Computers in Industry 148 (2023) 103892

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

Enabling causality learning in smart factories with hierarchical digital twins

Marco Lippi, Matteo Martinelli *, Marco Picone, Franco Zambonelli

Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, via Amendola 2, 42122 Reggio Emilia, Italy

ARTICLE INFO

Dataset link: Enabling Causality wDTs (Code and data)

Keywords:
Smart factories
Causal models
Digital twins

ABSTRACT

Smart factories are complex systems where many different components need to interact and cooperate in order to achieve common goals. In particular, devices must be endowed with the skill of learning how to react in front of evolving situations and unexpected scenarios. In order to develop these capabilities, we argue that systems will need to build an internal, and possibly shared, representation of their operational world that represents causal relations between actions and observed variables. Within this context, digital twins will play a crucial role, by providing the ideal infrastructure for the standardisation and digitisation of the whole industrial process, laying the groundwork for the high-level learning and inference processes. In this paper, we introduce a novel hierarchical architecture enabled by digital twins, that can be exploited to build logical abstractions of the overall system, and to learn causal models of the environment directly from data. We implement our vision through a case study of a simulated production process. Our results in that scenario show that Bayesian networks and intervention via do-calculus can be effectively exploited within the proposed architecture to learn interpretable models of the environment. Moreover, we evaluate how the use of digital twins has a strong impact on the reduction of the physical complexity perceived by external applications.

1. Introduction

Manufacturing operations strongly depend on a variety of different factors and situations whose management and coordination typically require domain knowledge, decision making and problem solving skills, and ability to react to rare or unexpected events. These challenges are nowadays exacerbated by increasing market demands, customised products, shorter lead times, faster deliveries, search for sustainable solutions (ElMaraghy et al., 2012; Wiendahl and Scholtissek, 1994).

For all these reasons, a smart factory is a clear example of a highly complex domain where artificial intelligence (AI) systems have to be combined and integrated within a physical, dynamic environment (Salierno et al., 2021). Dealing with such a scenario clearly poses a number of challenges for the management of efficient smart devices. On the one hand, there is the need to cope with the physical heterogeneity and fragmentation of operational devices, that can strongly limit data collection, interoperability and interaction. On the other hand, AI systems embodied within the environment should be easily inspectable by humans, thus providing an *interpretable* interface to explain their choices and behaviour (Gao et al., 2020).

Within this framework, digital twins (DTs) can provide the strategic infrastructure for an effective abstraction and coordination of novel smart devices. DTs can in fact enable collection and integration of data coming from heterogeneous sources, providing a common ground for the uniform digitisation of the whole information process driving the factory of the future (Tao et al., 2019b; Hribernik et al., 2021). Furthermore, they can augment the capabilities of the deployed devices, by modelling relationships that characterise physical assets (PAs), and by enabling their composition into new digital entities that support not only discovery and learning (Eirinakis et al., 2020), but also reasoning upon dynamic strategies for optimisation goals (Abburu et al., 2020).

All the devices and components acting in such a complex system, in fact, need to develop the capability to autonomously and dynamically learn how to react and to adapt in front of evolving situations, both at the individual and at the collective level. It is, in fact, unfeasible to embed smart devices with handcrafted models to be used for decision making in any critical situation. In order to gradually acquire these adapting skills, it is crucial for the systems to learn an own internal model of the environment, representing relations and dependencies between different components and variables. The abstraction level provided by DTs represents a key enabler for this ambition. Since many control and management tasks in manufacturing processes deal with the identification of reasons for specific behaviours and situations (Miguéis et al., 2022), such as in root cause analysis (Vo et al., 2020), we argue that the learned models should represent *causal*

* Corresponding author.
E-mail addresses: marco.lippi@unimore.it (M. Lippi), matteo.martinelli@unimore.it (M. Martinelli), marco.picone@unimore.it (M. Picone), franco.zambonelli@unimore.it (F. Zambonelli).

https://doi.org/10.1016/j.compind.2023.103892
Received 13 April 2022; Received in revised form 15 December 2022; Accepted 3 March 2023
Available online 21 March 2023

relations. Exploiting causality, in fact, contributes both to enhance system explainability (Chou et al., 2022) (a crucial property having humans-in-the-loop) and to improve generalisation skills, by transferring the acquired knowledge to different situations and tasks, including scenarios that involve forms of reasoning (Schölkopf et al., 2021).

Classic applications of causal models and causality learning to smart factories include fault detection and root cause analysis (Vuković and Thalmann, 2022). While the former focuses on the task of detecting, classifying, and thus properly handling faults to minimise their impact on production activities, the latter aims to identify the very first cause in the cause–effect chain, and eliminate it with a corrective permanent action. Causality can also improve different forms of automation. For example, in the scenario of dynamic and adaptive scheduling (Li et al., 2023; Chen et al., 2018; Lou et al., 2012), a machine stop can be classified in different ways according to the root cause of the fault. In this regard, different types of causes can lead to different rescheduling approaches, maintaining an overall transparency to the management with respect to the choices made by the automatic system.

Despite this increasing interest in the application of causal models to the industrial domain, the problem of modelling a homogeneous infrastructure from the diversity of physical devices has too often been decoupled from the high-level aspect of causal learning and discovery.

In this paper, we present a novel architecture based on DTs that enables the implementation of hierarchical abstractions of a manufacturing process, supporting the development of machine learning applications and focusing on causal model learning. Our contributions can be summarised as follows.

- We describe a hierarchy of abstractions that exploits DTs for modelling the digital counterpart of a physical manufacturing process.
- As a proof-of-concept of this vision, we present a scenario that simulates the dynamics of a production process in a smart manufacturing factory.
- By exploiting the proposed hierarchical structure, we show how the learning process of a causal model representing the environment can take place, without being affected by the complexity and heterogeneity of the different devices in the physical layer.
- To this aim, we exploit Bayesian networks and intervention via do-calculus (Pearl and Mackenzie, 2018) to directly learn a causal model from data observations.
- We estimate the impact of digital twins on reducing the physical complexity of the considered system.

The paper is structured as follows: Section 2 presents related works, whereas in Section 3 we illustrate the architecture of the system that we envisage by enabling the use of DTs within a smart factory. Then, in Section 4 we give some background on causal models and we highlight the importance of adopting such frameworks to enhance system interpretability. Section 5 describes the experiments that we performed on a simulated case study. Finally, Section 6 concludes the paper with open challenges and future research directions.

2. Related works

The possibility to effectively merge the digital and physical layers recently emerged as the one of the key challenges for the factory of the future. This scenario, in fact, needs the integration of a plethora of heterogeneous components characterised by a massive fragmentation in terms of protocols, data formats and interaction patterns. No single framework or recipe can obviously tackle the existing multifaceted nature of the problem, and the current approaches (Vargas et al., 2016; Ferrer et al., 2018; Liu et al., 2017; Schranz et al., 2021) mainly based on siloed, centralised and static management models are not expected to be the right long-term solution.

Digital Twins, introduced between 1999 and 2002 (Tao et al., 2019b), have recently been classified as one of the top-ten strategic technological trends of the last years in manufacturing (Botkina et al., 2018) and some argue that half of all corporations worldwide might be using them in the near future (Tao and Qi, 2019). DTs are quickly evolving from their original research field and emerging as a new cross-domain paradigm to design and implement cyber–physical applications through the creation of synchronised software replicas of physical devices, products, and organisations (Tao et al., 2019a). Research is moving fast towards the definition of a general way to bridge the physical and the digital spaces (Saracco, 2019), also thanks to a new pervasive vision that aims to create a distributed ecosystem of interconnected DTs (Ricci et al., 2022), fostering initial visions proposed in earlier literature (Members of the Digital Framework Task Group, 2018; Gelernter, 1991).

Recently, different applications and frameworks have been proposed to support DTs in the context of product design, simulation, manufacturing and augmented reality. The proposed architectures have been reasonably designed with the aim to target a specific application domain and without providing an interoperable approach where multiple DTs can coexist and cooperate (Cimino et al., 2019). New platforms have recently been introduced (e.g., Microsoft Azure Digital Twins[1] and Eclipse Ditto[2]) to try to effectively connect the physical and the cyber world through DTs and a shared set of "as-a-service" APIs and functionalities. Unfortunately, in such references and also within some of the more recent research activities (Autiosalo et al., 2021; Eramo et al., 2021) DTs have been mainly exploited as data structure repositories, rather than active software components with an internal behaviour and a set of core responsibilities to handle the interaction with the physical world. The responsibility to communicate with the physical counterpart to collect data and send commands is often delegated to external applications and typically through platform-specific services (e.g., APIs or SDKs) instead of being a specific core responsibility of each DT. This aspect impacts the digitisation process, that in state-of-the-art platforms results somewhat fragmented across several modules and strongly limits the envisioned and desired autonomy of DTs (Hribernik et al., 2021).

