu.se, martin.heyn@gmail.com (H.-M. Heyn), tufei.mao@siemens.com (Y. Mao), rolandweiss@siemens.com (R. Weiß), Eric.Knauss@cse.gu.se (E. Knauss).

https://doi.org/10.1016/j.jss.2025.112691

Received 8 May 2025; Received in revised form 27 August 2025; Accepted 27 October 2025

setting. With CDD, we suggest a concept to RE that connects contextual assumptions and data requirements through graphical causal models for which a rich mathematical framework exists that can be used to derive data requirements (Schölkopf, 2022). Specifically, this paper contributes in the following ways:

- **C1** Based on a series of workshops with industrial practitioners, we propose a concept for causality-driven development (CDD) for software with ML components. This technique complements natural language requirements-driven software development.
- **C2** We demonstrate the application of CDD in practice on an industrial use case involving anomaly detection in low-voltage DC switching gear.
- **C3** We report experimental results suggesting that CDD has a positive impact on the performance and robustness of a trained ML model for anomaly detection in an industrial prototyping environment.

The article is structured as follows: Section 2 provides background information on causal models and Section 3 an overview of related work. Section 4 outlines the research methods of this study. Section 5 introduces CDD as a complement to requirements elicitation for software of industrial cyber-physical system (CPS) with ML models and their training data. Section 6 presents results from a case study where we applied CDD in an industrial context. Section 7 discusses the application of causal modelling in the context of systems prototyping, outlines limitations of CDD, offers suggestions for future research, as well as addresses threats to validity. Section 8 provides a conclusion.

2. Background

2.1. ML development pipeline

In an ML development pipeline, system goals and high-level requirements are met by accumulating data to train an ML model until the stakeholder needs are met. Steps in an ML development pipeline typically include:

1. System goals and context assumptions: Business and problem understanding;
2. Building datasets: Data collection, understanding, and preprocessing;
3. Building the ML model: Model selection and training on the prepared datasets;
4. Analysing: Evaluation and tuning of the model;
5. Deploying: Final deployment and monitoring in the field.

These steps are typically iterative because if the evaluation of the final model is unsatisfactory, additional data may be collected, and the model is retrained and re-evaluated. ML development pipelines often base on the cross-industry standard process for data mining (CRISP-DM), originally introduced by Wirth and Hipp (2000) and Chapman et al. (2000), and widely applied in both data science and machine learning problems (Schröer et al., 2021; Martínez-Plumed et al., 2019). This workflow for ML model development has several limitations. First, data requirements derived from high-level system goals are often poorly specified. As a result, models trained on such data frequently fail to meet the stakeholders' needs and do not generalise well to changes in the operational environment (Heyn et al., 2023). Second, the operational context is often underspecified, resulting in datasets, and consequently ML models, that perform poorly under real-world conditions (Heyn et al., 2022). These two limitations are typically mitigated by collecting large volumes of training data in the hope of covering all potentially relevant operational contexts and use case scenarios. Finally, the analysis and validation of the ML model require testable conditions based on stakeholder needs and requirements. However, to the best of our knowledge, no clear path exists from high-level stakeholder requirements for ML systems to testable conditions that can be used for purposes such as model validation and runtime monitoring. Establishing such a path would allow traceability of design decisions, such as the data collection, back to overarching system goals.

2.2. Causal inference

While humans often intuitively understand the direction of cause and effect (a drop in the temperature measurement does not cause the sun to set, even though both variables are associated in a dataset), today's ML approaches cannot infer causal structures from observational data alone (Pearl, 2019). There are two reasons for this limitation suggested in the literature:

1. **Lack of observability**: In conventional ML we only have a limited sample set available to infer properties of an underlying function, i.e., "we want to estimate a property of an object we cannot [entirely] observe" (Peters et al., 2017). This first lack of observability is typically met by "throwing more data at the problem"[1].
2. **ML models represent associations**: Even if one were able to perfectly observe and reconstruct the underlying function, the trained ML model is still only a probabilistic representation of the underlying problem, i.e., it represents associations but not causal relationships between variables. This is not enough to infer a suitable causal model for the desired operational context, because even a perfectly learnt probabilistic model can relate to any one of several possible causal models that are compatible with the data (Pearce and Lawlor, 2016).

Without recognising the causal structure of a problem, the incorporated ML model may learn a probabilistic representation that seems compatible in a training context, but as soon as it is deployed in a slightly different environment, its performance may deviate drastically from the expectations (D'Amour et al., 2022). It is therefore necessary to find a path from the expected cause-effect relationships to the necessary data. Two approaches in the realm of *causal learning* seem possible to solve the outlined problems of underspecification of data for ML:

One approach is the development of methods that allow for *causal discovery* from observational data. While data-driven causal discovery made significant progress in algorithms in recent years, see for example the review by Vowels et al. (2022), these algorithms still rely on "strong and often untestable assumptions". Causal discovery therefore usually only allows the identification of a so-called *Markov equivalence class* for a causal graph. This means that several distinct causal graphs can be equally compatible with the data, making it impossible to uniquely identify the underlying causal structure without additional knowledge.

A second approach is *causal inference* based on the explicit inclusion of human "insight" by defining, even partially, the expected causal model of the operational context. This prior knowledge allows to reason about data, assumptions, and tests that are needed to arrive at a probabilistic model that correctly represents the environment (Hernán et al., 2019).

Directed acyclic graphs (DAGs). DAGs provide an accessible visualisation of prior knowledge about causality, with nodes representing the variables of a system of interest and directed edges representing the direction of cause-and-effect (Elwert, 2013). DAGs are a qualitative graphical representation of causal models. They provide information about the direction of cause-and-effects between variables, but they do not provide information about the strengths or functional properties of the causal relations (Pearce and Lawlor, 2016). Mathematically, the graph structure of a DAG represents a structural causal model (SCM). A SCM is a formal assignment of random variables $X_1, \ldots, X_n$ through a set of functions f_i

[1] https://www.forbes.com/sites/kalevleetaru/2019/07/07/automatic-image-captioning-and-why-not-every-ai-problem-can-be-solved\-through-more-data, accessed 2024-10-17

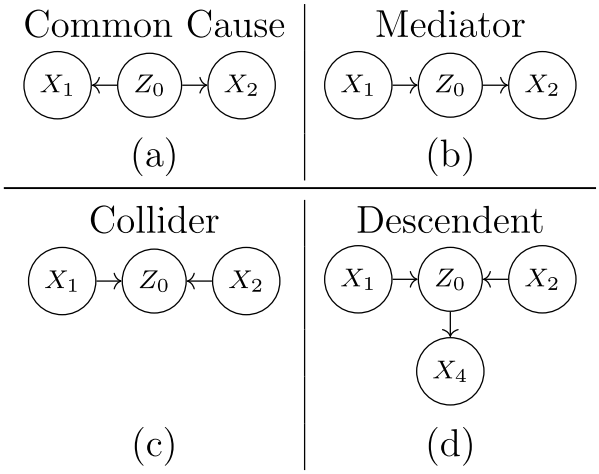


Fig. 1. Elemental structures in a causal model.

and unexplained, jointly independent variables U_i (Schölkopf, 2022):

$$X_i := f_i(\mathbf{PA}_i, U_i) \quad (i = 1, \ldots, n). \tag{1}$$

In a SCM, a variable X_i is a *direct cause* of another variable X_j, $i \neq j$ if X_j appears in the function f_i that assigns the value of X_i (Glymour et al., 2016). If each variable is modelled as a node, and the *direct causes* as directed edges between the nodes in a DAG, then the *parents* in the graph, denoted as $\mathbf{PA}_i$, of a node X_i are all nodes that have a directed edge pointing into the node X_i. DAGs are directed, i.e., the arrows can only have a single head pointing from the cause towards the effect (Rohrer, 2018). They are acyclic because a variable at a given point in time cannot be the cause of itself (Pearl et al., 2016). In DAGs, four fundamental structural patterns can occur which are illustrated in Fig. 1.

Confounding. DAGs can clarify assumptions about *confounders*. A typical case of confounding occurs when a variable Z_0 acts as common cause of two unrelated variables X_1 and X_2 such that a *spurious relationship* between X_1 and X_2 can be observed: $X_1 \leftarrow Z_0 \rightarrow X_2$ (Fig. 1 (a)). Such associations arise through *confounding paths*, i.e., non-causal paths that induce statistical dependence between variables. In this example, an association between X_1 and X_2 is observable because there exists a path $X_1 - Z_0 - X_2$ when ignoring the direction of the arrows in the DAG. Association can "flow" along such paths against the direction of causality. Conditioning on the common cause Z_0 blocks this flow and removes the spurious association between X_1 and X_2.

Confounding can also arise for other reasons besides a common cause: A *collider* blocks the flow of association unless conditioned upon. That is, if in a collider structure, as shown in Fig. 1 (c), a learning algorithm conditions the data on Z_0, it will *open* the flow of association between X_1 and X_2, which results in a spurious association and confounding. This situation is often referred to as *selection bias*. Similarly, conditioning on a *descendent* of a collider, as illustrated in Fig. 1 (d), has the same effect. Here, conditioning on X_4 will re-open the flow of association between X_1 and X_2 because X_4 is a descendent of the collider Z_0. These cases show how any decision on which variables should be included or excluded in a statistical analysis, or as data for a learning algorithm, can either block or open non-causal paths.

d-separation. D-separation is a formal criterion, introduced by Pearl (2009), that allows to reason systematically about conditional independence in DAGs. It determines whether there exists an "open" path of association between two variables, i.e., a path on which association can flow. A *confounding path* of association exists when association can flow against the assumed direction of cause-and-effect in a DAG. Pearl et al. (2016) states that such a confounding path p can be blocked by a set of nodes Z if (i) p contains a chain $X_1 \rightarrow Z_0 \rightarrow X_2$ or a common cause $X_1 \leftarrow Z_0 \rightarrow X_2$ such that the middle node Z_0 is in Z, or (ii) p contains a collider $X_1 \rightarrow Z_0 \leftarrow X_2$ such that neither Z_0 nor any of its descendants are in Z. The rules of d-separation apply to all possible paths in a DAG. In practice, tools such as `DAGitty` (Textor et al., 2011) can determine which variables need to be conditioned on to block any confounding path implied by a DAG elicited from prior knowledge.

3. Related work

The aim of RE is to identify system goals and elicit requirements for software development. Recognising causality is important in specifying the desired behaviour of a software system. For example, BDD allows to identify system goals and to elicit requirements in software development by specifying the desired behaviour of a system using natural language in a "Given-When-Then" structure (Binamungu et al., 2018). This structure provides high level domain-specific scenarios that guide both requirements specification and testing. An RE-governed development process typically entails the following steps:

(a) describing textually examples of the desired functionality,
(b) developing of system prototypes that exhibit the desired functionality,
(c) testing and deploying the system based on test cases derived from the described examples of the desired functionality (Smart, 2014).

Although RE approaches, such as BDD, have been applied successfully to conventional software development, their usefulness for ML systems can be limited by gaps in the functional specification (Irshad et al., 2021) and a lack of understanding of the operational context (Heyn et al., 2022). The context in which an ML model operates is often only implicitly assumed during design, for example in the choice of data used for training (Mitchell et al., 2021). While natural language requirements offer certain advantages such as ease of use, accessibility, and flexibility compared to domain-specific language, they are also prone to ambiguity which can make specifications difficult to interpret (Gervasi and Zowghi, 2005; Binamungu et al., 2018). To our knowledge, usual approaches to RE lack a clear link between the specified desired behaviour and the data requirements for training and validating of ML models; they do not provide methods to guide the specification of the necessary data required to train the desired behaviour into an ML model. On the other hand, RE methods such as BDD inherently imply a causal structure: Given (*a context*)→When(*a cause*)→Then(*an effect*). Combining the naturalness of the "Given-When-Then" structure and the inability of ML to infer causal relationships from data, the idea is therefore to make implied causal relationships explicit through graphical models as part of the RE effort for software systems that use ML.

An example of an RE method that graphically defines causal relationships for software is goal-oriented requirement engineering (GORE) (van Lamsweerde, 2001). Goal models, represented through directed graphs, allow for the decomposition of high level goals to system level design decisions (Anwer and Ikram, 2006). However, with GORE it is not immediately obvious how to define clear and measurable goals in terms of the data required to achieve a desired behaviour in an ML-enabled system, leading to "uncertainty and unpredictability of [the] implementation" (Ishikawa and Matsuno, 2020). Goal models are a type of knowledge graph that generally represent knowledge models using a structure of nodes as entities and edges describing relationships between these entities. Ehrlinger and Wöß (2016) highlighted that a knowledge graph can be seen as a model that "acquires and integrates information into an ontology and applies a reasoner to derive new knowledge". However, unlike graphical causal models in the form of DAGs, which base on a mathematical framework for causality, a formal definition of knowledge graphs is missing (Buchgeher et al., 2021). Although there is significant prior work on knowledge inclusion into ML model development, see for example the literature review by Von Rueden et al. (2021), to our knowledge there has been no attempt to incorporate prior knowledge about causal relationships in the ML development pipeline. Other

RE methods, such as use case modelling or user stories, do not explicitly define causal relationships in a way that it could be applied to finding data for training an ML model as part of an ML development pipeline. The potential of including causal knowledge as part of RE has not yet been fully leveraged (Fischbach et al., 2020). In a recent systematic review, Giamattei et al. conclude that causal reasoning is increasingly recognised as a tool for software quality assurance, particularly in fault localisation and testing activities (Giamattei et al., 2024). The authors highlight that causal reasoning remains underutilised in earlier phases of the software lifecycle, such as RE. In a vision paper, we argued therefore that causal modelling is promising for RE for ML systems Heyn et al. (2025). We suggested that more research is needed to explore how causal modelling can support key RE tasks, including requirements elicitation, particularly in the context of complex, data-dependent characteristics of ML-based systems.

3.1. Research objectives and research questions

The scope of this article is to develop this vision of using causal models in RE activities further by proposing, arguing, and demonstrating that causal modelling can be used as requirements specifications in practice when building software systems with ML components. Specifically, we show how to discover iteratively data needs for the training and validation of ML models and demonstrate the approach on an industrial use case of a CPS for anomaly detection. The use case is relevant to industrial practitioners because building representative datasets for ML models in industrial settings for anomaly detection is challenging (Zheng et al., 2019):

1. **Data diversity:** Industrial sites can generate a wide range of data, but the data relevant to anomaly detection are very limited as abnormal situation happens rarely in productive system (e.g., once in ten years of continues operation).
2. **Cost:** The creation of data for rare and abnormal situations in real industry environments is often expensive and risky.

As a result, test benches and prototypes are used to simulate operational environments and collect data. To the best of our knowledge, there is yet no explicit treatment of causal relationships as part of requirement specifications for software that includes ML components. This research therefore has two main objectives:

Obj1 The first research objective is to define a procedure for causal analysis as a complement to conventional requirements elicitation for software systems with ML models.

Obj2 The second research objective is to explore in a case study whether the use of causal modelling during prototyping of a CPS has a positive impact on the robustness and performance of the ML components that are part of a CPS for anomaly detection.

Our hypothesis is that causal modelling can help to isolate and to specify clearly the intended causal behaviour of the ML model and the operational context in which the system operates. As a result, causal models support the specification of data needs for training the model in the given operational context. The following research questions (RQs) guided this study:

RQ1 How can causal models provide guidance for the development of software systems with ML components?

