The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-024-14508-5

ORIGINAL ARTICLE

Towards Frugal Industrial AI: a framework for the development of scalable and robust machine learning models in the shop floor

Giovanna Martínez-Arellano[1] · Svetan Ratchev[1]

Received: 22 August 2024 / Accepted: 16 September 2024
© The Author(s) 2024

Abstract
Artificial intelligence (AI) among other digital technologies promise to deliver the next level of process efficiency of manufacturing systems. Although these solutions such as machine learning (ML) based condition monitoring and quality inspection are becoming popular, these work under very limited conditions. Solutions do not scale-up in the real environment, where there is a mix of manufacturing equipment, where the quality and quantity of data available changes from machine to machine, or where the process changes, changing the distribution of data (i.e. concept drift). This is particularly challenging in highly reconfigurable and flexible environments. Having to develop machine learning models from scratch every single time is not cost-effective, time-consuming, requires expert knowledge that is typically not available in the manufacturing environment as well as can be challenging when data is not available in high volumes. Model robustness, reusability, adaptability and life cycle management are the keys to scale-up this technology in the manufacturing industry. In this work, a conceptual framework to enable simple and robust ML model development for the shop floor is introduced. Referred here as *Frugal Industrial AI*, the approach takes advantage of existing models and their context to build more robust ones in a data-efficient manner. Using a semantic knowledge base of how to construct these models for different manufacturing applications and semi-automating the development or reuse of solutions through semantic similarity, it is demonstrated how models can be developed in a more streamlined way. In addition, it is demonstrated how capturing process context information is important for the effective reuse of existing models through continual learning. This is key to building more robust ML solutions that can deal with real changing manufacturing environments, avoiding retraining from scratch as well as enabling the non-expert to use AI effectively on the shop floor.

Keywords Frugal machine learning · Flexible manufacturing · Continual learning · Semantic modelling · MLOps

1 Introduction

Rapid changes in customer demand as well as black swan events such as the COVID-19 pandemic are pushing the limits of current manufacturing systems. Companies are looking at different ways of realising flexibility in their manufacturing processes to remain competitive. Although Reconfigurable Manufacturing Systems (RMS) offer the capability to efficiently and quickly adapt to rapid changes [1], there is very little adoption of these until today [2]. On one hand, this is due to the complexity it involves moving toward a more modularised design of systems so that modules can be easily added, removed, replaced or repurposed. On the other hand, there is the complexity of operational decision-making. For example, maintenance decision-making in a typical production line is very complicated; however, it is even more complicated when the manufacturing system is changing [1].

Since the introduction of the Industry 4.0 initiative by Germany, there has been a strong interest on the digitalisation of factories through technologies such as cyber-physical systems, smart sensors, data analytics and AI. Cyber-physical systems such as cooperative industrial robots that are highly integrated with sensors, are some of the technologies that offer the modularity that is required for RMS [3]. Reconfigurable machine tools are also being proposed to enable different machining processes within the same equipment [4]. Data analytics and AI offer the possibility for developing

✉ Giovanna Martínez-Arellano
giovanna.martinezarellano@nottingham.ac.uk

Svetan Ratchev
svetan.ratchev@nottingham.ac.uk

[1] Institute for Advanced Manufacturing, University of Nottingham, Derby Road, Nottingham NG7 2GX, NG, UK

Published online: 01 October 2024

more *intelligent* decision-making algorithms in applications such as condition monitoring, quality monitoring, and overall process improvement [5], as well approaches for developing more adaptive robot control such as reinforcement learning, imitation learning and computer-vision based control [6]. Some authors start to explore architectures for integrating reconfigurability together with AI in the design and control of advanced production systems [7]. However, generally, AI models are trained on very specific conditions and on a reconfigurable environment, these may degrade their performance very quickly. Diverse conditions, in the context of computer-vision AI systems for example, such as lighting conditions, a field of view distance and a variety of object types, can considerably decrease the precision of the system [6]. In condition monitoring, operation change leads to evident concept drift which challenges the AI system to remain robust [8]. *Continually* adapting the AI when the physical world adapts becomes a challenge, especially when the industry lacks the digital skills to be maintaining, re-training and re-deploying models.