This challenging evolution calls for new software architectures and approaches featuring levels of flexibility beyond the ones provided by the solutions available in current state-of-the-art approaches. As clearly reviewed and pointed out also in Minerva et al. (2020), the literature is conceptually aligned on the idea of adopting DTs in multiple fields but each model is still built from scratch without common methodologies and with the concrete risk to generate a strong vendor lock-in. This trend might prevent the creation of a real open ecosystem where PAs, DTs and applications can cooperate through an effective service continuum: a setting that would instead be crucial for the development of smart factories.

In this context, with the aim to integrate DTs with cognitive, intelligent and analytical solutions, some works in the literature propose the adoption of semantic models and technologies to extract knowledge from data in order to enable twins to autonomously perform some intelligent tasks within the context of the PA (Rozanec et al., 2020; Eirinakis et al., 2020; Abburu et al., 2020). Furthermore, DTs become interesting for a plethora of specific approaches related to analytics (Riemer, 2018), behavioural modelling (Sleuters et al., 2019) or ontology definition (Steinmetz et al., 2018), basically acting as a support for higher-level data processing stages. In this vein, the novelty of our approach lies in the introduction of a hierarchy of abstractions for manufacturing operations, enabled by DTs, that can be exploited to learn causal models of the environment. The possibility of building hierarchical models for DTs has been already mentioned in a few works of the literature (Wang et al., 2020a; Shangguan et al., 2019), although

[1] https://azure.microsoft.com/en-us/products/digital-twins/
[2] https://www.eclipse.org/ditto/

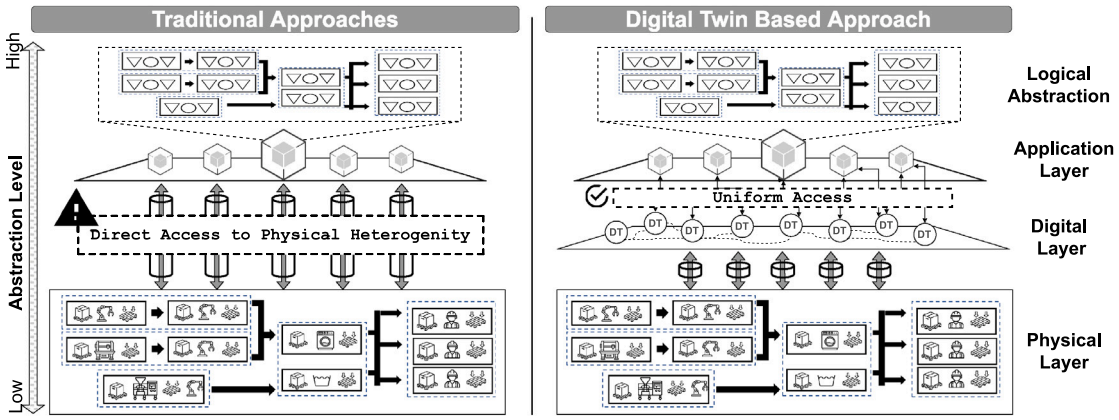

Fig. 1. The schematic representation of the gap between an envisioned logical abstraction functional to intelligent applications and the complexity of directly managing the physical layer and its devices.

neither applied to the context of manufacturing in smart factories, nor to causal model learning.

Within smart factories, DTs have already been envisioned as a key brick of shop-floor environments, due to the intrinsic heterogeneity of the involved devices, and the dynamicity of the processes. For example, (Tao and Zhang, 2017) describes the role that DTs can have in a smart factory: they collect data of existing equipment and share them using the Internet as a communication mean, in order to analyse them and implement realistic planning previsions that take into account customer orders, order priority and work centres average performance. The paper also envisions the use of DTs as a support for planning validation, in order to have affordable planning outputs. In Wang et al. (2020b), a DT is proposed as a mirror of the shop-floor, with a case study that considers a production line of a large-size product, whose output buffer is released only after the work-in-process material is processed. A proactive material handling method is proposed, which also exploits a forecasting module to anticipate the logistic service call. Another proposal from Friederich et al. (2022) proposes a data-driven approach to infer the shop-floor composition from the data generated throughout the production. The data extracted from the shop-floor logs are collected by DTs and analysed via process mining techniques to build a Petri Net, that can be used for further simulation and analysis. Applications of this kind are nowadays more and more necessary, due to the challenge of increasing demands, customisation, and shorter product life-cycles, which generates a continuous evolution in shop-floors. In Zhang et al. (2022), a multi-scale model that also exploits DTs and comprises several layers of a shop-floor architecture is proposed, yet without a direct modelling of the hierarchical levels inside the twins, and without any application to any learning task.

Finally, several applications of causal models can be found in the literature of manufacturing operations: the reader may refer to a recent survey for an overview of the field (Vuković and Thalmann, 2022). The majority of existing approaches focus on fault detection, root cause analysis, and on the use of causality to open the "black-box" of machine learning systems typically used for related tasks. Nevertheless, what is missing in this research line is the use of DTs to cope with the heterogeneous nature of physical devices involved in the manufacturing operations. The two aspects of PA modelling and causality learning are usually independently addressed. We believe that our proposed architecture can fill this gap, by enabling many pervasive intelligent applications in the manufacturing area.

3. A hierarchical architecture for Digital Twins in smart factories

Following recent research trends and analysis (Hribernik et al., 2021; Bellavista et al., 2021), we foster the idea that the envisioned decoupling between the physical layers, digital applications and their logical abstractions can be exploited by embracing DT. We argue that the adoption of DTs at different architectural layers can enable a native interoperability of systems and sub-systems in a smart factory, enhancing scalability, adaptation and coordination, promoting distributed autonomy and learning.

Specifically, we envision multiple digital abstraction layers responsible to effectively decouple the complexity of the physical layers from intelligent services. Such a hierarchical architecture has the induces a logical abstraction able to: (i) represent industrial assets without the effort of directly handling their physical complexity (in terms of communication protocols, data formats and interaction patterns) and enable a native interoperability; (ii) make the process of data collection, pre-processing, and formatting homogeneous and standardised; (iii) augment raw input signals to create more variables to be used by the learning process; and (iv) enable a first level of data analysis and decision making via learning algorithms, possibly to be used in a distributed learning.

Fig. 1 schematically compares traditional approaches, characterised by a direct link between digital services and the physical heterogeneity, with the hierarchical abstraction enabled by DTs. On the one hand (left side), the physical fragmentation is managed through vertical solutions characterised by a low level of interoperability and the consequent inability to effectively hide the underlying heterogeneity for the upper layers. On the other hand (right side), the adoption of DTs and the exploitation of their core capability to digitise and handle their physical counterparts opens to the possibility to build a new digital layer in charge of managing the fragmentation characterise the PAs and to expose a uniform digital abstraction to the application layer.

3.1. The role of Digital Twins

A DT represents the *digitised software replica* of a PA with the responsibility to clone available resources and functionalities and to extend existing behaviours with new capabilities. For example, with respect to an industrial robot a DT can mirror joints position and sensors telemetry; simplify the control of tasks and missions execution through dedicated exposed interfaces; augment original capabilities introducing anomaly detection functionalities to anticipate potential malfunctioning. DT modelling has been recently re-analysed and improved in the scientific literature (Minerva et al., 2020; Minerva and Crespi, 2021; Saracco, 2019) through the identification and definition of a list of abstract properties that should characterise their design and development allowing to build a unified conceptual framework without any lock-in to specific application domains or custom implementations. Each DT is uniquely identified and directly associated to