RQ2 In a case study, to what extent can the performance and robustness of the software system with ML for cyber-physical systems be improved by incorporating domain experts knowledge as graphical causal models during the system development?

4. Methods

We applied engineering research to explore, conceptualise, and evaluate CDD as a new and complementary approach for the elicitation of requirements for software with ML components. The research entailed three research cycles, whereof the first cycle focused on exploring causal models for capturing prior knowledge about the system and its operational context and the second cycle focused on conceptualising CDD for software with ML components. These two cycles of exploring prior knowledge and conceptualising CDD primarily addressed RQ1 on how causal models can guide development. The final third cycle focused on demonstrating the approach in an evaluative case study and addressed RQ2 by testing the extent to which causal modelling improves robustness and performance of ML.

In this section, we will first describe the study site and the industrial use case that became part of the case study. Then, we will describe the methodology for each of the three research cycles.

High-level requirements for arc detection system
RA1: *GIVEN* a low voltage DC system in normal operation *WHEN* an arc occurs *THEN* an alarm should be triggered.
RA2: *GIVEN* a low voltage DC system in normal operation *WHEN* **no** arc occurs *THEN* an alarm should **not** be triggered.

4.1. Description of study site

The research was conducted within the *Very efficient deep learning in the IoT* (VEDLIoT) EU Horizon 2020 project. The aim of the project was to develop tools, methodologies, and experience for supporting the development and deployment of AI in IoT systems (Kaiser et al., 2022). We chose Siemens, one of the industrial partners in VEDLIoT, as the case company because their use case, which involves anomaly detection in CPS, represented a typical industrial product development in its prototyping phase. It is at this prototyping stage that practitioners need guidance in understanding the CPS and in discovering and defining the necessary data in the context of the design accordingly (Diefenbach et al., 2019).

4.2. Description of use case

The use case on which we explored (first cycle), conceptualised (second cycle), and demonstrated (third cycle) CDD for software with ML components is a system for the detection of series arc faults in low voltage direct current (DC) distribution systems. The use case represents a CPS for anomaly detection using ML. Unlike parallel arc faults, which are mainly detected by mandatory over-current protection systems, serial arc faults are more challenging to detect with conventional systems (Lu et al., 2018). The company engineers provided the following high-level requirements for the use case system:

Safety is a critical aspect of arc detection and the false prediction rate therefore should be low. If an arc occurrence is missed, i.e., a false negative, and the power is not cut, it can lead to severe economical damage due to fire and injuries to personal. On the other hand, a false positive, i.e., a false alarm, can lead to economical damage for customers due to unwanted power cuts. Therefore, RA1 requires a low false negative rate, and RA2 specifically requires the prevention of false positive classification. Today's approaches rely on physical model analysis and statistical algorithms for anomaly detection (Xiong et al., 2017; Chae et al., 2016). The limitation of conventional methods is their low adaptability to different system structures and application scenarios. This makes their conversion to industrial products, where different scenarios need to be handled, challenging (Lu et al., 2021). In recent years, there has been a number of studies investigating the possibility and performance of various ML algorithms in DC series fault detection (Lu et al., 2020). However, ML approaches often neglect the underlying physical properties of the engineering problem which can lead to systems with unexpected behaviour (Frisch, 2014).

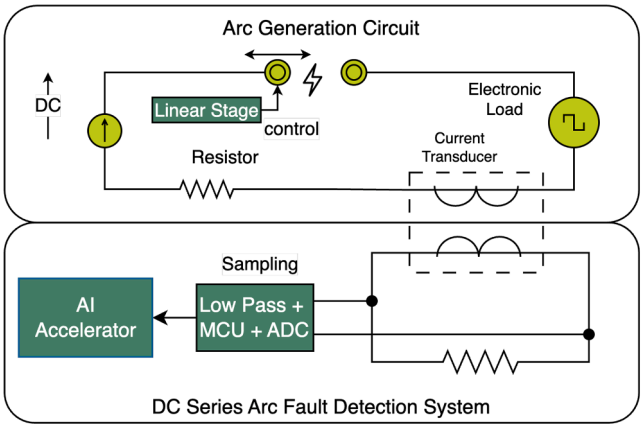


Fig. 2. Component diagram of the arc fault detection prototype system.

Description of prototyping system. The system in this use case is an industrial prototype for a DC switch gear arc detection system. The prototype followed a reference implementation for DC series arc fault detection systems which increased the realism of the case study (UL, 2018). An overview of the prototype system is depicted in Fig. 2.

The prototype consisted of an arc generation circuit and an arc detection system. The main circuit consisted of a DC power supply, an electronic load, a passive load, and a pair of electrodes that were connected during normal operation. One of the electrodes was controlled by a linear stage, allowing it to create an air gap to generate the arc. The current transducer of the detection system reduced the current proportionally and the signal was then sampled by an analogue-digital-converter (ADC). At runtime, the data was then further transmitted to an edge device for processing with the trained ML algorithms. A component list of the prototype system is provided in Appendix A.

4.3. First cycle: exploring prior knowledge integration through causal models

In the first cycle of this engineering research we explored how prior domain knowledge about the system under development can be captured in graphical causal models. We conducted three workshops with Siemens in October and November 2022. These were conducted remotely using Miro as an interactive whiteboard platform and the R-package *DAGitty* to interactively construct graphical causal models (Textor et al., 2011). Two company development engineers, a project manager, and two academic scientists participated in each workshop. The first author of this paper prepared a Miro board with an explanation of what causal models are and examples of graphical causal models illustrating causal relationships such as *rain → wet road → braking distance*. The company participants explained the use case in the form of scenario descriptions and provided data samples from initial experiments. Together, the scientists and engineers identified causal mechanisms, i.e., independent "logical" cause-and-effect relationships based on the use case scenario and the initial data. The result was a first causal model that contained the envisioned cause-effect relationships for the use case which were supported by the initial data samples. In the subsequent two workshops, the causal model was continuously extended and modified by incorporating new experimental data and by discussing underlying assumptions of the cause-effect mechanisms, contextual assumptions and possible interventions in the use case. This iteratively "drawing" of a graphical causal model helped the company experts to formalise their prior knowledge about the system. As a result of the workshops, the researchers and engineers started to see patterns in their work progress that lead to a systematic approach in how they refined the causal models for the use cases based on prior expert knowledge and experimental results.

4.4. Second cycle: conceptualising the CDD approach for software with ML components

Based on the progress and the discussions during the first cycle workshops, the scientists developed a heuristic for constructing graphical causal models using experts' prior knowledge. They also outlined a workflow for documenting assumptions about cause-effect relationships through causal models and for deriving both data and ML-model requirements from these models. In the final two workshops, the proposed heuristic and workflow were discussed and refined. The final causal model for the industrial use case was created, and data as well as ML model requirements were defined based on the CDD approach.

4.5. Third cycle: demonstrating CDD in practice on an evaluative case study

The final cycle of this engineering research focused on evaluating the proposed CDD approach in terms of ML model performance and robustness. The evaluation tests were run on a prototype of the arc fault detection system. The current signal served as data source for arc classification because it is a common feature used in studies on DC series arc fault detection (Lu et al., 2018). There are several reasons for this. The installation of a current sensor is more realistic in practice compared to voltage monitoring because the installation of current transducers does not require any additional intervention in the circuit. The current sensing is less susceptible to environmental disturbances than other sensors such as temperature, optical or acoustic sensors. Finally, a change in current measured at one point in the circuit can reflect the condition of the whole circuit which allows the number of sensors to be reduced

Two independent variables were manipulated during arc generation and data collection: the load profile and the electrode movement pattern. The load profile could be simulated with different settings of the electronic load. The electrode movement pattern could differ not only in the arc gap and arc duration, but also in its speed of movement and its behaviour after the arc had occurred.

5. Causality-driven development (CDD)

The idea of causality-driven development for ML is to add a step in the ML development pipeline that captures and documents systematically prior knowledge about the expected causal structure of the system. In particular, the additional step documents causal relationships that cannot be learned from data with conventional ML models today (Pearl, 2019). Domain experts must provide this additional data knowledge, but they are often neither ML model developers nor data scientists. Therefore, a common language is needed to communicate causal knowledge. Causal models in their graphical form as directed acyclic graphs can be used as a universal language between domain experts and developers because graphical causal models are well described and commonly applied in statistics to identify and to describe causal relationships (Vowels et al., 2022; Shpitser et al., 2010). Fig. 3 illustrates where in the ML development pipeline we envision to include prior knowledge in the form of causal models.

The causal relation network of complex systems however can be huge and inscrutable. It might not be feasible to immediately capture every possible cause-effect relation for a typical industrial CPS use case. We therefore propose an iterative approach including four steps to building causal models for ML systems. The model grows iteratively until the domain experts are satisfied that all relevant cause-effect relationships are included and the ML system behaves robustly enough in the intended operational context. If testing shows that the ML model is still not robust enough, or if new operational context assumptions are to be included, the causal model can be extended, new data generated according to the extended model, and the tests can be repeated. The iteration and the resulting data collection are guided by a causal model which allows

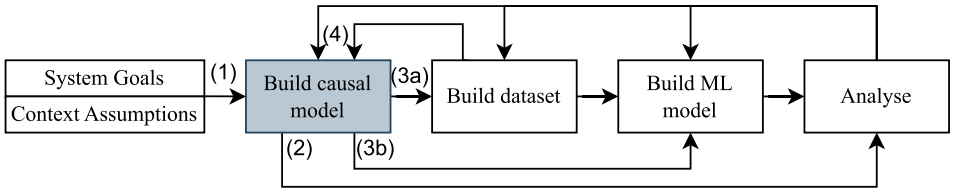


Fig. 3. Inclusion of causal models in the ML development pipeline. (1): Prior Knowledge (2): Context description and testable conditions (3a): Data requirements (3b): Model requirements (4): Data availability / observability.

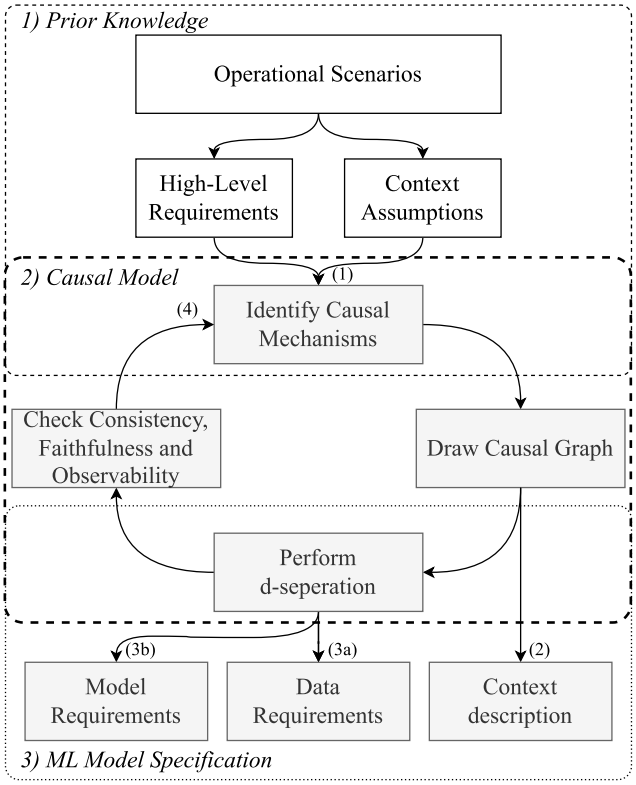


Fig. 4. Workflow for constructing causal models to support requirements specification in ML-based software. The numbers in parentheses refer to the stages depicted in Fig. 3.

for traceability of the design decisions and predetermination of data collection.

Fig. 4 details the building of causal models as part of an ML model pipeline and below we will explain each step in detail. We arrived gradually at the proposed workflow by working with our project partner in regular workshops.

Step 1: Identifying causal mechanisms from high level requirements and context assumptions. In the first step, causal mechanisms are identified that subsume the described functional behaviour using prior knowledge the developers or domain experts have about the system. Scenarios, elicited by the stakeholders, for example through user stories or BDD, contain information about high level requirements and context assumptions. Causal mechanisms that govern the behaviour described in these high level requirements need to be identified. This additional knowledge about causal mechanisms can be obtained through a set of different methods summarised for example in Molak and Jaokar (2023): In controlled experiments, researchers can "isolate" the object in a context that allows data collection in a controlled environment. In randomised controlled trials (RCTs), this "controlled environment" is created by randomly assigning an external factor to the objects of interest. That allows

Principle steps for causal-driven development (CDD) in Fig. 4.

Step 1, Identifying causal mechanisms: In this step, the expected outcome, relevant causes in the operational context, and other effects influencing the desired outcome are identified, for example through expert knowledge. This leads to a set of separate cause-effect relationships.

Example: Assume we design a system to warn if sensor noise affects the safety of an automated vehicle unacceptably much. Experts identify that Rain *affects* Sensor Noise*, which in turn affects* Safe Driving*. They identify also a second mechanism, namely that* Road Quality *affects* Braking Distance*, which in turns also affects* Safe Driving.

Step 2, Drawing a graphical causal model: The identified cause-effect relationships are connected to a directed acyclic graph (DAG). Steps 1 and 2 occur iteratively, i.e., each newly discovered cause-effect relationship from Step 1 is integrated in the causal model. The resulting graph documents the operational context and represents the expected cause-effect relationships for the system.

Example: The two identified causal mechanisms are added to one DAG. The expert noted that Rain *also affects the* Braking Distance*, which was previously not recognised. This relation is added to the DAG (here as dashed edge).*

Step 3, Performing d-separation: Based on the causal model, confounding paths and colliders are identified. The principle of d-separation is applied to determine which variables need to be conditioned on in order to block "non-causal" paths of association in the data. These "non-causal" paths can lead to undesired behaviour of a ML component by introducing spurious associations. The resulting set of variables that must be controlled provide requirements for the training dataset, the data that must be observable at runtime, and requirements towards the ML models in terms of data it must accept as well as model configurations. This step helps prevent the model from becoming biased due to the learning of spurious associations or inadvertently opening collider paths.

Example: To estimate the direct effect of *Sensor Noise* on *Safe Driving*, the expert identify a non-causal path: *Sensor Noise* ← *Rain* → *Braking Distance* → *Safe Driving*. The principle of d-separation tells us now that we must condition on *Rain* to block the spurious path. This gives a data requirement: The variable *Rain* must be available, e.g., through a rain sensor, and the training data must entail different "rain conditions" such that an ML model can learn the influence of rain on the entire system.

Step 4, Checking consistency, faithfulness, and observability: The resulting graphical causal model must not contain cyclic dependencies to be consistent. Furthermore, the graphical model must be faithful to the data: it should not imply associations that are absent in the data. Finally, the variables required for closing non-causal paths must be observable. If they are not observable, proxy variables can be identified and added to the causal model.

Example: The variable *Sensor Noise* is not directly observable. Instead, the expert identify that *Lidar Variance* can act as proxy for *Sensor Noise*.