Despite AI being promised to become one of the most disruptive technologies in the manufacturing sector [9] and its success from a research point of view, there are very few real use cases in the literature, being most of them at the pilot stage [10]. There are several factors that prevent this technology to scale-up in the shop floor. Firstly, AI solutions are developed by a data science expert and so pipeline development, monitoring and re-deployment of these solutions its a skill that is currently missing in the manufacturing industry. Second, these solutions are developed for very specific and constrained scenarios that are typically not representative of the complexity of the real environment. Third, there is currently no standard practice on how to manage the lifecycle of ML solutions (e.g. MLOPs) and to link these solutions to the manufacturing environment which is a core element for scalability. Fourth, they typically lack of adaptation which is necessary for an environment as complex as the shop floor. Techniques such as continual learning that allow to incrementally learn and develop more robust models in the long term are not exhaustively explored yet in the literature. For AI to scale in the manufacturing shop floor, there needs to be a move toward more simplified, less data-demanding models that can adapt in an *incremental* fashion to changes, as more often than not, a "complete" data set will not be available. Finally, it is difficult to reuse such solutions when data changes from machine to machine and neither process nor model development knowledge is captured to speed up developments. For operators to easily create and reuse solutions there needs to be a connection between process, data and AI solution to facilitate this.

In this context, this paper presents a novel framework for facilitating the development of robust machine-learning industrial solutions by the non-expert. At the core of the framework is a semantic model, which allows to capture both manufacturing and ML model development knowledge, linking the two. By capturing these interrelationships, it is possible to identify similar cases in which a solution exists, but also understand how it was developed and which data it consumes. Inferences can be made with regard to the potential of the model for using continual learning techniques or transfer learning which is required for robustness, or for reusing some of its pipeline steps. To identify similar cases, a similarity metric is proposed. The paper demonstrates the importance of identifying similar scenarios for the success of existing continual learning techniques and presents the different routes that are possible for creating a new model from existing data and models. In summary, the key contributions of this paper are the following:

1. A semantic model that unifies existing process and asset semantic models with new Industrial AI models that will capture the AI life cycle and how it relates to the manufacturing process and environment.
2. A methodology for matching problems to ML solutions and their pipelines which can be instantiated and orchestrated to easily reuse an approach as well as to find existing ML models based on process and environment similarities.
3. A demonstration of how capturing and linking manufacturing system and process to AI models supports a better selection for developing more robust and less data-exhaustive models that can deal with concept drift through continual learning strategies.

The rest of the paper is organised as follows: Sect. 2 presents the problem context in detail and the latest existing work on facilitating the development and operationalisation of industrial AI solutions. Section 3 introduces the framework and the different components that are used to facilitate machine learning model development and reuse. Section 4 presents a scenario to demonstrate the use of the approach, with particular focus on the use of the semantic model and how it can support continual learning for AI model robustness, and finally conclusions and future work are presented in Sect. 5.

2 Context and related work

Across the board, it has been demonstrated how AI, and Machine Learning techniques in particular, can be successfully applied in manufacturing contexts and learn very well patterns given a good quality dataset. For example, there is a vast amount of work done in tool wear prediction and classification using data sets such as the 2010 PHM data set [11, 12] and NASA milling data set [13], which captures different machining conditions varying the material type and cutting

parameters. However, even though very accurate models can be created with these datasets, the scenarios captured in these are still quite limited when looking at the real environment. Being machining such an essential operation, models that can be easily transferred to other conditions would be desirable, or methods that can work with limited data.

There are four key challenges for machine learning-based industrial solutions to scale-up and be robust to the changing manufacturing environment:

1. Isolated learning — monitoring assets (e.g. energy consumption, vibrations, forces) typically relies on the assumption that the process running in the equipment is repetitive. In a RMS where an asset may either perform a different capability, may be handling a different material type, or working with a new set of parameters, the patterns expected in the data may drift (i.e. *concept drift*). Existing off-the-shelf techniques cannot handle this drift, and this is reflected on an eventual decrease in accuracy [14]. Off-the-shelf ML models, particularly those based on neural architectures, cannot continuously learn and remain accurate in old and new data, unless they are trained from scratch (i.e. stateless retraining). This is due to a well-known problem of techniques such as deep learning called catastrophic forgetting, where the trained model loses accuracy on previously trained tasks as it is trained with only new data [15, 16]. In the context of RMS, a single asset may be performing different tasks or encountering unseen conditions whilst performing the same task for which stateless retraining is not ideal. There is a need for more *stateful training* (training with mostly new data without losing accuracy in old data) approaches.
2. Models developed by ML domain experts — effective ML-based solutions require both the data science expert knowledge on data processing and model selection and parameterisation as well as manufacturing process knowledge to ensure the best ML design decisions are taken. Understanding the manufacturing problem and how it translates into a definition of the machine learning problem (i.e., classification or regression challenge) is key to the success of the solution [17]. Although there are some recent efforts on the use of predefined ML pipelines for creating models, these are still domain-specific [18, 19]. To avoid "re-inventing the wheel" every time a new industrial solution is created, a systematic capture of what works (e.g. sensors, data frequency, features, models) for which application could enable a quicker solution development and potentially model reuse. For machine learning models to scale-up in the shop floor, there needs to be a way to streamline the development or reusability of models so that these can be easily deployed by the manufacturer without having to rely on the data scientist.
3. Data availability — choosing a suitable model and designing its structure depends directly on the amount and diversity of the data available. For example, most research literature comprise datasets of small- and medium-size for machining monitoring with very limited conditions [20]. Deep learning models trained with large image data sets such as ImageNet for example [21], have been successful to some extent in some industrial applications thanks to techniques such as transfer learning. Using this technique, it is possible to distill learnt features and speed up the learning process when presented with new data. This is particularly useful when there is limited data available. However, there is still a lack of large models that could be easily reused across multiple manufacturing problems. In one hand, this is due the lack of large manufacturing data sets available to create these manufacturing *foundation models* with (e.g. sensor data), but also due to the lack of standards for how companies can systematically capture and safely share data or offer trained models. Some models such as timeGPT start to emerge [22, 23], but applications in the manufacturing context are still limited.
4. ML model operationalisation — changing environments need for models to be monitored, re-trained and re-deployed. To understand when this re-training needs to happen would depend on defining the set of metrics that can allow the monitoring process to detect the drift. These metrics could be related to drifts on the model accuracy, or on the data itself [17]. MLOps practices for industrial AI solutions are starting to emerge in some of the literature, but more work on this area is essential to scaling AI in the shop floor, particularly on safety-critical applications. Further developments of AI in RMS need to look at continuous monitoring and re-redeployment as a key and integral element of AI solutions.