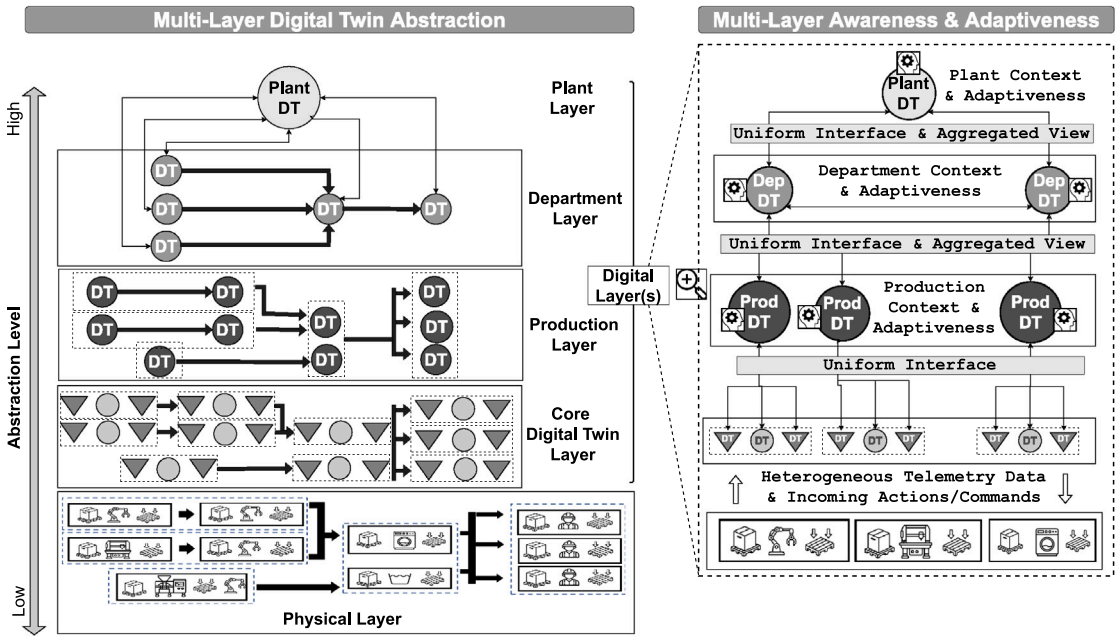

Fig. 2. The digitisation of physical assets brings a new flexibility in the architecture and multiple hierarchical layers can be easily created to aggregate assets and propose different level of observability to the industrial ecosystem and consequently multiple learning and decision point of views.

its physical counterpart and it is in charge of representing it as much as possible in terms of attributes (e.g. telemetry data, configurations, etc.), behaviours (e.g. actions that can be performed by the physical device or on it by external entities) and relationships (e.g. a link between two assets operating in the same logical space, or two subparts of the same device). The *representativeness* should be supported by a model defining how the DT maps the physical world with respect to a target context in which to operate (denoted as *contextualisation* property). In case the usage scenario of the DT is a specific environment (e.g., a production line), most likely only a subset of all the features, properties and information of the physical object are relevant to qualify the twin in the target digital world. Multiple representations and DT instances of the same PA can be defined in the same environment focusing on mapping different physical aspects in relationship to the context and application goals. For example, a robot manipulator can be digitised at the same time by two distinct DT instances, one focusing only on robot joints coordinates to build a synchronised 3D replica and the other one responsible to monitor tasks execution, collect sensor telemetry data and detect anomalies. The physical and digital counterparts mutually cooperate through a *bidirectional synchronisation* (aka shadowing, mirroring) meant to support the original capabilities of the mirrored device, while enabling and augmenting features and functionalities directly on the digital replica, both for monitoring and *control*. DTs consequently allow external services to design new cyber–physical behaviours and to execute high level policies without directly handling the complexity of end devices.

Nevertheless, in general PAs can be seen as groupings of sub-parts (e.g., a robot with multiple joints and a gripper) or as the combination of several individual entities operating together within the same environment (e.g., a production line composed by multiple interconnected machines). The *composability* property of DTs represents the capability to abstract the complexity of a large system and to focus on a few relevant properties and behaviours without having to consider the functioning of all the aggregated sub components. Through composability DTs can be in charge of abstracting a complex environment exposing only relevant information and functionalities to the application layer.

3.2. Hierarchical abstraction & Digitisation in smart factories

A major challenge in applying DTs to the context of smart factories is associated not only to the digitisation of a single physical asset but also to the mapping of complex and hierarchical environments like production lines, departments and plants characterised by a plethora of devices, relationships and data. The adoption of DTs can improve application awareness by introducing a structured and uniform mapping of the physical layer.

Exploiting the *representativeness* of DTs we can model the complexity of a smart factory through the creation of a set of hierarchical, incremental and connected abstraction layers using multiple DT categories. Each of them will be responsible to retrieve data from the underlying levels, elaborate and enrich information according to the target behaviour, and finally expose them with a specific granularity and a uniform interface to the upper layers. This approach allows to: (i) decouple responsibilities because each DT handles a small group of connected components; (ii) augment distributed awareness since each twin focuses on a specific context and abstraction goal; and (iii) enable interoperability through an uniform way to expose information across layers.

Furthermore, the *composability* can be also adopted within the proposed vision to model twins as the combination of multiple individual physical objects or other connected DTs according to the hierarchical level of interest. This possibility is not only useful from an operational point of view of connecting and aggregating multiple entities together but it can be exploited to support discoverability and awareness within the smart factory. Such a mechanism allows observers and applications to navigate the graph of DTs to find specific resources, functionalities or data with a limited prior knowledge.

Fig. 2 illustrates this multi-layer vision enabled through DTs where each abstraction level is on one hand able to build and operate on its local context with the data coming the uniform interface of the previous layer (e.g., the Production Layer with the information coming from core DTs) and on the other hand responsible to expose enriched data and capabilities to observers and consumers of the higher chain of abstraction (e.g., providing data to the Department Layer and its DTs). In the lower layer (the physical layer), the physical world is "sensed" from the distributed sensors and, in turn, affected via distributed actuators. Distributed sensors and actuators are grouped in machines or general equipment. Composed machines and equipment represent *transformation nodes* and *input or output buffers* in the *Core Digital Twin Layer*. Data collected in this layer represents the state

of associated nodes, like buffers levels, time needed for filling and emptying operations, state of the transformation node, etc. Aggregation of related buffers and transformation nodes forms the production node in the production layer. Here KPIs and parameters associated to the production node are tracked, like the relative OEE (Nakajima, 1995), production stop times and rates and relative reasons (like machine or equipment breakdowns, node starving or sating). Going up in the abstraction, aggregation of different production nodes forms departments nodes in the department layer, tracking their state and performance leveraging data coming from the underlying layer. Aggregation of different departments, warehouses and other operation functions (like maintenance or logistics) forms the plant node DT, the highest level of representation from an operations point of view.

As previously anticipated, the benefits of the proposed approach with respect to traditional solutions can be summarised as follows: (i) *Without DTs*: applications and services are directly exposed to the heterogeneity of the physical layer with the concrete risk of increasing their complexity, limiting their design and affecting their sensibility to technological lock-in and changes (e.g., service disruption, firmware update, data format variations or machine replacement); and (ii) *Without Hierarchical DTs*: the responsibilities to model and keep synchronised the existing physical relationships and to build a structured logical abstraction of a complex environment (e.g., an industrial plant with departments and production lines with multiple machines) is entirely delegated to fragmented application layer services that instead should be only focused on how to exploit available data and functionalities to build intelligent capabilities according to their behaviours and goals.

3.3. Enabling learning capabilities

In the architecture described so far, DTs can be used as a building block for machine learning activities within the smart factory (Jaensch et al., 2018). This could be the case, for example, of object detection in images or videos captured by on-board digital cameras, activity recognition from signals emitted by sensors placed in the environment, fault detection and time-series forecasting for predictive maintenance, coordination amongst goal-oriented smart devices (Maier et al., 2017). Some of these learning activities, especially low-level tasks (e.g., dealing with signal processing and data analysis) might take place also *within* DTs (Jaensch et al., 2018).

In this work, we are specifically interested in learning causal representations of the world, that is *explainable* models that can be easily interpreted by humans, helping analysis and decision-making processes. These models and frameworks are typically used to represent causality components and events, without entering the details of the physical implementation (Lechevalier et al., 2014). Since physical machines involved in the manufacturing process are usually composed of different devices, such as sensors and actuators, the hierarchical architecture we propose can be exploited to decouple the duties of individual components. Each single device, in fact, can be modelled with a DT, which typically elaborates its own data and provides them to the upper layer in the hierarchy, where they can be aggregated at the machine level, and further processed, thus exploiting representativeness, contextualisation and compositability. Therefore, the hierarchical architecture enabled by DTs naturally induces an abstraction, so that variables and links in a causal model can be associated to high-level concepts (see Fig. 2). Upper layers in the hierarchy have a wider view of the events happening within the considered scenario, and thus can capture dependencies between a larger number of variables: for example, the department layer can take into account the relations amongst all its production nodes. In the next section we will describe how the learning process can effectively take place.