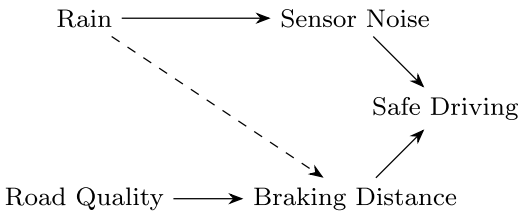


Fig. 5. Example of a DAG as part of CDD.

in most, but not all cases, see for example Kostis and Dobrzynski (2020), the attribution of observed differences in the outcome to the assigned external factor. However in many cases, such controlled experiments and RCTs are either prohibitively expensive (e.g., due to the amount of experiments necessary to discover every relevant causal relationship) or ethically impossible to conduct. Therefore, an alternative, or supplement, to controlled experiments is knowledge from domain experts. We assume that domain experts have a deep understanding of the area of expertise by carefully studying and collecting experience. However, using expert knowledge has limitations and risk as listed in Molak and Jaokar (2023) and Bradley et al. (2006): Experts might be overconfident, especially the more experience they have gained in a given field. Experts might also suffer from what Kahneman describes as "availability heuristic": Plausible solutions (here: causal relationships) that come to the expert's mind first have higher influence on the final decision (Daniel, 2017). The combination of personal experience and domain experience of a domain experts is still valuable input if two aspects are considered to mitigate the previously mentioned risks: First, a group of experts should work together to identify relevant causal relationships. Second, causal relationships relevant for the system under development must be documented to communicate assumptions made by the experts.

5.1. Step 2: Drawing of graphical causal model to document context assumptions from a set of causal mechanisms

In this step the identified causal mechanisms are merged into a DAG. A DAG, together with the assumption of independent noises,[2] for example sensor noises, can be used to decompose a joint probability function entailed in (1) into a causal factorisation (Schölkopf et al., 2021):

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p\left(X_i | \mathbf{PA}_i\right) \tag{2}$$

While a full structural causal model (SCM), as described in (1), of a real-world system is usually not attainable, the principle that causal dependencies can be factorised into local mechanisms, as suggested by (2), justifies the iterative construction of causal graphs. Each newly elicited causal mechanism can be added as an additional factor which allows the model to grow incrementally without requiring a full SCM. DAGs representing the causal structure of the system become therefore part of an ML model specification and can provide evidence about the assumed causal relationships in the operational environment of the ML model.

5.2. Step 3: Performing d-separation to derive data requirements and model requirements.

Through modelling expected cause-effect relations, causal models provide an insight into possible sources of confounding in the data. If confounding is not treated properly, the dataset presented to the ML model for training contains spurious correlations and the resulting ML model's behaviour will be biased. The aim of *d-separation* is to identify possible spurious correlation and thereby to avoid confounding or anti-causal behaviour of the final ML model by allowing the model to condition on possible confounders or by forcing the model not to condition on possible colliders in the causal model. D-separation does not rely on manual effort by experts alone because tools, such as `DAGitty` (Textor et al., 2011), can be used to automatically perform d-separation on a given DAG.

Data requirements from confounding. Assume a dataset containing three variables X_1, X_2, and Z_0. Assume further a causal structure in which Z_0 is a common cause of X_1 and X_2 (see Fig. 1 left top case). If one were to construct the training dataset to only contain the variables X_1 and X_2 in this case, the ML model would learn an association between X_1 and X_2 indicating a causal-effect relationship between the variables which in fact does not exist. The result is a spurious correlation between X_1 and X_2 through a common cause Z_0. Therefore, a data requirement resulting from the causal model is that the dataset shall explicitly contain the variable Z_0 to avoid learning a spurious correlation.

Continuation of the example in Fig. 5: The variable Rain *acts as a common cause for* Sensor Noise *and* Braking Distance. *It was therefore necessary in the example to include* Rain *in the training data to allow the ML model to learn the confounding influence of* Rain.

Data requirements from colliders. Another case arises when Z_0 acts as a collider between X_1 and X_2 (see Fig. 1 (c)). Here, if only X_1 or X_2 are presented in the training dataset, the ML model would not learn an association between the two variables, which is correct. However, if one were condition on Z_0 (i.e., using it as a feature), the association path between X_1 and X_2 could be opened because the ML-model could include Z_0 as predictor which leads to learning an association between X_1 and X_2 that in reality does not exist. The problem is that many ML-models, and especially deep learning models, are black-box models where we cannot actively choose which variables in the data are used as predictors. However, the ML-model cannot use Z_0 as predictor if the data for this variable is not present in the dataset.

Continuation of the example in Fig. 5: Assume X_1 = Rain, X_2 = Road Quality *and* Z_0 = Braking Distance. Braking Distance *is acting as collider between* Rain *and* Road quality. *If* Braking Distance *is included in the training data, an ML model might learn that* Rain *and* Road Quality *are negatively associated, i.e., rain improves the road quality*[3]*, and vice versa. In summary, the model may falsely infer that* Rain *and* Road Quality *are statistically dependent, even though they are causally independent, leading to biased predictions or incorrect attributions.*

Handling divergences in experts' knowledge. Ideally, the group of experts finds a consensus on the causal mechanisms that should be merged into one DAG. However, we acknowledge that such consensus among experts might not always be reached. In such cases, it is possible to create a number of different DAGs. Then, using d-separations, *independence conditions* can be found for each alternative DAG. Given the data of the use case, statistical tests of conditional independence can be performed to examine which of the alternative DAGs is consistent with the observed data (see checking for faithfulness in Section 5.3). This approach allows experts to converge by ruling out models that contradict empirical evidence. Documenting divergences as alternative DAGs provides traceability of assumptions and enables later reconciliation in case that more data or operational experience becomes available. Similarly, experts might agree on a *partial* causal model and apply data-driven causal discovery to automatically suggest missing elements in the model based on available empirical data. We discuss the potential use of causal discover in more detail in Section 7.4.

[2] If two noise terms were not independent, there must be some *common cause* to which an independent noise then applies (see Reichenbach's *common cause principle* (Reichenbach, 1956).

[3] Intuition: If one already knows the braking distance was long (conditioning on braking distance), and it is raining, it becomes less likely the road was bad because the rain alone could explain the long braking distance.

Updating causal models for new use cases. In some cases, such as the occurrence of new confounding environmental factors, the dataset needs to be adjusted by adding additional variables to avoid possible spurious correlation in the trained model. In other cases, in which the new environmental factor acts as collider, such an adjustment could however cause spurious correlation in the trained model which ultimately can lead to bias in the ML model's behaviour. The causal model guides developers towards which data variables are needed as input to the ML model (i.e., do we only need X_1 as input variable, or should we also design the system such that it can perceive other variables as input at runtime in order to mitigate confounding?) by providing information about the expected cause-effect relationships in the operational context of the ML model. It is important to communicate these causal assumptions about the operational context in case an ML model should be deployed into a new operational context. Here, causal models, as part of the requirements specification for ML models, provide this information about the assumed causal operational context. This information can then be used to decide if the original causal assumptions are still valid in the new operational context or if the causal model needs to be updated. Such an update can require new training data as new data requirements might arise, and consequently a re-training of the ML model to make it compatible with the new causal operational context.

5.3. Step 4: Checking consistency, faithfulness, and observability

Checking for consistency. Consistency refers to the direction of the edges in the graph are consistent with the causal relationship they represent based on the prior knowledge of the stakeholders. Edges should only point from causes to their effects, and not vice versa. Furthermore, there should not be any "causal-loop", i.e., a cycle or feedback loop between causes and effects because it would otherwise imply a self-reinforcing causal relationship. For causal analysis, acyclic graphs are used because "a variable cannot cause itself, either directly or through another variable", i.e., there cannot be cycles of cause and effect (Hernán and Robins, 2020). Time can be discretised to represent a feedback loop for example in a control system with a feedback loop from observation to a control unit that regulates an input to the system. This also reflects the usual assumption of discrete time steps in digital control systems, i.e., the treatment A (controller output) and the covariates L (system under control) change at discrete time steps $[n, n+1)$. The result is $A_{n-1} \rightarrow L_n \rightarrow A_n \rightarrow L_{n+1}$ (Hernán and Robins, 2020, Fine Point 20.1). In other words, the state of the system at time $n-1$ affects the controller at the next time step n, which in turn affects the system at that time step n, and so on.

Checking for faithfulness. Additionally, the *faithfulness* of the causal model can be checked. A causal model violates *faithfulness* when it suggests an association between variables that does not appear in the empirical data (Hernán and Robins, 2020). In other words, if the causal model suggests a causal relation, we would expect to observe corresponding association between the variables in the data. There are, however, rare cases in which causal effects can "cancel each other out," resulting in a null association even though a cause-effect relationship does, in fact, exist. These cases are rare because for two causes to have exactly the same effect and perfectly cancel each other out is highly unlikely (Spirtes et al., 2000, pp. 68–69).

Checking for observability. Another aspect that needs to be checked is *observability*. In many cases, variables in the causal model are not observable, such as the measurement noise of a sensor. In these cases, we either have to find alternative variables that allow for the identification of the missing system aspects, such as moderating variables, or the data can be provided through a simulation.

6. Results

First, this section presents the prior knowledge identified by the company expert with the help of causal modelling during the explorative first cycle of this engineering research. Then, a heuristic is presented for drawing these graphical causal models based on the conceptualisation of the CDD approach conducted in the second cycle of this research. The heuristic is described using the use case as a running example to conceptualise how high-level requirements, context assumptions, and prior knowledge of the company experts were incorporated into a graphical causal model. The resulting heuristic for eliciting graphical causal models and deriving requirements addresses RQ1. Finally, the results of the evaluative case study, which was the focus of the third cycle of this engineering research, are presented. The evaluation includes a comparison of the performance of a deep learning model trained on a previously existing dataset for the use case with the performance of a model trained on a dataset following the requirements derived from the causal model.

The resulting heuristic for eliciting graphical causal models and deriving requirements addresses RQ1. The evaluative case study comparing the basic and CDD-based ML models provides evidence for RQ2.

6.1. Prior knowledge about the system

During the explorative first cycle, the company experts iteratively drew a graphical causal model of the system. This process not only produced a graphical representation of the assumed cause-effect relationships of the system, but also helped company experts to "explore" and formalise their prior domain knowledge about the system.

In the end, the following prior knowledge (PK) was identified by the company experts:

PK1: The arc shows different behaviours under various load profiles, see for example studies such as (Dang et al., 2021; Chae et al., 2016) describing different arc generation setups and representative current signals during arc occurrence.

PK2: There is a relation between current, system power supply, which is assumed to be stable, and the load profile.

PK3: High frequency disturbances from unmodelled sources (active switching components such as inverter for example) can trigger nuisance tripping (Dang et al., 2021).

PK4: The current signal during arc occurrence has a distinct waveform in the frequency spectrum, which depends strongly on the system's dynamic behaviour. This feature has been used in related works for arc fault detection (Lu et al., 2018).

PK5: A jump in current can trigger nuisance tripping. Load changes caused by connecting or disconnecting of devices, which is predictable but unmodelled, can cause such current jumps (Chae et al., 2016).

PK6: The sensor signal can be noisy under operational environment.

6.2. A heuristic for the elicitation of a graphical causal model from prior knowledge and context assumptions

The goal of causal modelling is not to obtain estimates of a parameter of interest, but to clearly define the estimand in its relevant context that is estimated using ML or any other statistical method as an estimator. An estimand is the "target parameter" and represents a certain quantity of interest about which we want to draw conclusions based on causes that affect that parameter. Fig. 6 provides a running example based on the industrial use case on arc fault detection in low voltage DC system, outlining each step of eliciting a graphical causal model for our industrial use case. The variable names and definitions used in the causal models are listed in Table 1.

The nodes of the graphical causal models represent causes and effects, while edges represent the direction of cause and effect. The nodes

Table 1
Variables for arc fault detection use case. (meas.: measured; freq.: frequency).

Variable	Definition
Arc	Arc occurrence
Load	Load profile
I (I_s)	Current (measured)
HF_I	High frequency components extracted from the load profile, including HF from arc occurrence and HF noise.
$\|S(f)\|$	Frequency spectrum
U_I	Sensor noise

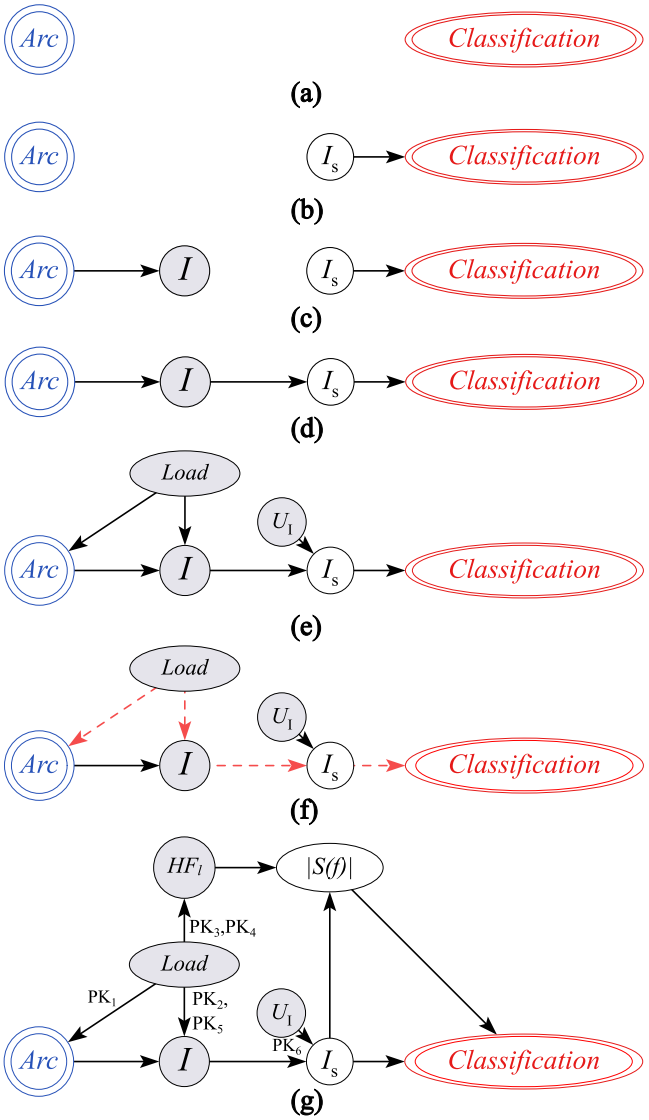


Fig. 6. Example of performing a causal modelling task: (a) Definition of relevant variables of outcome and causes of interest; (b) Identification of other causes affecting the outcome of interest; (c) Identification of other effects that are governed by the identified causes; (d) Establishing relations between causes and effects; (e) Identifying competing causes; (f) Analysing for confounding of the outcome through competing causes; (g) Closing confounding paths through additional observable variables. All variables in this diagram are defined in Table 1.

for the outcome and cause of interest are highlighted as a concentric ellipse and a circle. Grey nodes represent unobserved variables at runtime and white nodes represent observable variables at runtime.

Defining outcome and causes of interest. Fig. 6 (a) illustrates the initial step in constructing a graphical causal model, which involves defining the desired outcome and the cause of interest. The desired outcome is a classification result that is true, if an arc is present, and false, if an arc is not present (RA1 and RA2[3]). The (root) cause of interest that changes the classification result is the presence of an arc in the system. As we cannot measure the arc directly, we need another cause that acts as proxy. This proxy is the measured current, I_s in (b). We set the output of a current measure I_s as an input (direct cause) for the classification.

Defining other causes affecting the outcome and relations between causes and effects. In (c), we add that an arc has an effect on the current I of the system (PK1[4]). We also know that the measured current I_s must be directly influenced by the system current I, which is why we can draw a directed edge between these two variables in (d).