Given this context, the following subsections present some of the latest work that attempts to address some of these challenges.

2.1 Industrial model development pipelines

There have been some recent efforts in the research community on the development of frameworks to facilitate the development and management of industrial models and data. From the model development point of view, there are some recent works on the development of ML pipelines in the manufacturing context. For example, Frye et al. present a methodology for supporting decision-making during four main steps of the development of machine learning models; data preparation, model building, model deploy-

ment and model certification. The approach is based on an rule-based expert system and validated specifically in the model-building step, where hyper-parameters for model training are recommended according to what has worked in similar use cases [24]. Zhou et al. present a framework for the development of ML solutions through four different predefined pipelines [25, 26]. Using semantic models, the framework can identify the most appropriate pipeline to use and so it can automate the development through a predefined set of preprocessing steps which include statistical feature extraction, selection, and training two or three predefined models depending on the end application. One of the main limitations, in addition having a limited set of predefined models and architectures, is that it relies on the availability of an initial well-bounded data set expected to be a good representation of all possible future scenarios. In practice, this is almost never the case for dynamic environments. Even when data is coming from the same machine carrying out the same operation (concept drift), and so it is not possible to know a priori the bounds of such data set. In this context, it is not practical to assume data for that machine is available; instead, it is important to know if there is another model from another machine that has been trained in a similar scenario and use that as a starting point or employ techniques that work better with limited data sets before the model can then be updated with a more robust one once more data is available.

2.2 Manufacturing and ML knowledge capture

There have been several works on the capture of manufacturing knowledge to support decision-making in industrial contexts. Mourtzis and Doukas for example create a knowledge base using a Case Base Reasoning (CBR) approach to capture and reuse knowledge when solving design and planning problems (e.g. scheduling) in a manufacturing plant [27]. The approach is demonstrated on a moulding-making process, where the knowledge base is queried for finding cases of orders of similar product variants and understanding from past experience what worked well in terms of scheduling of the operations and machines involved in the production of such product variant and for estimating a delivery time to the customer.

There are a number of ontological approaches for knowledge capture and management that have been proposed in the literature. A strong focus has been put into modelling products, processes and capabilities. Jarvenpää et al. developed an ontology for capturing manufacturing capabilities to support the automation of matching assets capabilities to processes needed to manufacture a product [28, 29]. Mo et al. expanded on Järvepää's ontologies to include semantic models that capture system re-configuration with the aim to match process requirements to changes in the physical system and so enable faster reconfiguration in robotic cells [30].

With regard to capturing machine learning knowledge, Braga et al. propose an ontology, MLOnto, to describe the domain knowledge that encompasses the domain of machine learning [31]. The ontology was created based on a literature review focused at incorporating concepts of the Autonomous Systems sub-domain. This is one of the few efforts so far attempting to formalise the machine learning knowledge domain. How to apply these techniques and build machine learning pipelines is a domain knowledge that still needs to be formalised in order to start automating ML model development, and particularly for solving manufacturing problems. A few works start to emerge in this direction [26, 32] but more general frameworks need to be developed to link manufacturing to ML knowledge.