4. Interpretable model learning: The role of causality

Nowadays, one of the challenges of AI is to understand causal relations between the variables of a complex system (Schölkopf et al., 2021), in order to allow the interpretation of choices and decisions of intelligent systems by humans, and to improve the robustness of generalisation skills of smart devices across different situations.[3] In an industrial setting, typically, some of the variables within the environment can be modified by the actions of the devices, while others are clearly out of their control, and can only be observed by agents. In this work, we follow the ideas of Judea Pearl (Pearl, 2019) who introduced the concept of "ladder of causation", defining a hierarchy of causality levels that can be developed by an intelligent agent. The first level of the ladder consists in simply detecting relations as associations (correlations), whereas the second one assumes the possibility to intervene in the environment and observe the (causal) effects of the actions taken. Finally, the third level enables reasoning and planning on the basis of counterfactual analysis. As explained in the next sections, in this paper we will exploit Pearl's ideas of intervention and do-calculus to learn causal dependencies directly from data observations.

4.1. Bayesian networks

Bayesian networks (BNs) are one of the most widely used frameworks to represent interpretable models of the world (Jensen and Nielsen, 2007). A BN is a directed graph where nodes represent variables, and edges represent dependencies amongst them. Two variables that are not connected by an edge are conditionally independent one from the other. Each node (variable) is associated with a conditional probability table (CPT) which defines the conditional probability distribution of that variable, given the variables which it depends upon (i.e., its parent nodes). In the context of a smart factory, nodes can represent state variables of devices, i.e., the last signal emitted by a sensor, the fact that a bearing is currently faulty, the level of a production buffer. Edges represent relations that naturally occur between variables whose value affects each other: a faulty bearing might be responsible of a starving machine, whose buffer will therefore be stuck at the same level for some time.

Three main problems can be characterised when dealing with BNs: inference, parameter learning and structure learning. In case the BN is available (both the structure and the CPT values are known) then it is possible to perform probabilistic inference over it, that is to compute the probability distribution of some of the variables, by giving specific values to others, i.e., observing the chain of dependencies propagating through the graph. In some cases, instead, only the structure of the BN might be known (e.g., designed by a domain expert) whereas the parameters of the CPTs are to be learned from data observations. This is typically done via maximum likelihood estimation from a training set (Jensen and Nielsen, 2007). Finally, a much more challenging scenario is the one where not even the structure of the BN is available, but it should also be learned from data. In this paper, we are particularly interested in this latter task.

4.2. Interventions and do-calculus

Although they do not explicitly model *causal* links, but only conditional dependencies amongst variables, BNs are still widely used as a building block of causal models. In fact, causality can be inferred with *interventions* in the environment, i.e., via do-calculus (Pearl and Mackenzie, 2018). Specifically, given a direct link $X \rightarrow Y$ between two variables X and Y, one can say that X is a cause for Y if the probability of Y is affected by a direct intervention on X. By *imposing* a specific

[3] In a sense, this can be seen as an instantiation of eXplainable AI (XAI) (Arrieta et al., 2020; Chou et al., 2022).

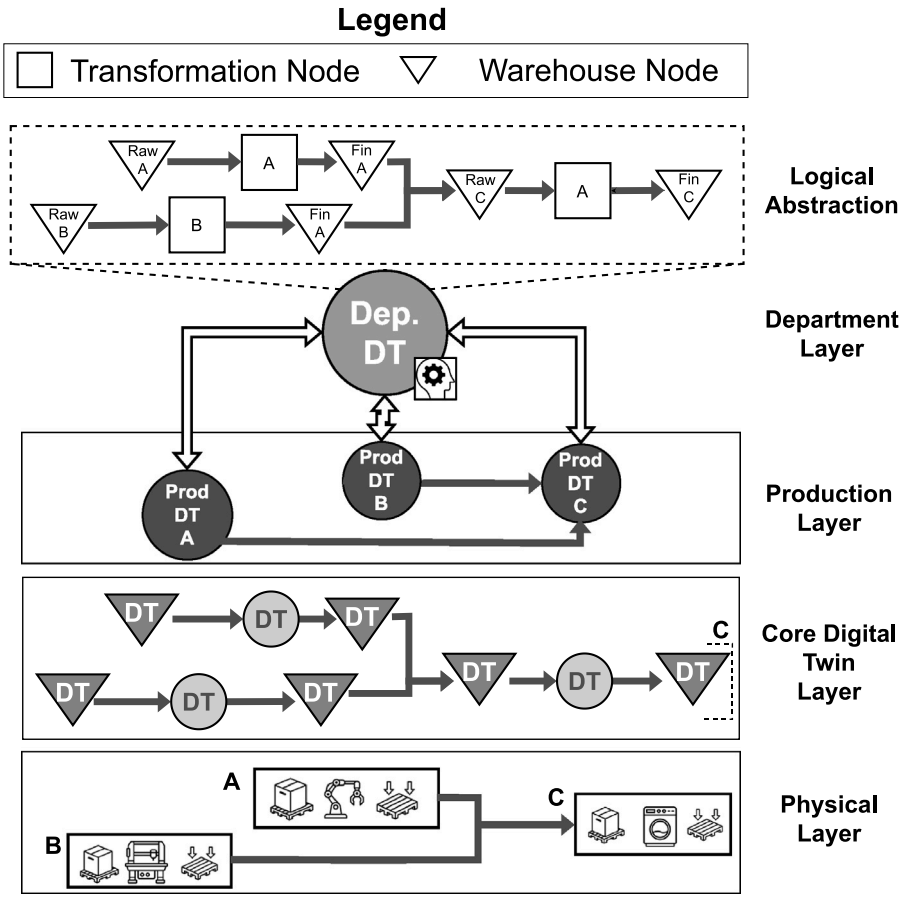

Fig. 3. Logical representation of the simulated production system.

value for X, one can observe the changes in the probability distribution of Y, and deduce the presence of a causal relation accordingly. A causal relation can be assumed to hold between X and Y if $P(Y|do(X))$ is significantly different from $P(Y)$, where the do notation is used to indicate that we are forcing a specific value assumed by X. In our application domain, this mechanism can be exploited to infer that a fault occurring in a given component is indeed the *cause* of some unexpected behaviour in the downstream of the manufacturing process or that, conversely, an increment in the production is due to certain situations and conditions occurring in the workflow.

5. Experiments

Fault detection and root cause analysis are classic applications of causal discovery in smart factories (Vuković and Thalmann, 2022). Lacking a full exploration of causal chains in an industrial environment, in fact, can easily lead to sub-optimal decisions that fail in solving the issue and have consequences on the whole production (Vo et al., 2020). This is especially true in manufacturing environments, where the output carried out by an activity in a certain workstation strictly depends on what happened in the previous steps of the supply chain. With increasingly faster production times, quickly understanding the origin of a certain problem can indeed shorten the root cause analysis time-span. In case the chain of causality and its propagation paths through the shop-floor are known, activities can be dynamically re-scheduled in all the machines that have been hit by the upstream breakdowns (Li et al., 2023).

In order to test our proposed architecture, we simulated a manufacturing environment. The whole environment has been represented via a high-level DT embedding key characteristics of the involved machines and tracking the evolution of its happenings. The goal of the experiment is twofold: (1) to learn a causal model of the manufacturing operations directly from data observations stored in the machine logs produced by all the DTs involved in the process; (2) to evaluate the impact of the use of DTs on reducing the physical complexity.

5.1. Production system model: description and terminology

Each production system can be considered as a system composed of different sub-systems. Essential elements are: an input warehouse, a shop-floor where the input material is transformed somehow, and an output warehouse. The same pattern can be recognised in the shop-floor: each production node is composed of an input buffer node, a processing node (which can be a transforming or an assembly stage), and an output buffer node. Each of them is in turn composed of different elements (which can be other buffers, many sub-processing stages and so on), breaking down the system into physical equipment. Therefore, we can assume that a certain aggregation of physical equipment composes the input buffer node, the processing node and the output buffer node. An aggregation of input buffer, processing and output buffer node composes a production node. One or many production nodes form the shop-floor. An aggregation of production nodes all aimed at reaching the same target composes a department. An aggregation of different departments composes a business unit. An aggregation of different business units composes a production plant.