Including competing causes. The measured current I_s, and the current of the system I might not be identical, because I_s is influenced by measurement noise U_I (PK6) and I can be influenced by other confounding factors. In the next step (e), such influencing external causes are added to the causal model. Besides the measurement noise U_I that influences I_s, the company experts identified the *Load* connected to the DC grid as influencing the current I (PK5). The load however also influences the characteristics and probability of an arc (PK1). It is therefore a common cause for both I and *Arc* and a directed edge is not only drawn from *Load* to I, but also to *Arc*. The causal models illustrated in Fig. 6 (a) - (e), are the result of applying the first two steps of CDD that were described in Section 5 and illustrated in Fig. 4. The resulting graphical causal model provides a visual context description of the ML problem.

Identifying and closing confounding paths. The next step in CDD involves applying *d-separation* because with *Load*, there is a confounding path in our system that can lead to spurious correlation. Certain external load characteristics may influence the current I in a way that the measured current I_s looks as if an arc occurs although that is not the case, or vice versa (PK3). This can lead to false positive and false negative classifications of arc occurrence. The confounding path is illustrated as red, dotted lines in Fig. 6 (f). D-separation tells us that information about the *Load* must be added to our estimand and consequently to the training dataset to close the confounding path. For the training data, the information about the connected load can be added to the dataset, which allows the ML model to identify the confounding factor. Unfortunately, we cannot measure the connected *Load* at runtime. However, the connected load causes a load-specific high frequency pattern HF_I that is distinct from the high frequency pattern of an arc occurrence (PK3). This high frequency characteristic can be extracted through a Fourier-transformation $|S(f)|$, based on the measured current signal I_s. By adding the load information in the training data and adding the high frequency information containing characteristics about the load to both the training and the runtime data, we allow the classification algorithm to learn about different load characteristics and thereby close the confounding path.

Final causal model and requirements for data and ML model. The final causal model in Fig. 6 (g) is consistent with the prior knowledge and context assumptions of the company experts, it is faithful to the training data, it does not contain any causal cycles, and we accounted for all non-observable variables by identifying proxy variables. The edges in

[4] The requirements (RA) and prior knowledge (PK) are listed in Section 4.2

Table 2
Requirements derived from causal model for arc fault detection.

ID	Requirement
A-D1	The current measurements should be disturbed by characteristic sensor noise.
Rationale:	*The causal model depicts U_I as influencing factor on the measured current I_s.*
A-D2	The current signal should be superimposed by additional load current (random and real scenario based).
Rationale:	*The dynamic of the load acts as confounder. Including data on load dynamics allows an ML model to learn the confounding factor.*
A-D3	The current signal should contain high frequency components from active switching components.
Rationale:	*At runtime, information about the dynamic load profile is not available. The high frequency spectrum can serve as proxy for obtaining information about the load.*
A-M1	The frequency spectrum $\|S(f)\|$ should be used as additional data input to the ML model.
Rationale:	*The causal model depicts $\|S(f)\|$ as additional input to the Classification model.*

the final causal model are annotated to establish a traceability between the causal assumptions and the prior domain knowledge (PK). The final estimand is a classification of arc occurrence under different load characteristics, given a measured current signal I_s and frequency spectrum information $|S(f)|$ at runtime. The resulting causal model provides a set of data requirements (A-D) and model requirements (A-M). The causal models indicates that the measured current I_s at runtime is superimposed by sensor noise which should be reflected in the training data (A-D1). Using the concept of d-separation, the identified confounding path is closed by adding the load profile in the training dataset (A-D2). Because we cannot measure the connected *Load* profile at runtime, we include the high frequency spectrum $|S(f)|$ of the current signal I_s as a proxy for the *Load* profile in both the training and runtime data (A-D3). This however requires our ML model to accept the additional signal $|S(f)|$ as input (A-M1). All requirements derived from the causal model are summarised in Table 2.

6.3. Experimental evaluation

An experiment was designed and conducted to demonstrate how the results of causal modelling improve model performance and robustness. Three requirements were selected: A-M1, A-D2, and A-D3. They were selected because the prototype system could be modified to accommodate the necessary data collection for these requirements. The evaluation therefore will investigate the answer to the following two questions:

- To what degree does including different electrical load profiles in training dataset, as suggested by A-D2, improve the robustness of the resulting ML model?
- To what degree does including features from the current's frequency domain as input to the ML model, as suggested by A-M1 and A-D3, improve the performance of the ML model?

The experiment was carried out in three steps: data generation, model training, and performance evaluation. In the first phase, the test bench was extended and new datasets were collected based on the CDD requirements A-D2 and A-D3. In the second phase, four models (two of which fulfilled the CDD requirement A-M1) were trained and validated to ensure overfitting did not occur. Finally, in the third phase, the performance of the four models, based on the different datasets as depicted in Fig. 7, were evaluated and compared.

Data generation. A prior dataset existed before the experiment that was created without the requirements elicited through CDD. This dataset is referred to as the *basic dataset* and the model trained with this dataset is referred to as the *basic model*. The basic dataset was collected from the test bench circuit with a resistor as passive load based on the industrial standard defined in UL1699B (UL, 2018). Fig. 8(a) shows a sample from the basic dataset.

In a first phase of the experiment, the existing test bench was extended with a dynamic DC electronic load in order to collect dynamic resistor load profiles as suggested in requirement A-D2. Fig. 8 shows data samples under different load profiles and electrode movement patterns.

Table 3
Circuit setups for datasets.

Item	Dataset		
	Basic	CDD-based	External
DC Power	50V - 80V	50V - 80V	50V - 80V
Passive Load	Resistor	Resistor	Resistor and Inductor
Electronic Load	None	Load pattern simulation under constant resistor mode	Constant current mode for circuit current regulation

The dataset collected with the additional dynamic resistor load profile is referred to as *CDD-based dataset*, and the corresponding model as *CDD-based model*. The electronic load acted a configurable resistor and its value changed according to a pre-programmed pattern based on current changes recorded under real-world conditions from DC switching gear. Fig. 8(b) shows a data sample from the CDD-based dataset. Additionally, a third dataset referred to as *external dataset* was created. This dataset includes a load profile from an inductor added to the setup. The idea behind this modification was to simulate a change in the operational environment. For example power electronics in electric vehicle chargers connected as load can cause such an additional inductance. The external dataset was collected with a resistor as passive load and the electric load operating under constant current mode. When the arc occurred, the electronic load changed its resistance such that the current in the circuit was regulated to a stable level. This represents a DC system with loads that require a constant current, such as certain components in electric vehicles. Fig. 9 shows a data sample from the external dataset. The external dataset was only used in the evaluation phase to test the ability of the basic model and the CDD-based model to perform in an operational environment that was not represented in the training data.

An overview of the test bench configurations for the three datasets is summarised in Table 3.

Model training. The basic dataset and the CDD-based dataset were shuffled and divided into three parts: training, validation and test sets with a ratio of 6:2:2. This ratio is common to split the datasets for training purpose in machine learning. The size of datasets are specified in Table 4 where the class ratio represents the imbalance of the dataset between the number of negative instances labelled "no-arc" and the positive instances labelled "arc". Transient data points were removed from the datasets. These transient data occurred when the experiment setup changed state between normal operation and arc occurrence, for example during electrode separation or reconnection. This caused short periods of unstable current and voltage signals that did not represent a steady-state behaviour of either class.

Four models were trained: the basic model without frequency information (FFT) input, the basic model with FFT input, the CDD-based model without FFT input, and the CDD-based model with FFT input. The detailed training, validation, and evaluation / testing procedure is illustrated in Fig. 7. Certain hyperparameters were controlled in order to reduce the influence of the hyperparameters on the performance of the models and therefore to ensure that the models remain comparable: The size of the training datasets were set to be 60 % of the available data

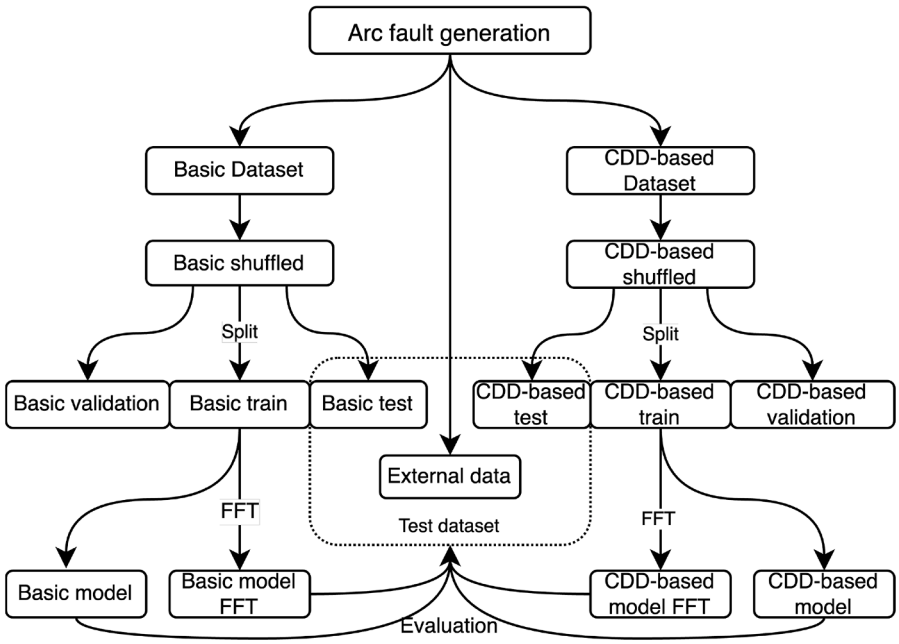
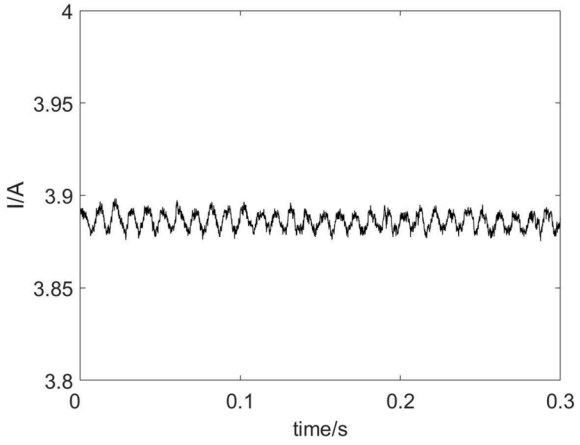
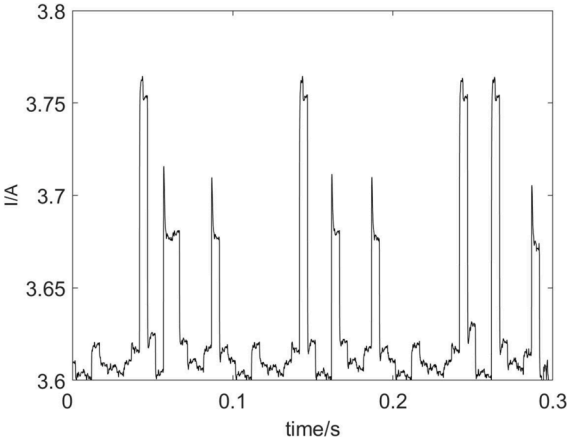


Fig. 7. Causal model validation procedure for arc fault detection. FFT refers to Fast-Fourier Transformation and indicates that frequency-spectrum information $|S(f)|$ is available.

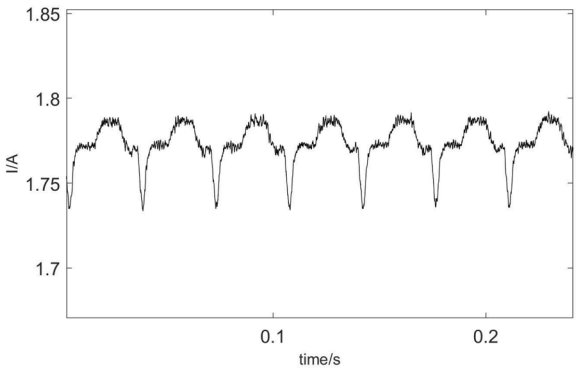


(a) Normal operation, without load profile simulation

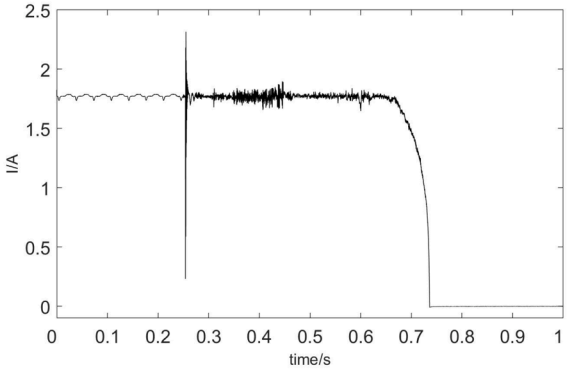


(b) Normal operation, with load profile simulation.

Fig. 8. Examples of the sampled current signal from the test bench for DC series arc fault detection..



(a) External dataset from a modified circuit setup



(b) DC series arc fault occurrence (0.26s to 0.7s) in external dataset

Fig. 9. Data sample visualisation of the external dataset.

Table 4
Size of each dataset in the validation.

Property	Dataset		
	Basic	CDD-based	External
Size (excl. transient-state samples)	31,200 (29690)	33,200 (31372)	6100 (5539)
Class ratio	4.7:1	2.5:1	3:1
Training	17,814	18,823	0
Validation	5938	6274	0
Test	5938	6275	5539

samples. The remaining 40 % were used for validation (20 %), and testing (20 %). One time-series of current data always contained 160 data points. Each data point is the current value in the circuit measured with a sampling rate of 16kHz. A fully connected neural network with three dense layers was chosen as model architecture. For each dense layer, batch normalisation was applied with a batch size of 64. *ReLu* was chosen as activation function, and each layer is followed by a dropout layer to mitigate overfitting. The models were trained with an Adam optimiser and binary cross entropy as loss function. The input layer for models with FFT accepts both time-series data and frequency-domain data, while the input layer for models without FFT only accepts time-series data. The frequency spectrum information $|S(f)|$ was created in a preprocessing step from the time-series current data through a fast-fourier transformation. The detailed specifications of the models are available in Appendix B. The trained models and their Python implementations are included as Jupyter Notebooks in the replication package of this paper.

Evaluation of the models' performance. The performance of the models was evaluated on all test datasets, i.e., the basic, CDD-based, and external dataset as listed in Tables 3 and 4. As classification metrics we included *Accuracy* and *F1-score* which indicate the overall predictive capability. *Sensitivity* measures the models' ability to correctly identify arc events, which is important to evaluate RA1.[5] On the other hand, *Specificity* is used to evaluate the ability to correctly detect a non-arc event and *Precision* assesses how many triggered alarms are actually correct which both assess RA2.[6] We also included the *Matthews Correlation Coefficient (MCC)* for a more balanced overall evaluation compared to the F1-score in the presence of imbalanced classes, which can be considered the case here (Chicco and Jurman, 2020). The individual metrics are defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{3}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{F1} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}}, \tag{5}$$

$$\text{MCC} = \frac{(\text{TP} \cdot \text{TN}) - (\text{FP} \cdot \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}, \tag{6}$$

where TP indicates true positive, TN true negative, FP false positive, and FN false negative.

The training and evaluation procedure were repeated ten times for each model. Each iteration was executed with randomly initialised weights, and randomly shuffled datasets to avoid bias introduced by the initialisation of parameters. The evaluation results for all models are presented in Fig. 10 and the confusion matrices for each model across each datasets are provided in Appendix C. Fig. 10 shows the mean and standard deviation of the metrics from all conducted experiment runs.