2.3 Industrial ML model operationalisation

From the model management perspective, there have been several technologies that have been proposed to support the operationalisation of Industrial AI solutions. With the success of continuous software engineering practices (DevOps), there has been an increase interest in the rapid deployment of machine learning models, referred as MLOps [33]. It mimics DevOps practices with additional actions that are specific to machine learning such as model monitoring and model retraining. In the literature, general architectures based on MLOps have been proposed for industrial contexts. Zhao for example, proposes an MLOps architecture for industrial settings integrating technologies such as Data Version Control (DVC) for data and model versioning, Amazon Web Services (AWS) for data storage, MLFlow for pipeline implementation [34]. Raffin et al propose a cloud-edge architecture for the operationalisation of machine learning models in manufacturing shop floors [35]. The architecture is based on five components: data storage and management, edge management, cloud, model development and versioning and model monitoring. The authors stress the importance of the correct management of complex data as this enables successful improvement of machine learning models in the long term. Although the main architectural elements are proposed, it is not clear how data, models and processes can all be linked to ensure an effective reusability or further training of ML models. Elements such as meta data, code repository and feature store can really help to shorten the transition of models from prototype to production [36]. Applying MLOps approaches in an Industry 4.0 context is mainly challenging due to the heterogeneity of the environment, having many individual AI solutions involved. A full MLOps architecture must cover the whole environment of the cyber-physical system and its context [37].

The Asset Administration Shell (AAS) has been recently proposed to life cycle manage AI solutions. AASs can be digital twins not only of physical objects but also of software programs. As proposed by Rauh et al., AI solutions can be described as an AI artifact or instance with three life cycle stages: before training to a data set, after training, and during runtime [38]. After the training, the model instance is frozen. As a result, a simple hierarchical structure is used to capture all the relevant changes of the AI model. The AAS is an obvious way to be able to link ML model to the physical asset it is supporting; however, it is not clear how this can enable future model reuse across other assets. In addition, the paper does not consider the monitoring of the model's performance after deployment, which is key for the continuous update and re-deployment of models.

2.4 Model reusability and continual learning

Achieving generalisation of machine learning models in manufacturing contexts is very challenging due to the changing environment. Data sets are typically not representative of all possible changes that may happen or they are highly imbalanced (e.g. the case of anomaly detection). There are several ways in which this challenge can be addressed. One way is to use data augmentation techniques. Ruediger-Flore et al., for example, create synthetic image data from CAD models in order to develop inspection systems based on Deep Learning when dealing with a shopfloor that is constantly handling new "unseen" products [39]. Lyu et al. use a technique based on Gaussian Noise and Signal Stretching techniques to augment vibration sensor data for motor fault diagnosis, improving accuracy when data sets are highly imbalanced [40]. Some work using Generative AI for augmenting sensor signals has been proposed as well [41, 42].

Another way of addressing this is looking at existing models developed in similar conditions and perform transfer learning to distill some of the learnt features and further train them, obtaining a model that is now good for the new conditions. This is particularly useful when data available of the new conditions is limited. Wang and Gao explore the use of foundation models used in image and object recognition for transfer learning in machine condition monitoring and the authors stress the need to further understand what the boundaries for effective model transferability are, particularly when using pre-trained models with non-manufacturing-specific data [43]. Giannetti and Essien demonstrate the effectiveness of transfer learning for predicting operational parameters of can body maker machines [44]. The authors propose the use of the F-score measure to select the model to use for transfer learning from a set of multiple models that were developed from similar machines. Their main finding was that the training strategy that is most effective depends on the data, and so contextual information of the process to understand the differences in the data is useful to ensure the effectiveness of transfer learning.

Another way to address this challenge is developing continual learning frameworks [45]. Continual learning is different from transfer learning as it focuses on sustaining good performance on previous knowledge, which can be forgotten as the model is trained only with new incoming data. In the continual learning context, new data might mean a drift on the data whilst the task and classes within that task remain the same — *domain incremental learning* , new data as a new task that needs to be learnt — *task incremental learning* — or new data as a new class that needs to be learnt within an existing task — *class incremental*. Both class and task incremental learning require architecture changes, as new output nodes are added to the last layer. The main advantage of using continual learning strategies is to avoid re-training a model from scratch every time there is a change on the data or the task changes. This is very particularly beneficial because it reduces time and computing power as training sets become larger and larger. It also takes advantage of the knowledge already acquired (like in transfer learning) but with the benefit that it also retains the previous knowledge, being able to incrementally deal with old and new data contexts.

Trinh et al. introduce a continual learning framework for a ML model that is supporting a robot on carrying out different milling processes. The model is used to predict the robot joint torque and it is demonstrated how it can incrementally learn to predict this torque whilst the robot is performing new tasks. Applying two different techniques of continual learning, elastic weight consolidation (EWC) and synaptic intelligence, the authors show an improvement on the retention of knowledge when the prediction model is trained in a task incremental way. The authors explore a short-term (2–3 months) retraining approach, highlighting the need to understand how these techniques could perform when the drift is larger between training iterations to account for the natural wear of the system.