The simulated scenario consists in three production nodes, named A, B and C, associated to a number of physical devices (see Fig. 3, top and bottom). Following the aforementioned pattern, each production node x is composed by three elements: (i) an input buffer node, where raw material is stored, with capacity IC_x; (ii) a processing buffer node, where raw material is processed; (iii) an output buffer node with capacity OC_x, where finished material is stored. Each of these

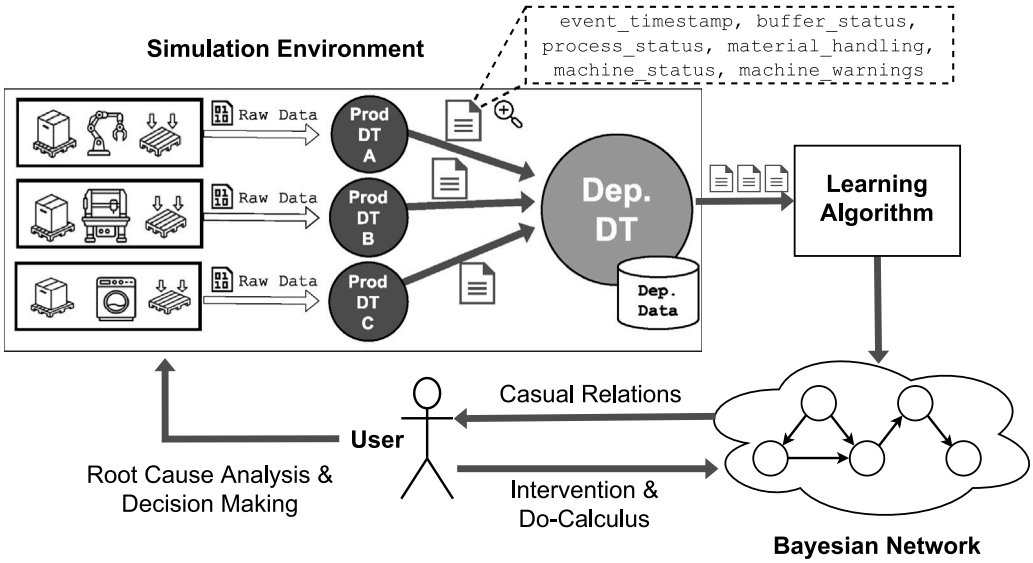

Fig. 4. Workflow of the experimental setting. The production nodes, using DTs, generate simulated data in the form of log files. That data is used to learn a Bayesian network, upon which users can perform intervention and do-calculus to infer causal relations. This information can be exploited for root cause analysis and decision making.

components is associated to a specific DT. When the level of input buffers for A and B is below a certain threshold, the buffers are refilled.[4] The input material picking process has been modelled taking into consideration the material handling time H_x (for machine x) which is also collected by the DT. The output material disposal process follows the same behaviour of the input process. Nodes A and B produce semi-finished parts which are inputs for node C (e.g., nodes A and B produce a gear and a shaft, respectively, using a computerised numeric control machine, whereas node C assembles the gear on the shaft). Node C can produce if and only if both input pieces are available.

During the operation phase, a production node breakdown may occur. In that case, the whole node operations are stopped until the node is restored, that is after the needed repairing time. As mentioned before, node C can produce only if its input buffer is filled with at least one piece of material coming from both A and B. When the input buffer for C is populated, then C operations starts. If no material is found in the input buffer, the associated operation stage does not start. If the output buffer is full, the operation stage stops and waits until some piece is removed.

Fig. 4 show our experimental workflow. The production nodes generate simulated data through the use of DTs, and contribute to a data set in the form of plain log files. This data is used to learn a Bayesian network, which users can interact with in order to perform intervention and do-calculus, and infer causal links to perform root cause analysis and decision making.

5.2. Simulation setting

We implemented our case study in Python, with the SimPy library.[5] All the code and data used to reproduce our experiments are publicly available on a Github repository.[6] As already said, we assume data to be stored in the form of a log file, continuously updated by the DTs during the simulation. The stored information includes:

1. the current timestamp;
2. the levels of input/output buffers for each machine;
3. the number of pieces totally produced until that moment;
4. a Boolean flag F_x to indicate an occurring failure for machine x;
5. a Boolean flag Q_x indicating whether the output buffer of a machine has recently received at least one piece (i.e., to detect whether the machine is currently producing).

Table 1
Summary of parameters considered in the simulation.

Parameter	Description
T	Number of simulation seconds
MTTF_x	Mean Time to Failure for machine x
MTTR_x	Mean Time to Repair for machine x
H_x	Material handling time for machine x
μ_x^P, σ_x^P	Processing time (mean and std) for machine x
IC_x	Input buffer capacity for machine x
OC_x	Output buffer capacity for machine x
Δ	Tolerance threshold for considering machine C as starving

The last two pieces of information will be particularly relevant for our case study: our aim will be to identify causal relations between the failure of a specific machine and the missing production for that machine, as well as for others. For example, in our scenario machine C can be in an unproductive state even without being in a faulty state, in case of failure for either machine A or machine B. This information is obtained with a simple control: considering machine x, if after a time equal to the mean processing time plus a tolerance Δ, a new piece is not placed in the output buffer of x, then flag Q_x is raised. The flag is reset to zero as soon as a new piece is placed in the output buffer. This control over flag Q_x is performed by each DT supervising the corresponding node in the production model.

Table 1 summarises the most important parameters considered within the simulation. We identified a reference setting to obtain a solid and realistic behaviour for the simulated scenario and to highlight the potential of the learning mechanism. Thus, there are no bottlenecks in the production process, all machines have similar production rates, the frequency of breakdowns has been chosen so as to produce a sufficient number of examples to train the system in a limited amount of time. In the future, we plan to conduct a thorough sensitivity analysis of the most significant parameters.

The processing time is modelled as a time delay with a stochastic behaviour following a normal distribution: for machines A and B we

[4] The details of this refilling process have not been considered as they are not relevant for the simulation purpose.

[5] https://simpy.readthedocs.io/

[6] https://github.com/AgentGroup/Enabling-Causality-Learning-in-Smart-Factories-with-Hierarchical-Digital-Twins

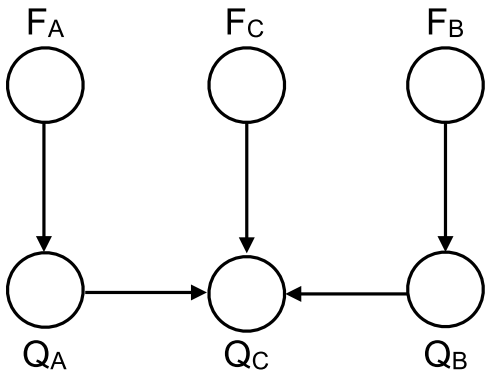

Fig. 5. Causal network learned from simulation data. F_A, F_B and F_C are the random variables representing machine faults. Q_A, Q_B and Q_C represent the three Boolean flags indicating whether machines are currently producing.

set $\mu_x^P = 250$ and $\sigma_x^P = 15$; for C we set $\mu_C^P = 230$ and $\sigma_C^P = 10$. For the sake of simplicity, we considered the material handling time H_x as a deterministic parameter, modelling an automated material handling. Breakdowns are characterised by randomness, thus are represented by random variables following an exponential distribution. For nodes A and B the MTTF is 77,760 and 86,400 s, respectively, whereas the MTTR was set to 10,800 s and 12,000 s, respectively. For node C, we set the MTTF equal to 100,800 s and the MTTR equal to 1,920 s. The input and output buffer capacities IC_x and OC_x were set to 500 pieces. The tolerance threshold used to raise flag Q_C and to indicate that machine C is starving has been set to $\Delta = 243$ seconds (the average of the production rates of the three machines).

Data is generated by simulating 36 working days with a single working shift per day, for a total of $T = 1,250,823$ seconds. After simulation, data is stored into a plain text file (comma-separated values) to be used for causality analysis.

5.3. Causality learning

To learn causal models, we used `CausalNex`,[7] a Python library that allows to perform learning and inference in Bayesian networks, and also do-calculus, in order to observe the effect of interventions and deduce causal links between the observed variables. `CausalNex` allows to learn both the structure and the parameters of a Bayesian networks directly from data observations. The default algorithm, which is the one used in our experiments, is *NOTEARS*, an efficient state-of-the-art method that exploits continuous optimisation to avoid combinatorial search (Zheng et al., 2018).

We used a data set containing 1,250,823 examples and six variables: namely, flags F_x and Q_x for the three machines A, B, C. The only hyper-parameter of the learning algorithm is a threshold value ω that is used to prune "close to zero" edges in the causal model. Following the empirical suggestions in the original paper (Zheng et al., 2018), we set $\omega = 0.3$. Since, in our simulation, we did not model any variable (either environmental or internal) that can be the cause of fault for either machine, we can impose an additional constraint in the structure learning algorithm, by imposing F_x nodes to be parent nodes only. The overall data set is collected by the department-level DT (Fig. 3) which has the vision of the whole system and activates the learning process.

The learned structure of the Bayesian network is depicted in 5. We can easily see that the model correctly captures the links between the failures of machines A and B and the stop in production for machine C.