[5] RA1: WHEN an arc occurs THEN an alarm should be triggered

[6] RA2: WHEN no arc occurs THEN an alarm should not be triggered

6.4. Significance testing

We followed a Bayesian approach for testing statistical significance of our results following the suggestions outlined by Torkar et al. (2022), the framework for Bayesian data analysis in SE outlined by Furia et al. (2022), and the practical implementations suggested by McElreath (2018)[7]. We analysed the following cases:

1. **Case (a), CDD-based Model With FFT vs. Basic Model No FFT**: Applying the data and model requirements (A-D2, A-D3, and A-M1) from the proposed CDD-based methods has a positive effect on the metrics compared to the original model.
2. **Case (b), CDD-based Model No FFT vs. Basic Model No FFT**: Applying the proposed data requirement of adding dynamic load data (A-D2) without adopting the model requirement of adding the frequency information (A-D3, A-M1) has a positive effect on the metrics compared to the original model.
3. **Case (c1), Basic Model With FFT vs. Basic Model No FFT**: Applying the proposed model requirement of adding the frequency information (A-D3, A-M1) but not the requirement of adding dynamic load data (A-D2) has a positive effect on the metrics compared to the original model.
4. **Case (c2), CDD-based Model With FFT vs. CDD-based Model No FFT**: Applying the proposed model requirement of adding the frequency information (A-D3, A-M1) and the requirement of adding dynamic load data (A-D2) has a positive effect on the metrics compared to a model that only applies the requirement of adding dynamic load data (A-D2).

For brevity, we performed this analysis only on the results of the metrics for the external dataset. We consider the external data to be the most difficult test of model performance and robustness because the data represents previously unseen operational conditions.

Model definition. We model the performance of the two models for each metric using a Bayesian generalised linear model (GLM) with normal likelihood. The parameter β_0 is the baseline mean, which represents the mean value of the performance metric for model 1 (reference model). The parameter β_1 represents the effect of introducing model 2, i.e., β_1 **represents the change in the mean of the metric under consideration when moving from model 1 to model 2.** The shared standard deviation across both models is represented by σ. The likelihood is defined as:

$$y_i \sim \mathcal{N}(\mu_i, \sigma), \tag{7}$$

with μ_i as linear predictor defined as:

$$\mu_i = \beta_0 + \beta_1 \cdot \text{group}_i, \tag{8}$$

and $\text{group}_i \in \{0, 1\}$ indicating if the i^{th}-observation is part of the baseline model 1 or of model 2.

Choice of priors. We use weakly informative priors to allow the data to dominate the inference of the posterior distribution:

- $\beta_0 \sim \mathcal{N}(0.5, 1^2)$, which centres the baseline mean around a plausible value (0.5) for the metric,
- $\beta_1 \sim \mathcal{N}(0, 1^2)$, which is symmetric around zero to reflect no a priori preference for either model,
- $\sigma \sim \text{Exponential}(\lambda = 10)$, which ensures positivity of the variance.

These priors can be considered *plausible* for our study because we checked that the chosen priors generate values within realistic ranges of the metrics and that they avoid extreme values.

[7] An alternative approach to significance testing using Welch's *t*-test can be found in Appendix E

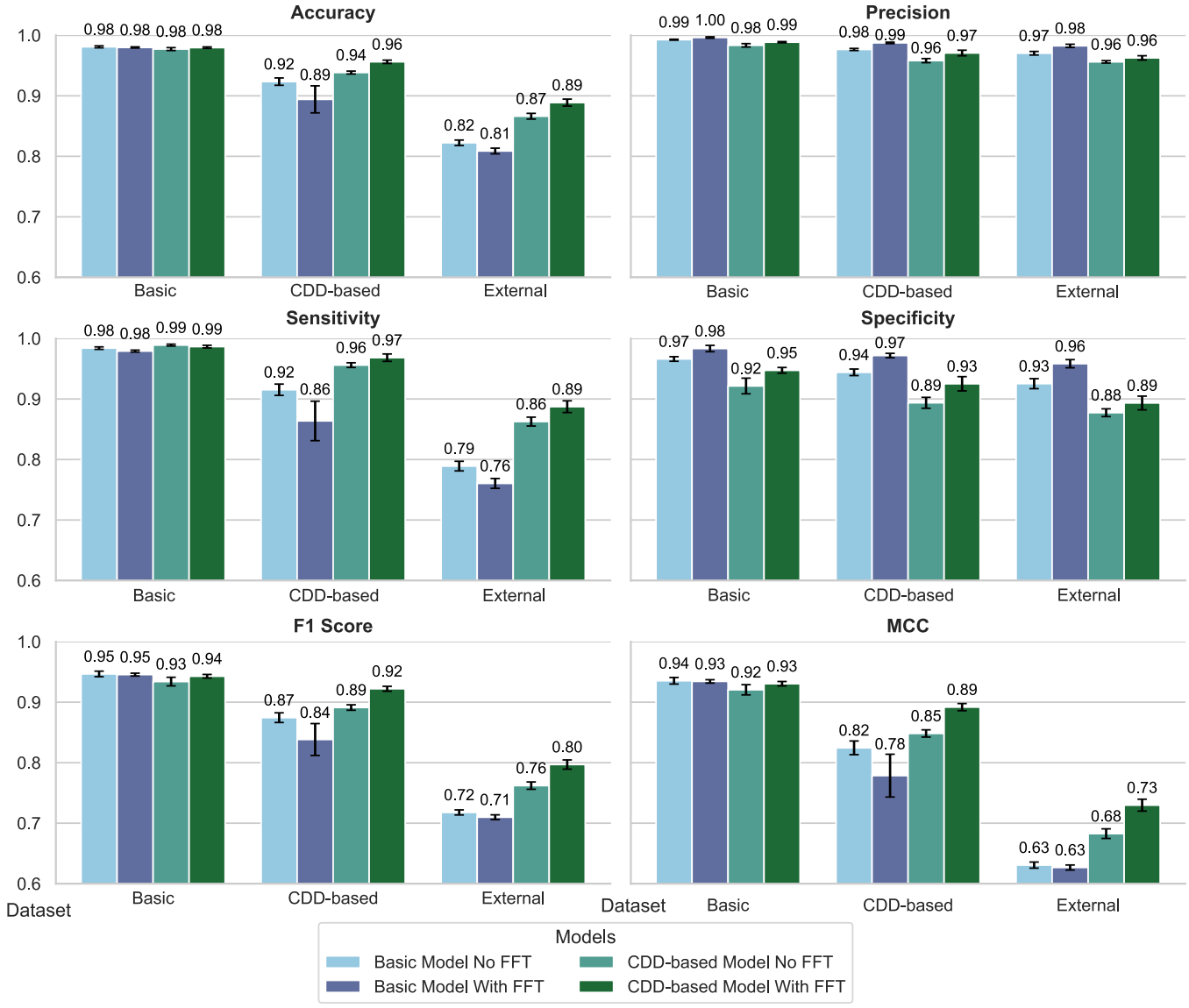
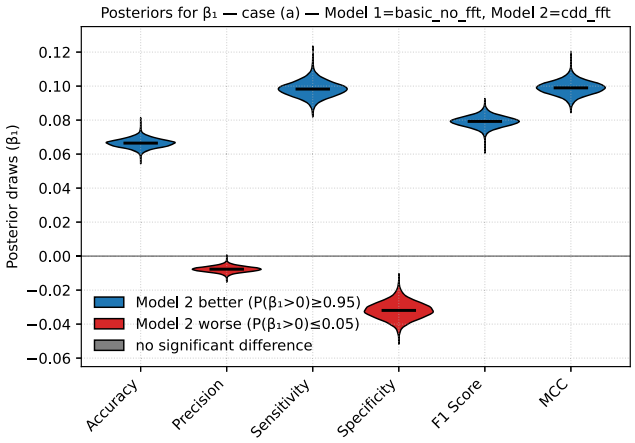


Fig. 10. Performance metrics for the use case models across all datasets..

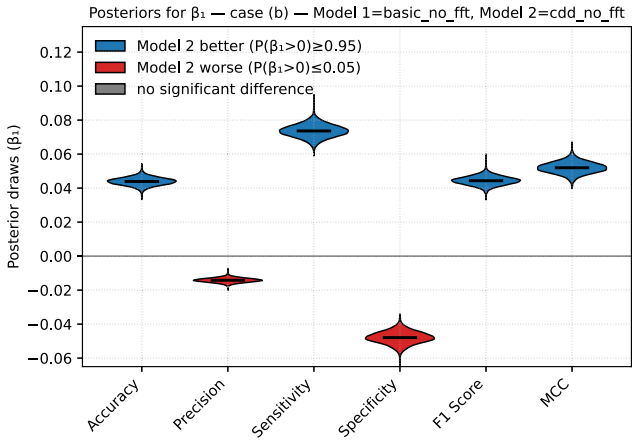
Model inference and workability. The models were fitted using the Python library `pymc` in version 5.25.1 which utilises a dynamic Hamiltonian Monte Carlo approach. We consider the models workable because we did not see divergent transitions during sampling, $\hat{R} < 1.01$, and the effective sample sizes do not show concerning decreases compared to the overall sample size. The estimated metric sizes (β_0) of the resulting posterior predictive distributions for the baseline models align well with the observed data, indicating that the results are adequate. The summaries of the posteriors for each metric and each case can be found as supplementary material in Appendix D.

Evaluation results. The estimated effect sizes β_1 for all cases (a) - (c2) are illustrated in Fig. 11. In addition to providing the density plots, the figure also indicates the posterior probability that the effect of introducing model 2 is greater than zero, i.e., $P(\beta_1 > 0)$. Specifically, if $P(\beta_1 > 0) \geq 0.95$, it indicates that there is a chance greater or equal 95 % that model 2 outperforms model 1 for this specific metric. In such cases, the posterior density is drawn in blue. Similarly, if $P(\beta_1 > 0) \leq 0.05$, it indicates that there is only a 5 % chance, or less, that model 2 outperforms model 1 for this specific metric. In such cases, the posterior density is drawn in red.

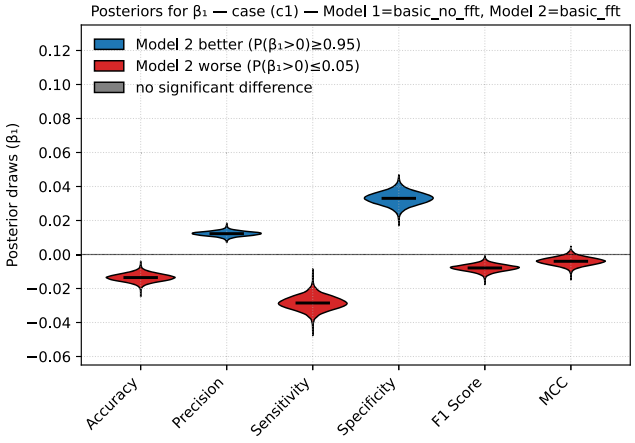
Introducing both dynamic load data (requirement A-D2), and frequency information (requirements A-D3 and A-M1) in case (a) has a significant positive effect on all considered metrics, except for *precision* and *specificity*. The *precision* of the new model is slightly lower (0.96 vs. 0.97 for the external dataset), while *specificity* is distinctly lower (0.89 vs. 0.93). However, compared to these two metrics, the positive effect on the remaining metrics is significantly higher for the CDD-based model. By separating the effect of introducing dynamic load data in case (b) from frequency information in cases (c1) and (c2), we find that it is the introduction of dynamic load (requirement A-D2) that has a negative effect on *precision* and *specificity*. On the other hand, the introduction of frequency information has a positive effect on both *precision* and *specificity*. If the frequency information is introduced to a model trained on data without dynamic load (case (c1)), the introduction of frequency information has a small negative effect on all metrics, except for *precision* and *specificity*. Introducing frequency information to a model trained on dynamic load data (case (c2)) however has a distinct positive effect on all metrics.



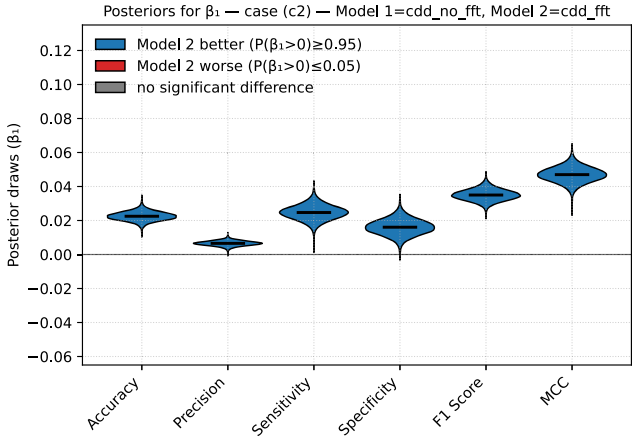
Case (a): Effect of applying both dynamic load data (A-D2) and frequency information (A-D3, A-M1).



Case (b): Effect of applying only the dynamic load data (A-D2).



Case (c1): Effect of applying only frequency information (A-D3, A-M1) on models trained without dynamic load data.



Case (c2): Effect of applying only frequency information (A-D3, A-M1) on models trained with dynamic load data.

Fig. 11. Effect sizes β_1 of introducing CDD-derived requirements on performance metrics.

7. Discussion

This section will first present a discussion of the experiment results, followed by a summary of the answers to the research questions. Then, the results of this study are discussed in the context of system prototyping, including a discussion on the current limitation of CDD, suggestions for further research, and a discussion on the threats to validity.

7.1. Discussion of the experiment results

The experiment results suggest that the CDD-based models outperform the existing basic models in terms of *accuracy*, *sensitivity*, *F1 score*, and *MCC* with significant improvement in cases when operational conditions change (external dataset), while on the other hand *precision* and *specificity* show reduced values.

Effect of introducing dynamic load data. The results suggest that it is specifically the introduction of dynamic load data (requirement A-D2) that significantly improves all metrics but also decreases *precision* slightly and *specificity* distinctly. By introducing dynamic load data, the model becomes more sensitive towards arc-events and therefore detects more true positive events. However, this shift in the decision boundary also leads to a higher chance of raising a false alarm, which is why *specificity* decreases. Given that the dataset has a significantly higher number of non-arc events (true negatives) compared to the number of arc events (true positives), the decrease in *precision* is far less distinct. However, the positive effect on sensitivity is by a factor of three larger compared to the negative effect on specificity. This is also reflected in the significant increase in overall performance indicated by the *F1 score* and *MCC*.

Effect of introducing dynamic load data
The introduction of dynamic data based on the CDD-derived requirements A-D2 leads to a model with slightly lower specificity (higher false positive rate) but substantially higher sensitivity (lower false negative rate).

Effect of introducing frequency information. The introduction of frequency information improves *specificity* and therefore also *precision*. In the case of a model trained on dynamic load data, the additional frequency information seems to also support the detection of arc events, which is why all metrics improve. However, the improvement in *specificity* due to the introduction of frequency information cannot compensate for the negative effect on *specificity* due to the introduction of the dynamic load data. Furthermore, if additional frequency information is

Effect of introducing frequency information
The introduction of frequency information based on the CDD-derived requirements A-D3 and A-M1 leads to a model with overall higher performance, but only if the model was trained on dynamic load data.

provided to a model not trained on dynamic load data, *sensitivity* and overall performance of the model is significantly reduced.