Despite the current efforts in different directions, there is not a clear approach for how the industry will operationalise AI if the current ML approaches proposed are focusing on large complex and data-intensive models that are not robust to change and that are so complex to develop. It is also clear from the literature that semantic models offer a good opportunity for better decision support for system reconfiguration. These could be leveraged to not only capture the physical world but the ML approaches that are linked to them to support their operationalisation. There are still "missing links" in these models: (1) from how the manufacturing system and process links to the data characteristics that can support a better model selection for transfer or continual learning, (2) from the operator that understands the process and that can play an essential role in the design of a successful solution,

(3) from the choices made during the ML model development process (i.e. ML pipeline), which is typically in hands of the data science expert, and finally (4) from the model itself, capturing development metrics to support ML model redeployment and reusability. All of these pieces of information play an essential role on the success of the scalability and robustness of ML solutions.

To tackle this major challenge, this paper proposes a framework to support the development and scaling up of robust ML solutions. By linking manufacturing system and process to data and ML solution, it is possible to streamline future developments of a similar solution on a different asset as well as to manage the life cycle of such solution. With an ontology model at its core, that leverages some of the existing semantic manufacturing models and expanding these further, intrinsic knowledge that is involved in the development of ML solutions can be defined, standardised and captured; from Asset, Process, and Capability to capture *manufacturing knowledge*, to Sensor, Data, Algorithm, Model and its metrics, to capture *data science knowledge*. By using the semantic links between manufacturing processes and AI solutions, which currently do not exist in practice nor in the literature, inference and similarity measures can be used to automate the identification of existing ML models or elements of the model development to be reused in a new context, either through transfer learning, continual learning and Automated Machine Learning (AutoML) tools. Having a way to identify existing models or data/pipelines from similar industrial AI solutions can speed up development, avoiding to create solutions from scratch. It is also an effective way to deal with limited data, as models deployed across other machines can be used for machines with low data availability and then further improved as data becomes available from the new machine. Additionally, the approach can enable the non-experts in the factory floor to not only create these solutions but also understand how they work and why they work; understand what approaches work best for which type of problems and when does it make sense to reuse models or create them from scratch.

In the following sections, an overview of the proposed framework will be presented with a particular focus on the semantic models and how they can be used to support effectively continual learning. Particularly, the experimental section will demonstrate how semantic similarity is an important factor for the identification of similar models and data, which has an impact on the level of success of the training approaches.

3 Framework for robust Frugal Industrial AI

To address the aforementioned challenges, a framework based on semantic models and advanced ML learning techniques together with the latest approaches in AutoML, AAS and a MLOps architecture is proposed (Fig. 1). The underpinning philosophy of the framework is based on the assumption that not all data is available at one time for robust models to be created, but rather sequentially available in time, enabling this way Frugal AI. This is particularly important as the real world, in particular, the manufacturing shop floor, deals with