How to appropriately evaluate the quality and, especially, the interpretability of such a model is an open problem in XAI. Taking inspiration from the literature in that area, we hereby focus on some properties that are seen as important features for an interpretable model (Lipton, 2018; Marcinkevičs and Vogt, 2020). In particular, we consider *correctness* (Marcinkevičs and Vogt, 2020; Carvalho et al., 2019) since an interpretable model indeed also has to provide correct answers; *trust*, that is closely related to correctness, but which, according to Lipton (2018), aims to consider not only "how often a model is right" but "for which examples it is right"; *causality*, as we focus on the analysis of cause–effect relations. Our experimental setup can be framed within the family of functionally-grounded evaluation methods for interpretability (Doshi-Velez and Kim, 2017), where proxy tasks are exploited, without the need to resort to human judgement to analyse the output provided by the system.

First of all, to validate the correctness of the learned model, we performed a standard classification task on the nodes of the Bayesian network. We split the data set into a training (90%) and a test set (10%), we learned the model parameters on the training set, and we performed classification on the test instances, using each of the non-root variables, in turn, as the prediction target. We measured performance using the Area Under Curve (AUC) metric (Bradley, 1997), obtaining a value of 1.00, 1.00 and 0.98 for Q_A, Q_B and Q_C, respectively.

Then, in order to infer causal relations from the network, we performed two subsequent experiments, exploiting interventions and do-calculus. First, we considered each machine independently, and we analysed the change in the probability distribution for flag Q_x after forcing a failure of the machine itself, thus an intervention on F_x by setting its value to 1. Formally, we compare $P(Q_x|do(F_x = 1))$ with $P(Q_x)$ for $x = \{A, B, C\}$. Results are reported in Table 2. For all the machines we observe an evident change in the distribution after intervention, which is a sign of causal dependency between the variables. For machines A and B, flag Q_x is almost always up in case of machine fault (probability around 97%, last row, which means that both machines typically break for a time longer than the mean production time (including tolerance Δ). This probability is slightly lower for machine C, indicating that machine C stops for a breakdown only in 78% of cases. In all the other cases, the breakdown is too brief for being registered by flag Q_C: in fact, the MTTR for machine C is smaller than that of A and B. Additionally, we evaluate the impact of upstream machines A and B over machine C, by analysing $P(Q_C|do(F_x = 1))$ with $x = \{A, B\}$. Results are reported in Table 3. Even in this case, the change in the distribution is a strong indicator of a causal link. In fact, when either machine A or B fails, machine C is getting to starve, as indicated by the raise in flag Q_C. The propagation is slightly more frequent for machine A (69% against 64%), due to the larger MTTF and MTTR values with respect to machine B.

Finally, in the direction of measuring the trust of the model, and in particular its capability to correctly identify the root cause of an occurring fault, we conduct a further experiment to infer the status F_x of each machine, given the observation of the Q_x signals. This setup resembles a root-cause analysis scenario in the industrial domain. We thus perform inference on the Bayesian network, after setting the values of specific observed nodes. When setting $Q_A = Q_B = 0$ and $Q_C = 1$, we obtain a probability for $F_C = 1$ equal to 0.77, whereas the probabilities of $F_A = 1$ and $F_B = 1$ are below 1%. As a second case, when $Q_A = Q_C = 1$, the probability of $F_A = 1$ raises to 0.99 whereas that of having $F_C = 1$ drops below 2%. Similar considerations hold for machine B in case $Q_B = 1$ instead of $Q_A = 1$. All these situations correspond to a correct identification of the root cause of the fault condition, since the combination of both flags Q_A (or Q_B) and Q_C raised to 1 indicates a failure in the upstream machine, which has the effect of blocking production also for machine C.

We conclude this subsection by highlighting two limitations of our experimental setting. First, the structure learning algorithm needs to set the ω hyper-parameter: in the original article describing the approach (Zheng et al., 2018) a value $\omega = 0.3$ is suggested, but there is no real connection with respect to the application domain.

[7] https://causalnex.readthedocs.io/

Table 2
Analysis of do-calculus over each machine independently. We consider the probability of Q_x being raised in case of fault of the corresponding machine, indicated by F_x.

	$x = A$	$x = B$	$x = C$
$P(Q_x = 0)$	0.846	0.864	0.811
$P(Q_x = 1)$	0.154	0.136	0.189
$P(Q_x = 0 \mid do(F_x = 1))$	0.028	0.033	0.220
$P(Q_x = 1 \mid do(F_x = 1))$	0.972	0.967	0.780

Table 3
Analysis of the impact of upstream machines A and B over C, via do-calculus. We consider the probability of Q_C being raised in case of fault of either A or B, indicated by F_x.

	$x = A$	$x = B$
$P(Q_C = 0)$	0.811	0.811
$P(Q_C = 1)$	0.189	0.189
$P(Q_C = 0 \mid do(F_x = 1))$	0.309	0.363
$P(Q_C = 1 \mid do(F_x = 1))$	0.691	0.637

Secondly, another crucial point is the tolerance Δ that evaluates when a piece is considered missed in the output buffer. This parameter is very important for the system to work, and, ideally, needs to be set in order to model when a piece is missed by a given probability. Such threshold can be seen as a sort of trade-off between false alarms (i.e., situations where no fault is actually in progress, but the production is just slower than usual) and missing detections (i.e., when a faulty scenario is not recognised by the system).

5.4. Impact of Digital Twins on physical complexity

The second goal of our experimental activity is to understand the impact of DTs on reducing the physical and technological complexity of our case study. We estimate the complexity of the scenario with and without their adoption to assess their impact in terms of digitisation and decomposition of responsibilities. To this end, we have defined an indicator, named *Physical Complexity Indicator* (PCI), which measures the perceived complexity associated to any application requiring the interaction with the physical layer (e.g., our causality learning module). PCI takes into account the following criteria:

- *Required Protocols (p)*: the number of application layer protocols required by a digital application or service to interact with the deployed physical assets to collect data and to send commands.
- *Communication Patterns (c)*: the number of communication patterns needed to interact with involved devices and platforms (e.g., Publisher/Subscriber or Request/Response);
- *Data Formats*: the number of different data formats, serialisation and information representation techniques required to read and send data from and to deployed devices;
- *Interaction Points (n)*: the number of different modules, services or platforms that an application should interact with to retrieve all the target data or consume services;
- *Aggregation Points (a)*: the number of aggregation or composition levels required to abstract the physical world into the right level of complexity with respect to the observers' typologies and their application goals (e.g., merging information and telemetry data from machines in the same production line).

Each criterion has been assigned a specific Importance Factor (IF) ranging from 1 (lower) to 3 (higher), to be used as a multiplicative score for the PCI of that criterion, according to its impact on the development, deployment and maintenance of the solution. The PCI is directly proportional to the exposed physical complexity that an external application is in charge to manage in order to communicate with deployed physical assets, collect data and send commands. The $PCI = \sum_{i=1}^{5} criterion_i \times IF_i$ represents the physical complexity level that the external application has to face with respect to the physical use case. In our scenario, we take into account typical configurations coming from the IoT and IIoT technological approaches and characteristics. In order to describe the impact in adopting DTs, 3 example use cases are proposed:

- *Use case 1 — One Production Line with Two Heterogeneous Machines ($p = 1, c = 1, m = 2, n = 2, a = 1$)*. In this scenario, we assume to have 2 different machines using the same communication pattern and a common application layer protocol (e.g., Publisher/Subscriber with MQTT). The two machines generate different data formats associated to their specific characteristics, for example in terms of functionalities and generated telemetry information. From an application perspective, the interaction points are 2 while the aggregation point is 1 in order to digitise the overall production line by merging information of the active devices. The resulting PCI is reported in Table 4, showing that the adoption of DTs brings a reduction of around 50%.
- *Use case 2 — Two Production Lines with four Heterogeneous Machines ($p = 2, c = 2, m = 2, n = 4, a = 2$)*: In this use case we assume to have two production lines, each composed of two machines. The two machines inside a single line are different, but the two composed lines are identical, running in parallel. Machines adopt 2 different protocols and 2 different communication patterns (e.g., Publisher/Subscriber with MQTT and Request/Response with ModBus) and expose a different and customised data format. Each machine represents a single interaction point, whereas the two lines are considered as independent aggregation points for observing and interacting applications. The PCI is reported in Table 4: using DTs it decreases from 26 to 8 (reduction ~70%).
- *Use case 3 — One Department, Three Production Lines and Five Machines ($p = 2, c = 2, m = 5, n = 5, a = 4$):*. This scenario is inspired by the case study proposed in the paper. We consider five different typologies of physical assets deployed through three production lines and one department. A conservative hypothesis is made with the adoption of only two application-layer protocols associated to two different categories of communication patterns such as Publisher/Subscriber and Request/Response (e.g., MQTT and ModBus). In this context, even if we can have common shared protocols, the adopted data formats and the interaction points can be reasonably different and customised for each type of deployed machine through the use of specific configurations and modules. Furthermore, with respect to the required aggregation points, we have to take into account the responsibility to merge information of each production line (A, B and C) and to build a unified view of the entire department. This is a key contribution of our hierarchical architecture. The PCI is reported in Table 4 as well as depicted in Fig. 6, showing that in this scenario the complexity reduction is around 81%.