The idea behind including frequency information in the training and runtime data of the models is to create a "proxy" for identifying the load in the system that acts as a confounder as discussed earlier. In case of the basic dataset there is no additional load present in the system. Therefore, the frequency information does not provide any useful additional information for the basic model. In contrast, dynamic loads are present in the dataset for the CDD-based model. Here, the additional frequency information (FFT) provides useful information to the model allowing it to differentiate the type of load connected to the system. Therefore, the basic model cannot make use of the additional frequency information. One could, however, argue that the frequency information is inherent to the time-domain signal, as it is also suggested by the DAG in Fig. 6 (g). The deep neural network of the CDD-based model without explicit access to the frequency signal might inadvertently perform internally a frequency analysis on the time-domain signal anyway. It might therefore not be necessary to provide additional frequency information. However, we cannot be sure that this frequency analysis is actually taking place within the neural network. If, as in our case, expert knowledge indicates benefits of providing frequency information, it is advisable to provide the frequency information explicitly to the model. This advice is supported by the results of case (c2).

Considerations on why the CDD-based model might fail. While we see that the new model based on the CDD-derived requirements improves robustness towards unseen data and sensitivity, there are scenarios where it may underperform compared to the basic model:

Adding dynamic load profiles (requirement A-D2) reduces missed arc events but increases false alarms. Depending on the use case, this might be unacceptable. In the considered DC switching gear use case, a missed arc event would be more problematic compared to a false positive. As the improvement in sensitivity is threefold compared to the reduction in specificity (increase of false alarm rate), the introduction of the CDD-based model has an overall positive effect on the use case.

We see further that false positives can occur in cases of high dynamic load variations. The occurrence of an arc is accompanied by peaks across the entire frequency spectrum. We assume that if the dynamic load has a very broad frequency signature, the resulting frequency components will become nearly indistinguishable from those produced by a real arc. That might explain also why adding explicit frequency information improves the specificity (decrease of false alarm rate) for the CDD-based model. While we can assume that the deep neural network might perform a frequency analysis inadvertently based on the provided time-domain data, we cannot ensure it does this analysis for all relevant frequency. By adding the frequency information for all relevant frequency explicitly, we improved the model's capability of distinguishing arc events from non-arc events with a highly dynamic load.

A possible reason for remaining misclassification can be the presence of *hidden confounders*. These are variables that influence both arc occurrence and the observed signals but were not included in the causal models because they were unknown to the company experts. If such hidden factors exist, the CDD-based requirements may not fully capture the operational context which leads to a degraded model performance. We outline a possible solution to this limitation in Section 7.5.

7.2. Answer to the research questions (RQs)

Answer to RQ1. RQ1 asked how causal models can guide the development of software systems with ML components. The case study showed how expert-elicited causal mechanisms could be systematically encoded in DAGs. We showed that DAGs offer several benefits for requirements elicitation, integration of prior knowledge, and representation of the system's operational context. For example the concepts of d-separation and proxy variables were used to translate causal assumptions into concrete data and model requirements. This demonstrates that graphical causal models provide actionable guidance during requirements elicitation and system design. As answer to RQ1, we proposed a heuristic for constructing DAGs from expert knowledge and for deriving low-level data and model requirements from these models.

Answer to RQ2. RQ2 asked to what extent the performance and robustness of a software system with ML can be improved by incorporating prior causal knowledge during the development. To address this question, we conducted an evaluative case study on anomaly detection in a DC power distribution system. We compared a deep learning model following the CDD-derived requirements on including dynamic load data (A-D2) and frequency-information (A-D3, A-M1) to a model trained on a pre-existing dataset. We see that the inclusion of dynamic load data lead to a significant reduction in missed arc-events, i.e., higher sensitivity and therefore lower false-negative rate. This comes at the cost of an increase in the specificity which suggests a higher false alarms rate. However, the improved detection ability significantly outweighs the increased false alarm rate. The improvement in sensitivity and overall performance of the model indicated through the *F1 score* and *MCC* is especially distinct when applying the external dataset. This suggest that the robustness of the model to changes in the operational environment improved. The causal model (Fig. 6 (g)) provides an explanation for this improvement: it reveals that load is a confounder. If this confounder is not recognised in the training data - as was the case in the pre-existing dataset - the resulting model will underperform in the presence of loads. We also see, as suggested by the causal model, that the inclusion of frequency information (requirements A-D3 and A-M1) leads to an improvement across all performance metrics. We assume that frequency information provides the model with additional data allowing it to distinguish between an arc and a normal-operation condition. As answer to RQ2, CDD not only led to an improvement in most performance metrics and robustness of the ML model, but also provided an explanation to why the basic model underperformed in the presence of varying loads: the omitted confounder *load profile* induced a spurious correlation. CDD makes such constellations explicit and therefore and it increases traceability between requirements, data, and model behaviour.

7.3. Causal modelling in the context of systems prototyping

Usually, complex cyber-physical systems are developed in several phases from laboratory environment to full-scale systems. Diefenbach et al. (2019) describe four phases of prototyping from problem discovery to product delivery: In the *discover phase*, the focus is on generating knowledge and gaining insight into the use case, for example by gathering information about the context and target user group. In the *define phase*, the focus shifts to filtering and analysing the collected knowledge. In practise this means that a requirements specification is created which describes the design goals of the system. The *development phase* involves continuous refinement and improvement. The *deliver phase* entails identifying the final solution and preparing it for market launch.

The results of this engineering research suggest that CDD is beneficial in all four phases of product prototyping for systems with ML components: In the discovery phase, causal models allow the formulation of prior assumptions about the context and the intended causal relationships between the variables that describe the system. CDD can then help stakeholders, such as project managers, data scientists, and

ML experts, to define the machine learning problem in a given causal operational context and to elicit requirements for the features that need to be included (or excluded) in the dataset even before the data collection begins. In the define phase, causal models serve as an effective communication tool to visualise and analyse the causal relationships of the system in a defined context. These causal models in their graphical representation as DAGs can be considered knowledge graphs. Tiddi and Schlobach (2022) highlight the positive aspects of using knowledge graphs in ML system development in terms of explainability. However, unlike knowledge graphs, the underlying mathematical framework of causal models, such as *do-calculus*, allow for reproducible and standardised extraction of causal meaning from graphical causal models. In particular, we showed that the analysis of potential confounding between variables defining the system provide guidance on data selection for ML models early on in the design process. In the development phase, causal models can serve as guiding templates to which causal relationships can be added or removed based on the results of prototyping iterations and validations. CDD allows for a transparent and iterative definition of the causal relationship between variables from expert knowledge, which can reduce the need for data collection and ML model training iterations. This aspect of transparency in the use of expert knowledge has the additional benefit that it can lead to traceability of data requirements back to causal assumptions and expert knowledge about the use case. This traceability is essential when building a safety case for critical ML applications or for any kind of certification. In the deliver phase, causal models can effectively communicate the final causal context and data assumptions which can play an important role in certification processes (Borg et al., 2019), such as safety certification, or compliance with legislation such as the EU AI Act (Floridi et al., 2022).

7.4. Identifying inconsistencies in prior knowledge

The workflow for constructing causal models from prior knowledge assumes an iterative identification of causal mechanisms that can be added to a causal graph. For example, experts can be confronted with experimental results from a prototyping system and asked to refine the assumed causal models based on these findings. This process can reveal inconsistencies and allows for adjustments of the model. However, errors in confounders identification can directly bias the training data. If, for example, a confounder is omitted, the ML model might learn a spurious association, as shown in the arc-detection case where omitting the load profile in the training data leads to reduced robustness. However, the use of DAGs provides transparency about the assumptions made in the DAG. An approach to reduce possible inconsistencies in prior knowledge are data-driven causal discovery methods. They can support the construction of causal models by suggesting additional dependencies, for example by testing independence structures in the observed data. Methods based on independence testing, such as the PC algorithm (Spirtes and Glymour, 1991) or the fast causal inference (FCI) algorithm (Spirtes, 2001), assume that the data is independent and identically distributed (i.d.d.), which is usually not the case in practice. There are, however, recent methods for causal discovery from time-series data that are applicable to non-i.d.d. settings such as the PCMCI method (Runge et al., 2019). The performance of data-driven approaches to causal discovery depends on the availability of interventional data, i.e., data obtained under controlled interventions that disturb the system to reveal causal directions. However, if such data were available in sufficient quantity, it would also allow ML approaches, such as deep neural networks, to identify the correct causal effects.

In that sense, solely data-driven causal discovery is not effective, and we assume that human prior knowledge is still needed. We see two promising uses of data-driven causal discovery as part of CDD:

- causal discovery can suggest the existence of confounders to experts based on identified independence relations in the data or temporal dependencies (e.g., by testing for Granger causality Granger, 1988);
- causal discovery methods can be used to test emerging causal models derived from expert knowledge. Automatic tests based on causal discovery approaches can warn if a causal model suggests independence conditions that are not consistent with the data.

7.5. Limitation of CDD and future work

We recognise a number of limitations in the application of CDD:

Adopt causal models to RE. We proposed the use of DAGs as representation for causal models because of the existing body of knowledge on how to extract actionable intelligence from these graphs. However, for RE purposes, DAGs in their current form may need to be modified. We believe that CDD can be integrated into an agile workflow because the models can be iteratively extended with new knowledge and they can serve as a communication medium about causal knowledge between different stakeholders (Heyn and Knauss, 2022). For example, the edges in a DAG could carry IDs or other information that would allow for an easier mapping of elements in the DAG to the elicited requirements.

Hidden confounders. A risk to validity of the iteratively elicited causal model are hidden confounders. These are confounders that the expert did not include in the DAG because they were not aware of them (unknown unknowns). While this risk is partially mitigated in CDD by documenting causal assumptions as DAG explicitly, future research can explore how to further reduce the risk of hidden confounders. An approach could be to apply sensitivity analysis techniques that quantify how strong the influence of an unobserved variable would have to be to overturn the conclusions drawn from the current causal model.

Runtime conditions from causal models. A common problem in ML is that the models become “stale”, where changes in the operational environment over time require retraining of the model (Prapas et al., 2021). An interesting avenue for further investigation is whether identifying and documenting prior knowledge about causal mechanisms through CDD enables the derivation of runtime checks, based on the conditional independence criteria implicitly encoded in causal models. Such an approach could enhance not only the robustness of ML systems to changes in the operational environment, but also their resilience to adversarial attacks. Recent research suggests that many such attacks exploit unrecognised confounding in the model’s operational context (Ren et al., 2022; Zhao et al., 2022).

Scalability of CDD. For larger systems, constructing and maintaining comprehensive causal models may be challenging, and the approach may need to be applied modularly (e.g., by separating sub-systems) and with the support of automated tools as discussed in Section 7.4. Approaches such as *CausalOps*, introduced by Maier et al. (2024), can be used to maintain an industrial lifecycle of larger causal models. Future research is needed to assess scalability in larger settings and to build tool support that can reduce manual effort.

Transferability of CDD to other domains. In this study, we only have discussed the case of system prototyping in a research and development environment. Further research is needed to establish a more general applicability of causal models in RE for ML by applying CDD to additional use cases. We believe that causal models can lead to a better explainability of decisions made by ML, because clear cause-effect structures help stakeholders to understand the reasons for a certain ML output, see for example the work of Chou et al. (2022). However, the way CDD is applied may vary across domains: in safety-critical CPS development, such as the one described here, expert elicitation may dominate. In data-rich domains, such as e-commerce, causal discovery methods may play a larger role.

7.6. Threats to validity

We identified the following threats to validity based on criteria for judging the quality of case studies by Yin (2003):

Construct validity. A lack in construct validity can occur if the measures employed for the case study cannot accurately and comprehensively capture the phenomenon under study. For the experimental part of this study, we included metrics common in statistical learning and ML. We also included MCC as a metric used in scenarios where both false positives and false negatives must be evaluated jointly, making it particularly suitable for imbalanced anomaly detection tasks such as arc-fault detection. In cases where clear metrics were not available, we tried to describe the problem and potential solution such that we can establish a chain of evidence.

Internal validity. A lack of internal validity can cause confounding and consequently bias in the results. This is the case for example if a lack of rigour (i.e., degree of control) in the study design occurred (Slack and Draugalis Jr, 2001). We increased the degree of control over the case study with different mechanisms: We followed a defined protocol for the workshops and documented the results of each workshop for later analysis. We analysed the data with a group of four researchers to reduce the impact of personal beliefs, making the findings less prone to personal bias. We maintained consistency in controlling hyper-parameters across validation experiments to mitigate bias in the results.

External validity. The purpose of the case study was to explore a phenomena, i.e., the use of causal graphs as support for BDD in a realistic setting. With an exploratory case study we therefore cannot claim direct generalisability of the results. To ensure however that the case study represents a realistic setting, we conducted the case study together with an industrial partner. A remaining validity threat is that the use case was in a prototyping state of development and not intended yet for the mass markets. We reflect this limitation in the description of the case study. We also discuss scalability and transferability as current limitation in Section 7.4.

Reliability. To increase reliability, we publish the workshop protocols, survey data, datasets, and models as additional replication material alongside this article. Throughout the article we described the steps taken during the data collection and analysis.

8. Conclusion

The development of software systems that incorporate ML requires insight into the causal relationship between different variables to make informed decisions when specifying data for the training and validation of the ML component. In this article we discussed the limitations of current RE methods for specifying the desired behaviour of ML-enabled systems. In particular, we discussed the limitations of describing prior knowledge about cause-effect relationships in a given context, and the missing link between high-level requirements and data requirements in current RE methods. We argued that causal models in the form of DAGs can help define the operational causal context and support the elicitation of data requirements by making assumed cause-effect relationships explicit. We explored the use of causal models as part of the model specification in a case study on DC series arc fault detection in collaboration with an industrial partner. In a series of workshops, we developed and tested a concept for causality-driven development (CDD). We also demonstrated the effectiveness of the data requirements derived from the causal model of the arc fault detection use case in terms of performance and robustness of the resulting ML model. In addition, the causal model as output of CDD provided evidence as to why the robustness of the model increased by allowing developers to identify confounding and unobservable variables in the operational context. The findings answer RQ1 by showing how causal models can be used as part of requirements elicitation to derive explicit data and model requirements, and RQ2, by demonstrating that such requirements improved robustness and performance in our case study. In summary, this study showed how CDD can create a link between a specified desired causal behaviour, data requirements, and the resulting ML model behaviour.

Additional data

A replication package accompanying this article is available at https://doi.org/10.7910/DVN/XEK72T. The replication data package contains the protocols of the workshops, the final causal models, the use case datasets and the Python code for the DL models used in the evaluation of the case study.

CRediT authorship contribution statement

Hans-Martin Heyn: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization; **Yufei Mao:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Investigation, Data curation; **Roland Weiß:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition, Data curation, Conceptualization; **Eric Knauss:** Writing – review & editing, Writing – original draft, Visualization, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Data availability

All data and code available on Harvard dataverse: https://doi.org/10.8447910/DVN/XEK72T

Declaration of competing interest

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

Acknowledgements

This project has received funding from the EU's Horizon 2020 research and innovation program under grant agreement No 957197 (VEDLIoT).

Appendix A. Component list and description for the test bench representing the use case of serial arc fault detection in DC power systems

The test bench was built in accordance to the UL1699B standard which describes photovoltaic (PV) DC arc-fault circuit protection mechanisms. The following components were used:

- A power supply which provided stable voltage up to 100V.
- A programmable linear stage allowed the adjustment of the air gap between the electrodes where the arc is generated.
- A passive load in form of a resistor and/or conductor.
- A programmable electronic load which could simulate different load behaviours by changes of its inner resistor value under different modes. It was integrated in the prototype as a consequence of the causal modelling because different load profiles were identified as a confounding element. The constant resistor and constant current modes were used in this case study. The constant resistor model allowed the electronic load to change its resistor value over time according to a pre-programmed pattern. The electronic load under constant current mode could adapt its resistor value to regulate the circuit current to a given current value.