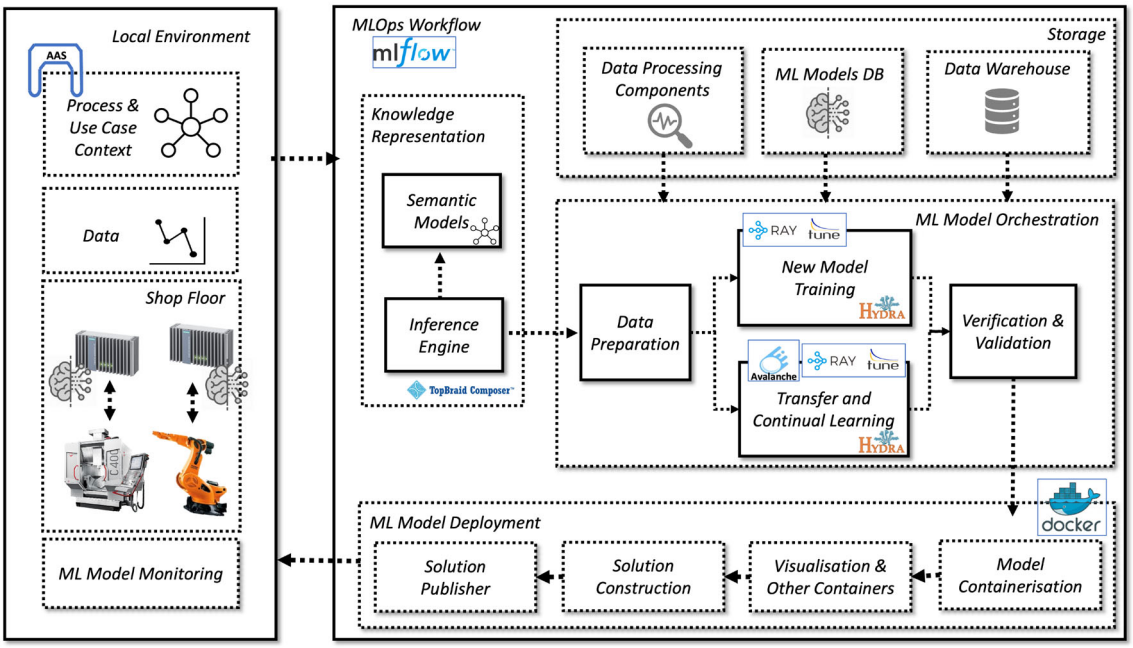

Fig. 1 Framework for the development and management of Industrial AI solutions

this challenge [46, 47]. It is then important that the framework is also underpinned by stable algorithms that can learn from such real environment. Continual Learning is a learning strategy that is necessary when a model needs to learn from a continuous stream of data that may change through time but it is not know how or when. In addition to the computational efficiency that a continual learning approach provides, the framework makes use of data augmentation techniques to handle both reduced data sets was well unbalanced data sets. The complete framework is comprised by the following components:

- MLOps workflow — This component encapsulates four main elements. The first one is the *Knowledge Representation*. Based on semantic models, this element underpins the framework providing a way to capture knowledge with regard to ML development as well as manufacturing process knowledge, and linking the two. As it will be presented in detail in Sect. 3.1, this allows to identify relevant models, data and existing pipelines that can be used for different industrial AI solutions. Second is the *Storage*, which includes models and data to support model training (transfer or continual learning according to the application). Third is the *ML Model Orchestration*, which contains the functionalities to execute AutoML pipelines through MLFlow and Ray Tune based on the knowledge extracted from the semantic model. Finally, the ML Model Deployment component containerises a trained model and any other elements of the Industrial AI Solution such as visualisation, Key Performance Indicator (KPI) monitoring and data storage containers. Using the Docker environment, it integrates these and prepares the necessary configuration for the solution to be published. The solution can then be published by the publisher into an edge device through the device's AAS.
- Local environment — the local environment comprehends the set of hardware and software that are at the factory floor or "at the edge" of the process. This will include the asset itself (e.g. robot, computer numerical control (CNC) machine), the edge device on which the ML solution is deployed into and an instance of an asset administration shell that serves as the two-way access point of data from and into the rest of the framework. From this local environment data from the sensors as well as ML solution outputs and ML monitoring KPIs are gathered through the AAS. The asset administration shell has references to the instance of the model deployed locally through semantic identification (ID), being able that way to infer all the information related the creation and deployment of such model. In the same way, the AAS has semantic IDs for the physical asset and sensors that are generating the data locally. This can be used to ensure data can be semantically linked when it is stored in the data warehouse.

3.1 Semantic models

At the core of the framework are the semantic models. For automating or supporting the automation of the development

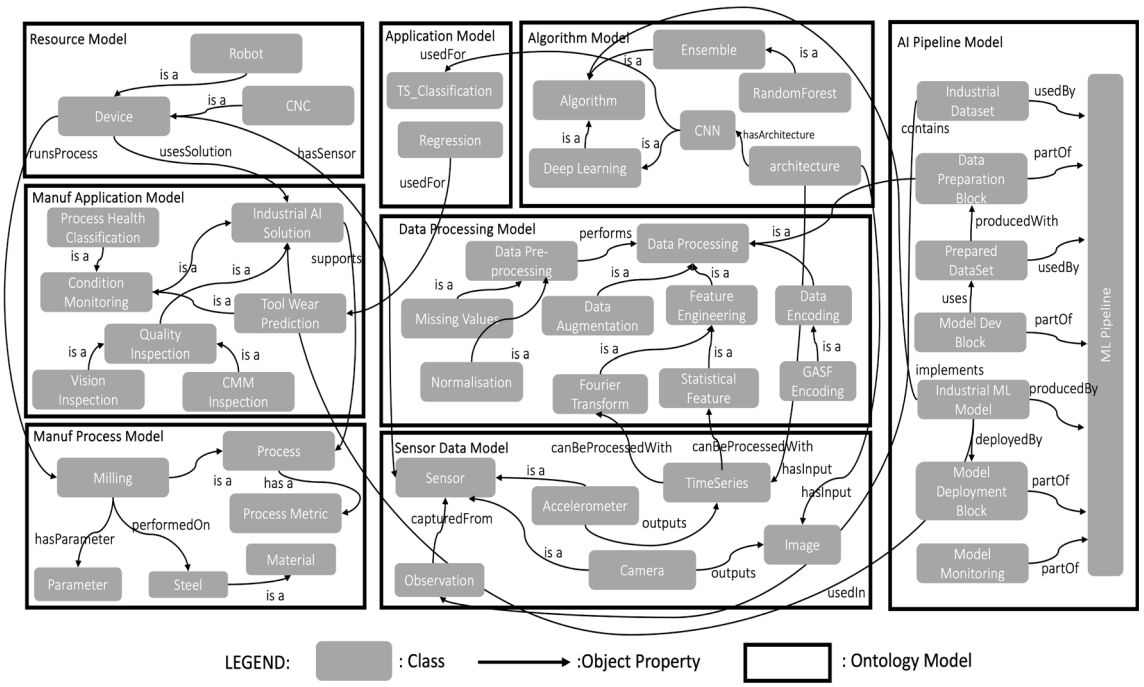

Fig. 2 Semantic models for the developing and managing AI solutions in manufacturing environments