This analysis shows how the proposed approach and the introduction of multiple hierarchical DTs allow to build an effective uniform digital abstraction reducing the perceived heterogeneity in terms of protocols, communication patterns and data format thanks to the adoption of a shared communication layer and a unique representation of the information. It is important to highlight how, thanks to the introduction of DT layers, the physical complexity perceived by an application willing to interact with machines, production lines or the department is always the same ($PCI = 8$) since the increased physical heterogeneity and the associated management cost is hidden behind DTs. This management is not only associated to low-level technological aspects (such as communication patterns and protocols) but it massively relies also on the possibility to effectively map and model existing physical relationships (e.g., two machines interacting on the same production line), assets composition (e.g., multiple production areas with their machines belonging to the same department) and to properly structure the knowledge in a uniform and exploitable way, reducing interaction points and simplifying aggregation.

Table 4
Estimation of the Physical Complexity Index for the 3 proposed use cases. Without DTs, PCI grows with the number involved devices and protocols. Instead, DTs decouple the physical complexity from their digital counterpart, simplifying the interaction with external applications.

Criterion	IF	Use Case 1		Use Case 2		Use Case 3	
		w/o DT	DT	w/o DT	DT	w/o DT	DT
Required Protocols (p)	2	1	1	2	1	2	1
Communication Patterns (c)	1	1	1	2	1	2	1
Data Formats (m)	3	2	1	2	1	5	1
Interaction Points (n)	2	2	1	4	1	5	1
Aggregation Points (a)	3	1	0	2	0	4	0
PCI	–	16	8	26	8	43	8

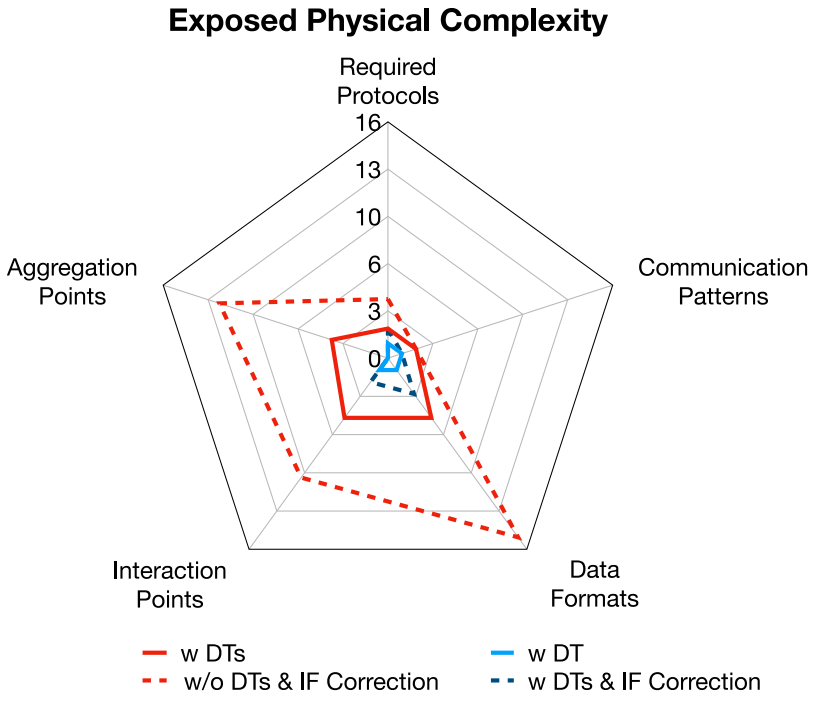

Fig. 6. Graphical representation of the Physical Complexity Index for our scenario, taking into account results with and without the correction of the Importance Factor (IF).

6. Conclusions

One of the main benefits of Industry 4.0 is the ability of collecting information generated in manufacturing environments, such as shop-floors. The challenge stands in managing the huge amount of generated data, to organise it, and extract value for the process stakeholders. DTs are the ideal instrument to address the steps of data collection, organisation, abstraction and analysis, at different operational levels.

Having a system that can understand *why* a machine stops most of the times (whether for its own breakdowns, whether for upstream breakdowns or other potential happenings) can help operations management activities, highlighting what is going on on the shop-floor and the underlined reasons. Actually, human-driven root cause analysis are based on *lean* and *agile manufacturing* tools, applied by technical leads which have to go on the shop-floor, collect data, climb the cause–effect ladder of facts and, maybe, find the root cause of a certain happening. The process usually takes weeks if not months.

In this paper, we presented a hierarchical architecture enabled by DTs, which is functional for the development of learning procedures that can build causal models of the environment directly from data observations collected from the physical devices. We presented a proof-of-concept of this architecture by simulating a manufacturing production process and performing intervention and do-calculus to infer causal links between the observed variables. We also estimated the impact of the use of DTs on the reduction of the physical complexity of the considered scenario. DTs bring abstraction capabilities to the scenario, making it possible to perform learning and inference over the collected data set, without the necessity of knowing the technical details of the physical machine that each single DT represents.

In the future, we plan to extend our work to larger models, which brings up the challenge of scalability of the learning algorithms. In this direction, we aim to exploit distributed learning solutions, where each DT would provide its own view of the environment, and a shared representation would be necessary in order to collectively build a joint model of the overall system.

CRediT authorship contribution statement

Marco Lippi: Conceptualization, Methodology, Supervision, Writing. **Matteo Martinelli:** Methodology, Investigation, Software, Validation, Writing. **Marco Picone:** Methodology, Investigation, Software, Validation, Writing. **Franco Zambonelli:** Methodology, Supervision, Writing.

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.

Data availability

I have shared the link to

Enabling Causality wDTs (Code and data) (Github).

Acknowledgments

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme (GA 952215).

References

Abburu, S., Berre, A.J., Jacoby, M., Roman, D., Stojanovic, L., Stojanovic, N., 2020. Cognitwin – hybrid and cognitive digital twins for the process industry. In: IEEE Int. Conf. on Engineering, Technology and Innovation. ICE/ITMC, pp. 1–8.

Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al., 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion 58, 82–115.

Autiosalo, J., Siegel, J., Tammi, K., 2021. Twinbase: Open-source server software for the digital twin web. IEEE Access 9, 140779–140798.

Bellavista, P., Giannelli, C., Mamei, M., Mendula, M., Picone, M., 2021. Application-driven network-aware digital twin management in industrial edge environments. IEEE Trans. Ind. Inform. 17, 7791–7801.

Botkina, D., Hedlind, M., Olsson, B., Henser, J., Lundholm, T., 2018. Digital twin of a cutting tool. Procedia CIRP 72, 215–218.

Bradley, A.P., 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognit. 30, 1145–1159.

Carvalho, D.V., Pereira, E.M., Cardoso, J.S., 2019. Machine learning interpretability: A survey on methods and metrics. Electronics 8, 832.

Chen, C., Ji, Z., Wang, Y., 2018. Nsga-ii applied to dynamic flexible job shop scheduling problems with machine breakdown. Modern Phys. Lett. B 32.

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

Cimino, C., Negri, E., Fumagalli, L., 2019. Review of digital twin applications in manufacturing. Comput. Ind. 113, 103130.

Members of the Digital Framework Task Group, 2018. White Paper: The Gemini Principles. Technical Report, Centre of Digital Built Britain, Available at https://www.cdbb.cam.ac.uk/DFTG/GeminiPrinciples. (Last Accessed 01 April 2021).

Doshi-Velez, F., Kim, B., 2017. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608.

Eirinakis, P., Kalaboukas, K., Lounis, S., Mourtos, I., Rožanec, J.M., Stojanovic, N., Zois, G., 2020. Enhancing cognition for digital twins. In: 2020 IEEE International Conference on Engineering, Technology and Innovation. ICE/ITMC, pp. 1–7.

ElMaraghy, W., ElMaraghy, H., Tomiyama, T., Monostori, L., 2012. Complexity in engineering design and manufacturing. CIRP Ann. 61, 793–814.

Eramo, R., Bordeleau, F., Combemale, B., van den Brand, M., Wimmer, M., Wortmann, A., 2021. Conceptualizing digital twins. IEEE Softw.