Table B.5
Hyperparameters and setting for the FNN models without and with FFT features.

Parameter	FNN without FFT	FNN with FFT
Activation	ReLU (after Batch Normalization)	
Architecture	Fully connected feedforward neural network (FNN)	
Batch size	64	
Class weighting	Inverse frequency weighting	
Dropout rate	0.25	
Epochs	Variable (`epoch` argument)	
Hidden layers	320, 640, 480 units	640, 1280, 480 units
Input dimension	160 features (time-domain only)	320 features (FFT + time-domain)
Loss function	Binary cross-entropy	
Metrics	Binary Accuracy, F1-score (threshold = 0.5)	
Optimiser	Adam, learning rate 1×10^{-6}	
Output bias initialization	$\log(\frac{\#pos}{\#neg})$	
Output layer	1 unit, sigmoid activation	
Regularization	L2, $\lambda = 1 \times 10^{-5}$	
Weight initialization	Glorot Normal	

- An ADC ADS131A02 collected data at a sampling rate of 16kHz.
- A current transducer reduced the current signal proportionally for sampling to the ADC sensing range.
- An anti-aliasing low pass filter was integrated with a cutoff frequency of 160kHz.
- An AI accelerator was utilised for data processing and execution of machine learning algorithms.

Appendix B. Detailed specifications of the FNN models used in this study

Table B.5 outlines the specifications of the FNN models trained in this study, including all relevant hyperparameter settings. The models' Python code and the trained models are available in the replication package of this study.

Appendix C. Confusion matrices for all models across all datasets

Fig. C.12 contains the confusion matrices for all models across all datasets.

Appendix D. Summary of the Bayesian GLM posteriors for the evaluation of the models

Tables D.6–D.9 lists the posterior summaries of the Bayesian GLMs for each performance metric and each case (a) - (c2) outlined in Section 6.4.

Appendix E. Significance testing using Welch's *t*-test and Hedges' g for effect size suggestion

We conducted statistical testing on the models' results on the unseen external datasets following the suggestions for statistical testing in SE experiments outlined by Arcuri and Briand (2014). For each metric outlined in Eqs. (3)–(6), we conducted the following hypothesis test:

H_0: There is no difference between the two model variants on the chosen metric.
H_1: There is a difference (two-sided).

Formally, the hypotheses can be expresses as follows:

$$H_0 : \mu_{\text{model}_1} = \mu_{\text{model}_2} \tag{E.1}$$

$$H_1 : \mu_{\text{model}_1} \neq \mu_{\text{model}_2} \tag{E.2}$$

We compared the same cases as outlined in Section 6.4. We used the *Welch's t-test* assuming approximate normality of the results because the approach is robust to unequal variances, and it works well on small sample sizes (in our case $n=10$ for each metric). Arcuri and Briand (2014) recommend to report effect sizes alongside the test results. We therefore computed *Hedges' g* to quantify the magnitude of the observed differences between the models. This is important because there are cases (metrics) in which the new model performs worse compared to the original model. Hedges' g measures the difference between two means in a unitless way relative to the pooled standard deviation of both groups (Rosenthal et al., 1994), which makes the effect size better comparable across the different metrics and models. We used a significance level of $\alpha = 0.05$ because it is considered an academic default. Table E.10 lists the results of the significance testing. The results of the *t*-test approach are mostly identical to the Bayesian approach in Section 6.4. The only minor deviation is that for case (c1), the change in MCC is not significant according to the Welsh *t*-test. However, even under the Bayesian approach, the probability $P(\beta_1 > 0)$ for MCC equals 0.04. This is very close to the threshold of $P(\beta_1 > 0) \leq 0.05$.

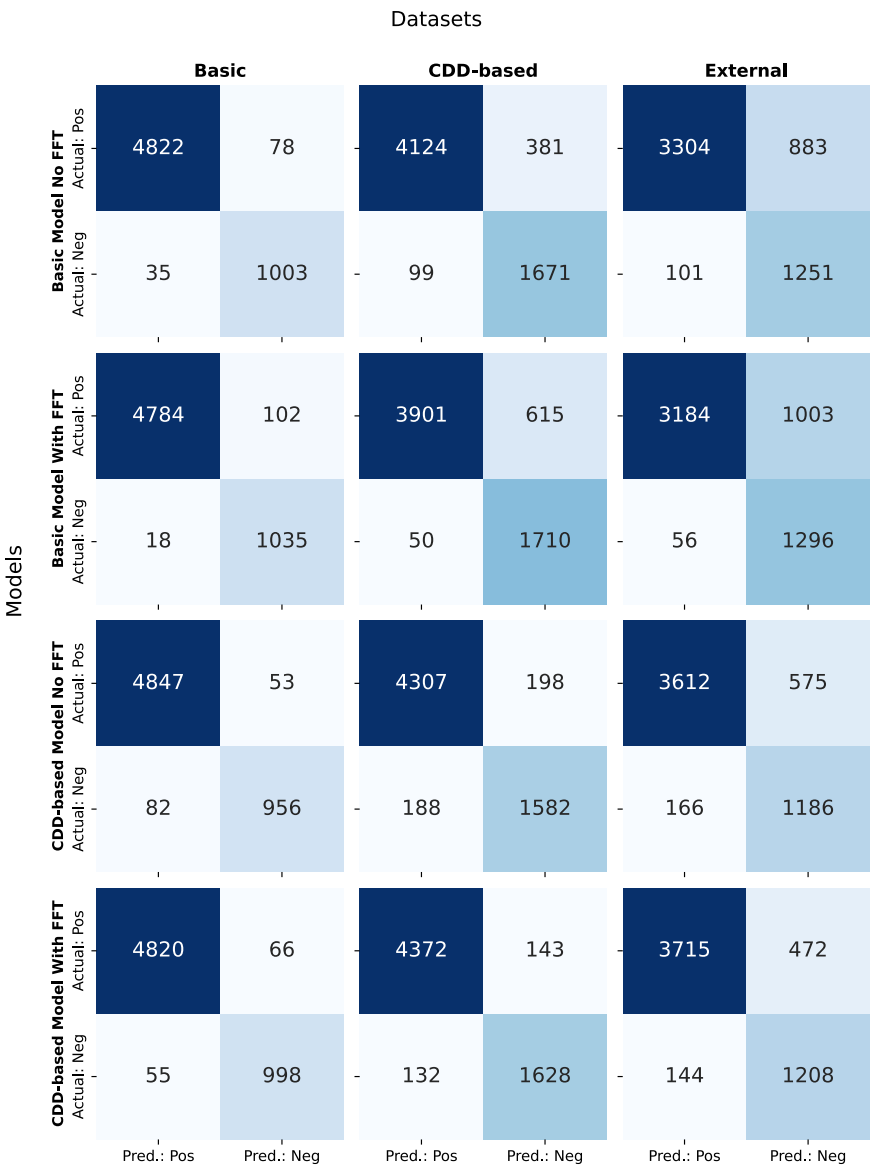


Fig. C.12. Confusion matrices for all models across all datasets.

Table D.6
Posterior summaries for case (a): Model 1 = basic_no_fft, Model 2 = cdd_fft.

Metric	β_0 mean (SD)	β_1 mean (SD)	$P(\beta_1 > 0)$	ESS_{β_0}	ESS_{β_1}	$\hat{R}_{\beta_0}$ / $\hat{R}_{\beta_1}$
Accuracy	0.8224 (0.0018)	0.0665 (0.0026)	1.00	3190	3283	1.000 / 1.000
Precision	0.9704 (0.0012)	-0.0077 (0.0016)	0.00	2831	3059	1.000 / 1.000
Sensitivity	0.7891 (0.0031)	0.0983 (0.0044)	1.00	3298	3440	1.000 / 1.000
Specificity	0.9254 (0.0035)	-0.0319 (0.0049)	0.00	3378	3245	1.000 / 1.000
F1 Score	0.7178 (0.0022)	0.0792 (0.0031)	1.00	3015	2780	1.000 / 1.000
MCC	0.6308 (0.0028)	0.0990 (0.0039)	1.00	3459	3349	1.000 / 1.000

Table D.7
Posterior summaries for case (b): Model 1 = basic_no_fft, Model 2 = cdd_no_fft.

Metric	β_0 mean (SD)	β_1 mean (SD)	$P(\beta_1 > 0)$	ESS_{β_0}	ESS_{β_1}	$\hat{R}_{\beta_0}$ / $\hat{R}_{\beta_1}$
Accuracy	0.8223 (0.0016)	0.0439 (0.0023)	1.00	3646	3602	1.000 / 1.000
Precision	0.9703 (0.0009)	-0.0142 (0.0013)	0.00	3270	3449	1.000 / 1.000
Sensitivity	0.7891 (0.0027)	0.0736 (0.0038)	1.00	3573	3397	1.000 / 1.000
Specificity	0.9253 (0.0026)	-0.0479 (0.0036)	0.00	3508	3394	1.000 / 1.000
F1 Score	0.7177 (0.0018)	0.0443 (0.0026)	1.00	3370	3368	1.000 / 1.000
MCC	0.6307 (0.0024)	0.0520 (0.0033)	1.00	3406	3321	1.000 / 1.000

Table D.8
Posterior summaries for case (c1): Model 1 = basic_no_fft, Model 2 = basic_fft.

Metric	β_0 mean (SD)	β_1 mean (SD)	$P(\beta_1 > 0)$	ESS_{β_0}	ESS_{β_1}	$\hat{R}_{\beta_0}$ / $\hat{R}_{\beta_1}$
Accuracy	0.8224 (0.0016)	-0.0136 (0.0023)	0.00	3647	3593	1.000 / 1.000
Precision	0.9704 (0.0010)	0.0123 (0.0014)	1.00	3225	3383	1.000 / 1.000
Sensitivity	0.7891 (0.0029)	-0.0285 (0.0040)	0.00	3118	2981	1.000 / 1.000
Specificity	0.9253 (0.0026)	0.0330 (0.0037)	1.00	3216	3314	1.000 / 1.000
F1 Score	0.7178 (0.0014)	-0.0079 (0.0020)	0.00	3154	3224	1.000 / 1.000
MCC	0.6307 (0.0016)	-0.0040 (0.0023)	0.04	3245	3217	1.000 / 1.000

Table D.9
Posterior summaries for case (c2): Model 1 = cdd_no_fft, Model 2 = cdd_fft.

Metric	β_0 mean (SD)	β_1 mean (SD)	$P(\beta_1 > 0)$	ESS_{β_0}	ESS_{β_1}	$\hat{R}_{\beta_0}$ / $\hat{R}_{\beta_1}$
Accuracy	0.8663 (0.0018)	0.0225 (0.0026)	1.00	3365	3316	1.000 / 1.000
Precision	0.9561 (0.0010)	0.0066 (0.0014)	1.00	3056	3056	1.000 / 1.000
Sensitivity	0.8627 (0.0031)	0.0247 (0.0043)	1.00	3403	3348	1.000 / 1.000
Specificity	0.8774 (0.0033)	0.0161 (0.0046)	1.00	3170	3269	1.000 / 1.000
F1 Score	0.7620 (0.0024)	0.0350 (0.0033)	1.00	3330	3325	1.000 / 1.000
MCC	0.6828 (0.0031)	0.0470 (0.0045)	1.00	2814	2926	1.000 / 1.000

Table E.10
Welch's *t*-test on the External dataset for comparisons (a)–(d). SD: Standard Deviation. Diff: Difference. "Rej. H_0?" indicates whether the null hypothesis is rejected at $\alpha = 0.05$. Metrics in bold indicate that the Hedges g suggests a positive effect of model 1 compared to model 2 for this metric.

Metric	Model 1 Mean (SD)	Model 2 Mean (SD)	Diff.	t	Hedges g	p	Rej. H_0?
(a) Model 1 = cdd_fft, Model 2 = basic_no_fft							
Accuracy	0.8888 (0.0058)	0.8224 (0.0045)	0.0664	28.7088	12.2965	$7.57 \cdot 10^{-16}$	Yes
F1 Score	0.7969 (0.0076)	0.7178 (0.0042)	0.0792	28.8224	12.3451	$7.50 \cdot 10^{-14}$	Yes
MCC	0.7298 (0.0098)	0.6307 (0.0051)	0.0991	28.3598	12.1470	$2.18 \cdot 10^{-13}$	Yes
Precision	0.9627 (0.0035)	0.9704 (0.0030)	-0.0077	-5.2758	-2.2597	$5.67 \cdot 10^{-5}$	Yes
Sensitivity	0.8873 (0.0097)	0.7891 (0.0080)	0.0982	24.7336	10.5938	$5.84 \cdot 10^{-15}$	Yes
Specificity	0.8934 (0.0113)	0.9254 (0.0083)	-0.0320	-7.1913	-3.0802	$1.82 \cdot 10^{-6}$	Yes
(b) Model 1 = cdd_no_fft, Model 2 = basic_no_fft							
Accuracy	0.8663 (0.0047)	0.8224 (0.0045)	0.0439	21.2530	9.1030	$3.54 \cdot 10^{-14}$	Yes
F1 Score	0.7621 (0.0060)	0.7178 (0.0042)	0.0443	19.0775	8.1712	$1.94 \cdot 10^{-12}$	Yes
MCC	0.6827 (0.0079)	0.6307 (0.0051)	0.0520	17.4814	7.4876	$1.65 \cdot 10^{-11}$	Yes
Precision	0.9561 (0.0020)	0.9704 (0.0030)	-0.0143	-12.6347	-5.4116	$1.13 \cdot 10^{-9}$	Yes
Sensitivity	0.8627 (0.0073)	0.7891 (0.0080)	0.0736	21.5187	9.2168	$3.15 \cdot 10^{-14}$	Yes
Specificity	0.8774 (0.0064)	0.9254 (0.0083)	-0.0480	-14.5020	-6.2115	$5.66 \cdot 10^{-11}$	Yes
(c1) Model 1 = basic_fft, Model 2 = basic_no_fft							
Accuracy	0.8088 (0.0047)	0.8224 (0.0045)	-0.0136	-6.6009	-2.8273	$3.40 \cdot 10^{-6}$	Yes
F1 Score	0.7099 (0.0040)	0.7178 (0.0042)	-0.0079	-4.3416	-1.8596	$3.96 \cdot 10^{-4}$	Yes
MCC	0.6267 (0.0042)	0.6307 (0.0051)	-0.0040	-1.9152	-0.8203	$7.21 \cdot 10^{-2}$	No
Precision	0.9827 (0.0026)	0.9704 (0.0030)	0.0123	9.8579	4.2223	$1.31 \cdot 10^{-8}$	Yes
Sensitivity	0.7605 (0.0081)	0.7891 (0.0080)	-0.0286	-7.9590	-3.4090	$2.65 \cdot 10^{-7}$	Yes
Specificity	0.9584 (0.0067)	0.9254 (0.0083)	0.0330	9.7714	4.1852	$1.87 \cdot 10^{-8}$	Yes
(c2) Model 1 = cdd_fft, Model 2 = cdd_no_fft							
Accuracy	0.8888 (0.0058)	0.8663 (0.0047)	0.0225	9.5585	4.0941	$2.51 \cdot 10^{-8}$	Yes
F1 Score	0.7969 (0.0076)	0.7621 (0.0060)	0.0348	11.3377	4.8561	$2.21 \cdot 10^{-9}$	Yes
MCC	0.7298 (0.0098)	0.6827 (0.0079)	0.0471	11.7888	5.0494	$1.12 \cdot 10^{-9}$	Yes
Precision	0.9627 (0.0035)	0.9561 (0.0020)	0.0066	5.1106	2.1890	$1.51 \cdot 10^{-4}$	Yes
Sensitivity	0.8873 (0.0097)	0.8627 (0.0073)	0.0246	6.4087	2.7450	$7.00 \cdot 10^{-6}$	Yes
Specificity	0.8934 (0.0113)	0.8774 (0.0064)	0.0161	3.8957	1.6686	$1.58 \cdot 10^{-3}$	Yes

References

Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., Grundy, J., 2023. Requirements engineering for artificial intelligence systems: a systematic mapping study. Inf Softw. Technol. 158.