of Industrial AI solutions, it is essential to formalise the different elements of knowledge that need to be captured and for these to be machine-interpreted. Building from the work from Järvenpää et al. [29] on the Process and Resource Models, the work from Janowicz et al. [48] on Sensors and Samples, and the Algorithms and Application models from Braga et al. [31], new models are introduced which aim to link AI methods to Industrial AI solutions that are then semantically linked to one or multiple manufacturing problems. These additional models are Manufacturing Application, AI Pipeline and Data processing. A general view and interconnections of the eight models that comprise this framework are presented in Fig. 2. Each of these semantic models support in different ways the development, identification and reuse of ML models:

1. Resource model — this semantic model captures all data related to an asset (e.g. robot, CNC machine, operator). The asset is linked to a process that is or has been carried out and to the sensors installed on it. From this, the data that is produced from that asset for a particular process can be inferred. This helps identify the best ML models to be used for transfer learning or for its data to train other models. This is particularly useful when data is limited.
2. Manufacturing application model — this captures the specific industrial ML solutions and how they are linked to the process. For example, a vision inspection solution may support/control the quality of a 3D printing process, or a condition monitoring solution may be related to a machining process. Capturing the industrial solution is essential for querying the ontology for similar ML models.
3. Manufacturing process model — defines common processes that are needed to deliver a product, and which are linked to the capabilities of assets. A process is characterised by its parameters as well as by metrics that are used to monitor such process, which provide context when selecting models to be reused.
4. Application model — defines a solution or industrial problem from an ML perspective as either a classification or prediction problem. These classes help to link ML applications to industrial solutions as well as to the ML techniques that can be used to solve such application, supporting the design of ML pipelines.
5. Algorithm model — captures AI/ML approaches and their parameters. The same approach might be linked to multiple Manufacturing Applications the same way that a Manufacturing Application may be solved by different algorithms.
6. Data processing model — defines all the different processing steps that are involved in a machine learning pipeline. The model links certain data types to techniques, supporting the automation of the pipeline.
7. Sensor data model — captures characteristics of measuring devices on assets as well as enables semantic annotation of the samples gathered from the device. It is then possible to infer which process is connected to the data through the links between sensors, assets and process.
8. AI pipeline model — brings together all the elements that are used for the development of an industrial solution and to track its performance. This supports future model development by using complete or partial pipelines of existing models from similar process/machines as well as model monitoring for continuous adaptation.

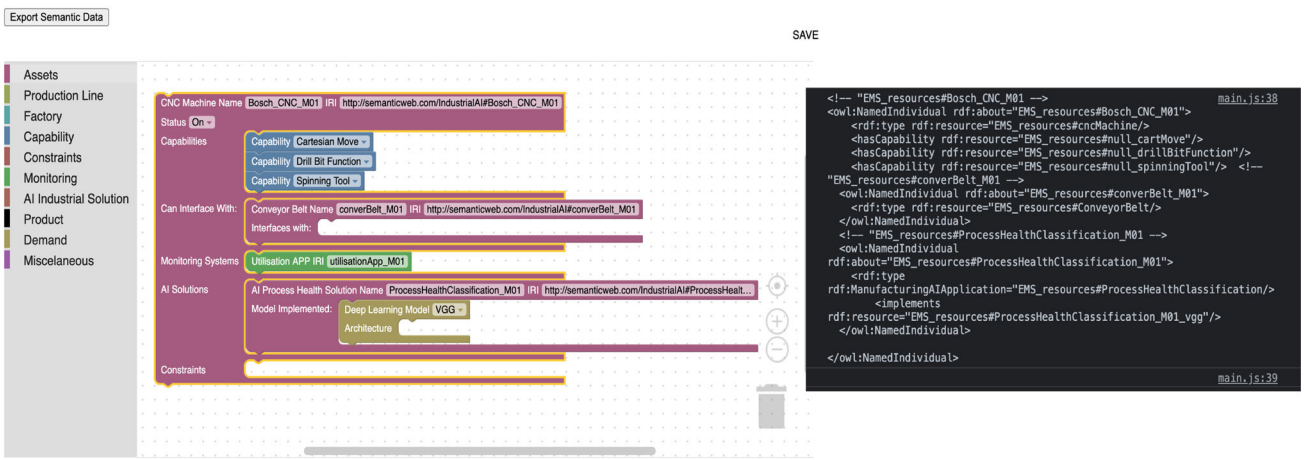

Fig. 3 (Left) Drag and Drop environment developed in Blockly. The environment uses the concepts defined on the semantic models and offer the user the possibility to use these as blocks to define instances related to the manufacturing environment and models. (Right) Owl-formatted output generated by the blockly tool