Ferrer, B.R., Mohammed, W.M., Martínez Lastra, J.L., Villalonga, A., Beruvides, G., Castaño, F., Haber, R.E., 2018. Towards the adoption of cyber–physical systems of systems paradigm in smart manufacturing environments. In: 2018 IEEE 16th International Conference on Industrial Informatics. INDIN, pp. 792–799. http://dx.doi.org/10.1109/INDIN.2018.8472061.

Friederich, J., Francis, D.P., Lazarova-Molnar, S., Mohamed, N., 2022. A framework for data-driven digital twins of smart manufacturing. Comput. Ind. 136.

Gao, R.X., Wang, L., Helu, M., Teti, R., 2020. Big data analytics for smart factories of the future. CIRP Ann. 69, 668–692.

Gelernter, D., 1991. Mirror Worlds Or the Day Software Puts the Universe in a Shoebox: How Will It Happen and What It Will Mean. Oxford University Press, Inc., New York, NY, USA.

Hribernik, K., Cabri, G., Mandreoli, F., Mentzas, G., 2021. Autonomous, context-aware, adaptive digital twins—state of the art and roadmap. Comput. Ind. 133, 103508.

Jaensch, F., Csiszar, A., Scheifele, C., Verl, A., 2018. Digital twins of manufacturing systems as a base for machine learning. In: 2018 25th International Conference on Mechatronics and Machine Vision in Practice. M2VIP, IEEE, pp. 1–6.

Jensen, F.V., Nielsen, T.D., 2007. Bayesian Networks and Decision Graphs, Vol. 2. Springer.

Lechevalier, D., Narayanan, A., Rachuri, S., 2014. Towards a domain-specific framework for predictive analytics in manufacturing. In: 2014 IEEE International Conference on Big Data. Big Data, IEEE, pp. 987–995.

Li, Y., Tao, Z., Wang, L., Du, B., Guo, J., Pang, S., 2023. Digital twin-based job shop anomaly detection and dynamic scheduling. Robot. Comput.-Integr. Manuf. 79, 102443.

Lipton, Z.C., 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. Queue 16, 31–57.

Liu, Y., Peng, Y., Wang, B., Yao, S., Liu, Z., 2017. Review on cyber–physical systems. IEEE/CAA J. Autom. Sin. 4, 27–40.

Lou, P., Liu, Q., Zhou, Z., Wang, H., Sun, S.X., 2012. Multi-agent-based proactive-reactive scheduling for a job shop. Int. J. Adv. Manuf. Technol. 59, 311–324.

Maier, A., Schriegel, S., Niggemann, O., 2017. Big data and machine learning for the smart factory—solutions for condition monitoring, diagnosis and optimization. In: Industrial Internet of Things. Springer, pp. 473–485.

Marcinkevičs, R., Vogt, J.E., 2020. Interpretability and explainability: A machine learning zoo mini-tour. arXiv preprint arXiv:2012.01805.

Miguéis, V.L., Borges, J.L., et al., 2022. Automatic root cause analysis in manufacturing: an overview & conceptualization. J. Intell. Manuf. 1–18.

Minerva, R., Crespi, N., 2021. Digital twins: Properties, software frameworks, and application scenarios. IT Prof. 23, 51–55.

Minerva, R., Lee, G.M., Crespi, N., 2020. Digital twin in the IoT context: A survey on technical features, scenarios, and architectural models. Proc. IEEE 108, 1785–1824.

Nakajima, S., 1995. Introduction to TPM: Total Productive Maintenance. Productivity Press.

Pearl, J., 2019. The seven tools of causal inference, with reflections on machine learning. Commun. ACM 62, 54–60.

Pearl, J., Mackenzie, D., 2018. The Book of Why: The New Science of Cause and Effect. Basic Books.

Ricci, A., Croatti, A., Mariani, S., Montagna, S., Picone, M., 2022. Web of digital twins. ACM Trans. Internet Technol. 22 (4), 1–30.

Riemer, D., 2018. Feeding the digital twin: Basics, models and lessons learned from building an iot analytics toolbox (invited talk). In: 2018 IEEE International Conference on Big Data. Big Data, IEEE, p. 4212.

Rozanec, J.M., Jinzhi, L., Kosmerlj, A., Kenda, K., Dimitris, K., Jovanoski, V., Rupnik, J., Karlovcec, M., Fortuna, B., 2020. Towards actionable cognitive digital twins for manufacturing. In: Proceedings of the International Workshop on Semantic Digital Twins Co-Located with the 17th Extended Semantic Web Conference. SeDiT@ESWC 2020, Heraklion, Greece, June 3, 2020, In: CEUR Workshop Proceedings, vol. 2615, CEUR-WS.org.

Salierno, G., Leonardi, L., Cabri, G., 2021. The future of factories: Different trends. Appl. Sci. 11, 9980.

Saracco, R., 2019. Digital twins: Bridging physical space and cyberspace. Computer 52, 58–64.

Schölkopf, B., Locatello, F., Bauer, S., Ke, N.R., Kalchbrenner, N., Goyal, A., Bengio, Y., 2021. Toward causal representation learning. Proc. IEEE.

Schranz, M., Di Caro, G.A., Schmickl, T., Elmenreich, W., Arvin, F., Şekercioğlu, A., Sende, M., 2021. Swarm intelligence and cyber–physical systems: Concepts, challenges and future trends. Swarm Evol. Comput. 60, 100762.

Shangguan, D., Chen, L., Ding, J., 2019. A hierarchical digital twin model framework for dynamic cyber–physical system design. In: Proceedings of the 5th International Conference on Mechatronics and Robotics Engineering. pp. 123–129.

Sleuters, J., Li, Y., Verriet, J., Velikova, M., Doornbos, R., 2019. A digital twin method for automated behavior analysis of large-scale distributed iot systems. In: 2019 14th Annual Conference System of Systems Engineering. SoSE, IEEE, pp. 7–12.

Steinmetz, C., Rettberg, A., Ribeiro, F.G.C., Schroeder, G., Pereira, C.E., 2018. Internet of things ontology for digital twin in cyber physical systems. In: 2018 VIII Brazilian Symposium on Computing Systems Engineering. SBESC, IEEE, pp. 154–159.

Tao, F., Qi, Q., 2019. Make more digital twins. Nature 573, 490–491.

Tao, F., Zhang, M., 2017. Digital twin shop-floor: A new shop-floor paradigm towards smart manufacturing. IEEE Access 5, 20418–20427.

Tao, F., Zhang, H., Liu, A., Nee, A.Y.C., 2019a. Digital twin in industry: State-of-the-art. IEEE Trans. Ind. Inform. 15, 2405–2415.

Tao, F., Zhang, M., Nee, A., 2019b. Chapter 1 - background and concept of digital twin. In: Tao, F., Zhang, M., Nee, A. (Eds.), Digital Twin Driven Smart Manufacturing. Academic Press, pp. 3–28. http://dx.doi.org/10.1016/B978-0-12-817630-6.00001-1.

Vargas, I.G., Gottardi, T., Braga, R.T.V., 2016. Approaches for integration in system of systems: A systematic review. In: 2016 IEEE/ACM 4th International Workshop on Software Engineering for Systems-of-Systems. SESoS, pp. 32–38. http://dx.doi.org/10.1109/SESoS.2016.014.

Vo, B., Kongar, E., Suárez-Barraza, M.F., 2020. Root-cause problem solving in an industry 4.0 context. IEEE Eng. Manag. Rev. 48, 48–56.

Vuković, M., Thalmann, S., 2022. Causal discovery in manufacturing: A structured literature review. J. Manuf. Mater. Process. 6, 10.

Wang, Y., Wang, S., Yang, B., Zhu, L., Liu, F., 2020a. Big data driven hierarchical digital twin predictive remanufacturing paradigm: Architecture, control mechanism, application scenario and benefits. J. Clean. Prod. 248, 119299.

Wang, W., Zhang, Y., Zhong, R., 2020b. A proactive material handling method for cps enabled shop-floor. Robot. Comput.-Integr. Manuf. 61.

Wiendahl, H.P., Scholtissek, P., 1994. Management and control of complexity in manufacturing. CIRP Ann. - Manuf. Technol. 43, 533–540.

Zhang, H., Qi, Q., Tao, F., 2022. A multi-scale modeling method for digital twin shop-floor. J. Manuf. Syst. 62, 417–428.

Zheng, X., Aragam, B., Ravikumar, P.K., Xing, E.P., 2018. Dags with no tears: Continuous optimization for structure learning. Adv. Neural Inf. Process. Syst. 31.