Anwer, S., Ikram, N., 2006. Goal oriented requirement engineering: a critical study of techniques. In: 2006 13th Asia Pacific Software Engineering Conference (APSEC'06). IEEE, pp. 121–130.

Arcuri, A., Briand, L., 2014. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. Softw. Test. Verif. Reliab. 24 (3), 219–250.

Binamungu, L.P., Embury, S.M., Konstantinou, N., 2018. Maintaining behaviour driven development specifications: challenges and opportunities. In: 2018IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, pp. 175–184.

Borg, M., Englund, C., Wnuk, K., Duran, B., Levandowski, C., Gao, S., Tan, Y., Kaijser, H., Lönn, H., Törnqvist, J., 2019. Safely entering the deep: a review of verification and validation for machine learning and a challenge elicitation in the automotive industry. J. Autom. Softw. Eng. 1 (1), 1–19.

Bradley, J.H., Paul, R., Seeman, E., 2006. Analyzing the structure of expert knowledge. Inf. Manag. 43 (1), 77–91.

Buchgeher, G., Gabauer, D., Martinez-Gil, J., Ehrlinger, L., 2021. Knowledge graphs in manufacturing and production: a systematic literature review. IEEE Access 9, 55537–55554.

Carlini, N., Wagner, D., 2017. Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (Sp). IEEE, pp. 39–57.

Chae, S., Park, J., Oh, S., 2016. Series DC Arc fault detection algorithm for DC microgrids using relative magnitude comparison. IEEE J. Emerg. Sel. Top. Power Electron. 4 (4), 1270–1278.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., Wirth, R., 2000. CRISP-DM 1.0: Step-by-Step Data Mining Guide. SPSS, Chicago.

Chicco, D., Jurman, G., 2020. The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation. BMC Genomics 21 (1), 6.

Chou, Y.-L., Moreira, C., Bruza, P., Ouyang, C., Jorge, J., 2022. Counterfactuals and causability in explainable artificial intelligence: theory, algorithms, and applications. Inf. Fus. 81, 59–83.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M.D., et al., 2022. Underspecification presents challenges for credibility in modern machine learning. J. Mach. Learn. Res. 23 (226), 1–61.

Dang, H.-L., Kim, J., Kwak, S., Choi, S., 2021. Series DC Arc fault detection using machine learning algorithms. IEEE Access 9, 133346–133364.

Daniel, K., 2017. Thinking, Fast and Slow. Farrar, Straus og Giroux, New York.

Diefenbach, S., Christoforakos, L., Maisch, B., Kohler, K., 2019. The state of prototyping practice in the industrial setting: potential, challenges and implications. In: Proceedings of the Design Society: International Conference on Engineering Design. Vol. 1. Cambridge University Press, pp. 1703–1712.

Ehrlinger, L., Wöß, W., 2016. Towards a definition of knowledge graphs. SEMANTiCS (Posters, Demos, SuCCESS) 48 (1–4), 2.

Elwert, F., 2013. Graphical causal models. In: Handbook of Causal Analysis for Social Research. Springer, pp. 245–273.

Fischbach, J., Hauptmann, B., Konwitschny, L., Spies, D., Vogelsang, A., 2020. Towards causality extraction from requirements. In: 2020IEEE 28th International Requirements Engineering Conference (RE). IEEE, pp. 388–393.

Floridi, L., Holweg, M., Taddeo, M., Amaya Silva, J., Mökander, J., Wen, Y., 2022. CapAI-a procedure for conducting conformity assessment of AI systems in line with the EU artificial intelligence act. Available at SSRN 4064091 .

Frisch, M., 2014. Causal Reasoning in Physics. Cambridge University Press, Cambridge.

Furia, C.A., Torkar, R., Feldt, R., 2022. Applying bayesian analysis guidelines to empirical software engineering data: the case of programming languages and code quality. ACM Trans. Softw. Eng. Methodol. 31 (3), 1–38.

Gervasi, V., Zowghi, D., 2005. Reasoning about inconsistencies in natural language requirements. ACM Trans. Softw. Eng. Methodol. 14 (3), 277–330.

Giamattei, L., Guerriero, A., Pietrantuono, R., Russo, S., 2024. Causal reasoning in software quality assurance: a systematic review. IST, 178 107599.

Glymour, M., Pearl, J., Jewell, N.P., 2016. Causal Inference in Statistics: A Primer. John Wiley & Sons, Hoboken, New Jersey.

Goodfellow, I., McDaniel, P., Papernot, N., 2018. Making machine learning robust against adversarial inputs. Commun. ACM 61 (7), 56–66.

Granger, C. W.J., 1988. Some recent development in a concept of causality. J. Econom. 39 (1–2), 199–211.

Habiba, U.-E., Haug, M., Bogner, J., Wagner, S., 2024. How mature is requirements engineering for AI-based systems? a systematic mapping study on practices, challenges, and future research directions. Requir. Eng.. 29 (4), 567–600.

Hernán, M.A., Hsu, J., Healy, B., 2019. A second chance to get causal inference right: a classification of data science tasks. Chance 32 (1), 42–49.

Hernán, M.A., Robins, J.M., 2020. Causal Inference: What If. Chapman & Hall/CRC, London.

Heyn, H.-M., Knauss, E., 2022. Structural causal models as boundary objects in AI system development. In: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI, pp. 43–45.

Heyn, H.-M., Knauss, E., Malleswaran, I., Dinakaran, S., 2023. An investigation of challenges encountered when specifying training data and runtime monitors for safety critical ML applications. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, pp. 206–222.

Heyn, H.-M., Mao, Y., Weiss, R., Knauss, E., 2025. Causal models in requirement specifications for machine learning: a vision. In: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, pp. 1402–1405.

Heyn, H.-M., Subbiah, P., Linder, J., Knauss, E., Eriksson, O., 2022. Setting AI in context: a case study on defining the context and operational design domain for automated driving. In: International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, pp. 199–215.

Irshad, M., Britto, R., Petersen, K., 2021. Adapting behavior driven development (BDD) for large-scale software systems. J. Syst. Softw. 177, 110944.

Ishikawa, F., Matsuno, Y., 2020. Evidence-driven requirements engineering for uncertainty of machine learning-based systems. In: 2020IEEE 28th International Requirements Engineering Conference (RE). IEEE, pp. 346–351.

Kaiser, M., Griessl, R., Kucza, N., Haumann, C., Tigges, L., Mika, K., Hagemeyer, J., Porrmann, F., Rückert, U., vor dem Berge, M., et al., 2022. Vedliot: very efficient deep learning in iot. In: 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, pp. 963–968.

Knauss, A., Damian, D., Franch, X., Rook, A., Müller, H.A., Thomo, A., 2016. Acon: a learning-based approach to deal with uncertainty in contextual requirements at runtime. Inf. Softw. Technol. 70, 85–99.

Knauss, A., Damian, D., Schneider, K., 2014. Eliciting contextual requirements at design time: a case study. In: 2014 IEEE 4th International Workshop on Empirical Requirements Engineering (EmpiRE). IEEE, pp. 56–63.

Kostis, J.B., Dobrzynski, J.M., 2020. Limitations of randomized clinical trials. Am. J. Cardiol. 129, 109–115.

Kuwajima, H., Yasuoka, H., Nakae, T., 2020. Engineering problems in machine learning systems. Mach. Learn. 109 (5), 1103–1126.

Lu, S., Ma, R., Sirojan, T., Phung, B.T., Zhang, D., 2021. Lightweight transfer nets and adversarial data augmentation for photovoltaic series arc fault detection with limited fault data. Int. J. Electr. Power Energy Syst. 130, 107035.

Lu, S., Phung, B.T., Zhang, D., 2018. A comprehensive review on DC arc faults and their diagnosis methods in photovoltaic systems. Renew. Sustain. Energy Rev. 89, 88–98.

Lu, S., Sahoo, A., Ma, R., Phung, B.T., 2020. DC Series Arc fault detection using machine learning in photovoltaic systems: recent developments and challenges. In: 2020 8th International Conference on Condition Monitoring and Diagnosis (CMD). IEEE, pp. 416–421.

Maier, R., Schlattl, A., Guess, T., Mottok, J., 2024. Causalops-towards an industrial lifecycle for causal probabilistic graphical models. Inf. Softw. Technol. 174, 107520.

Martínez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernández-Orallo, J., Kull, M., Lachiche, N., Ramírez-Quintana, M.J., Flach, P., 2019. CRISP-DM twenty years later: from data mining processes to data science trajectories. IEEE Trans. Knowl. Data Eng. 33 (8), 3048–3061.

McElreath, R., 2018. Statistical Rethinking: A Bayesian Course with Examples in R and Stan. Chapman and Hall/CRC.

Mitchell, S., Potash, E., Barocas, S., D'Amour, A., Lum, K., 2021. Algorithmic fairness: choices, assumptions, and definitions. Annu. Rev. Stat. Appl. 8, 141–163.

Molak, A., Jaokar, A., 2023. Causal Inference and Discovery in Python. Packt Publishing, Birmingham.

Pearce, N., Lawlor, D.A., 2016. Causal inference-so much more than statistics. Int. J. Epidemiol. 45 (6), 1895–1903.

Pearl, J., 2009. Causality. Cambridge university press.

Pearl, J., 2019. The limitations of opaque learning machines. In: Brockman, J. (Ed.), Possible Minds: 25 Ways of Looking at AI. Penguin Press, London. chapter 2, pp. 13–19.

Pearl, J., Glymour, M., Jewell, N.P., 2016. Causal Inference in Statistics: A Primer. John Wiley & Sons.

Pei, Z., Liu, L., Wang, C., Wang, J., 2022. Requirements engineering for machine learning: a review and reflection. In: 2022IEEE 30th International RE Conference Workshops (REW). IEEE, pp. 166–175.

Peters, J., Janzing, D., Schölkopf, B., 2017. Elements of Causal Inference: Foundations and Learning Algorithms. The MIT Press, Cambridge, Massachusetts.

Prapas, I., Derakhshan, B., Mahdiraji, A.R., Markl, V., 2021. Continuous training and deployment of deep learning models. Datenbank-Spektrum 21 (3), 203–212.

Reichenbach, H., 1956. The Direction of Time. Vol. 65. University of California Press, Berkeley.

Ren, M., Wang, Y.-L., He, Z.-F., 2022. Towards interpretable defense against adversarial attacks via causal inference. Mach. Intell. Res. 19 (3), 209–226.

Rohrer, J.M., 2018. Thinking clearly about correlations and causation: graphical causal models for observational data. Adv. Methods Prac. Psychol. Sci. 1 (1), 27–42.

Rosenthal, R., Cooper, H., Hedges, L., et al., 1994. Parametric measures of effect size. Handbook Res. Synth. 621 (2), 231–244.

Runge, J., Nowack, P., Kretschmer, M., Flaxman, S., Sejdinovic, D., 2019. Detecting and quantifying causal associations in large nonlinear time series datasets. Sci. Adv. 5 (11), eaau4996.

Schölkopf, B., 2022. Causality for machine learning. In: Probabilistic and Causal Inference: The Works of Judea Pearl, pp. 765–804.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N.R., Kalchbrenner, N., Goyal, A., Bengio, Y., 2021. Toward causal representation learning. Proc. IEEE 109 (5), 612–634.

Schröer, C., Kruse, F., Gómez, J.M., 2021. A systematic literature review on applying CRISP-DM process model. Procedia Comput. Sci. 181, 526–534.

Shpitser, I., VanderWeele, T., Robins, J.M., 2010. On the validity of covariate adjustment for estimating causal effects. In: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, pp. 527–536.

Slack, M.K., Draugalis Jr, , J.R., 2001. Establishing the internal and external validity of experimental studies. Am. J. Health-Syst. Pharm. 58 (22), 2173–2181.

Smart, J., 2014. BDD in Action: Behavior-driven development for the whole software lifecycle. Simon and Schuster, New York.

Spirtes, P., 2001. An anytime algorithm for causal inference. In: International Workshop on Artificial Intelligence and Statistics. PMLR, pp. 278–285.
Spirtes, P., Glymour, C., 1991. An algorithm for fast recovery of sparse causal graphs. Soc. Sci. Comput. Rev. 9 (1), 62–72.
Spirtes, P., Glymour, C.N., Scheines, R., Heckerman, D., 2000. Causation, Prediction, and Search. MIT press, Cambridge, Massachusetts.
Textor, J., Hardt, J., Knüppel, S., 2011. DAGItty: a graphical tool for analyzing causal diagrams. Epidemiology 22 (5), 745.
Tiddi, I., Schlobach, S., 2022. Knowledge graphs as tools for explainable machine learning: a survey. Artif. Intell. 302, 103627.
Torkar, R., Furia, C.A., Feldt, R., de Oliveira Neto, F.G., Gren, L., Lenberg, P., Ernst, N.A., 2022. A method to assess and argue for practical significance in software engineering. IEEE Trans. Softw. Eng. 48 (6), 2053–2065.
UL, 2013. UL 1699B: Outline of Investigation for Photovoltaic (PV) DC Arc-Fault Circuit Protection. Underwriters Laboratories Inc., Northbrook, IL, USA.
van Lamsweerde, A., 2001. Goal-oriented requirements engineering: a guided tour. In: Proceedings 5th IEEE International Symposium on Requirements Engineering. IEEE, pp. 249–262.
Von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., et al., 2021. Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems. IEEE Trans. Knowl. Data Eng. 35 (1), 614–633.
Vowels, M.J., Camgoz, N.C., Bowden, R., 2022. D'ya like dags? a survey on structure learning and causal discovery. ACM Comput. Surv. 55 (4), 1–36.
Wirth, R., Hipp, J., 2000. CRISP-DM: towards a standard process model for data mining. In: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining. Vol. 1. Manchester, pp. 29–39.
Xiong, Q., Ji, S., Zhu, L., Zhong, L., Liu, Y., 2017. A novel DC arc fault detection method based on electromagnetic radiation signal. IEEE Trans. Plasma Sci. 45 (3), 472–478.
Yin, R.K., 2003. Designing case studies. Qualit. Res. Methods 5 (14), 359–386.
Zhao, H., Ma, C., Dong, X., Luu, A.T., Deng, Z.-H., Zhang, H., 2022. Certified robustness against natural language attacks by causal intervention. In: International Conference on Machine Learning. PMLR, pp. 26958–26970.
Zheng, H., Wang, R., Yang, Y., Yin, J., Li, Y., Li, Y., Xu, M., 2019. Cross-domain fault diagnosis using knowledge transfer strategy: a review. IEEE Access 7, 129260–129290.