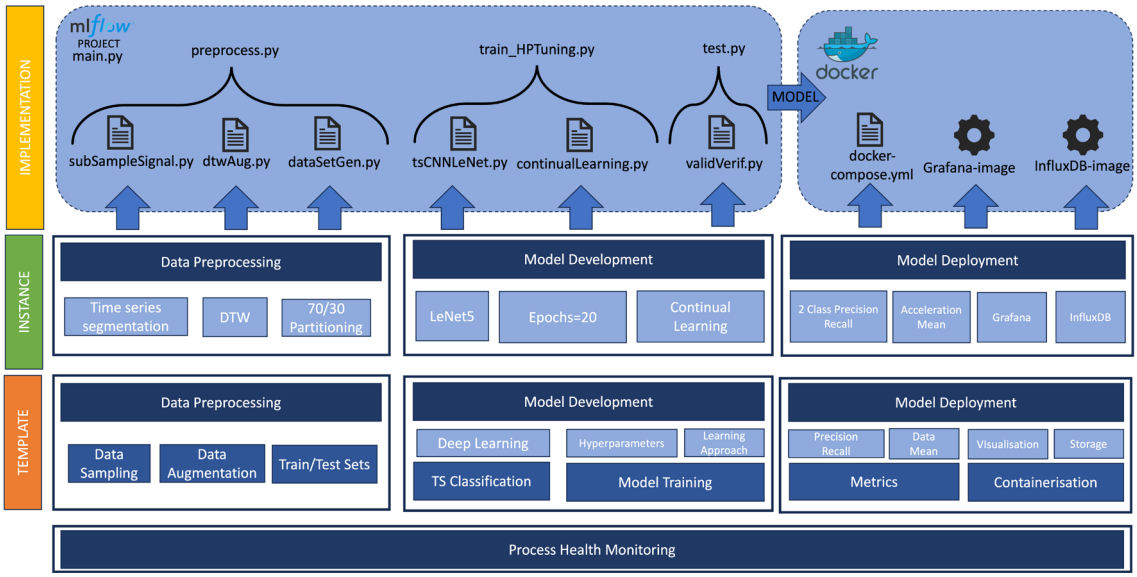

Fig. 4 Machine learning template pipeline, instance and linked implementation for performing process health monitoring through time series classification. Code modules are independent, allowing for other instance pipelines to use these

3.1.1 Instance creation

From the defined semantic models, instances for existing assets on the shop floor can be created. Although these are typically created manually using tools like Protégé, a drag and drop approach using Blockly has been created to facilitate any user to create these instances. Figure 3 presents a snapshot of the tool developed to facilitate this task.

In regard to ML pipelines, a starting point is to build *template* pipelines. Template pipelines provide a blue print for creating instances of ML pipelines for solving specific problems and link those instance pipelines to actual snippets of reusable code. Rather than "reinventing the wheel", the current state-of-the-art research literature on the most common successful applications of AI in manufacturing is used. For this paper, particularly process health monitoring pipelines are defined. Munaro et al. provide an exhaustive classification of AI techniques used for tool wear monitoring with details on the data used, such as sensor signals, material type, cutting parameters, type of machining operation (e.g. drilling, milling and turning) and the most common features in the time and frequency domains used [49]. These can then be used with AutoML tools to fine tune them for a particular data set. As it can be seen at the bottom of Fig. 4, a template pipeline defines that the problem of process health monitoring in this case can be solved by doing time series classification, using time series data augmentation as an strategy to deal with conditions where the asset that will be deployed to has limited data. Other template pipelines may contain class balancing through removal of the majority class as a different approach to the same problem. The template pipeline in the figure indicates that for time series classification a metric of precision and recall (as they will be defined later on in Sect. 4) can be used to monitor the accuracy of the model. In the middle section of Fig. 4, an instance of such template pipeline is depicted, that corresponds to an actual existing implementation of that template pipeline. The instance pipeline provides a link to the actual snippets of code that are produced for each of the pipeline steps, which can then be executed and reused across multiple MLFLow projects. MLFlow allows to write general main code structures that calls different snippets of code for the different steps of the pipeline.

Figure 5 presents an extract of semantic instances of a pipeline created based on the template pipeline of Fig. 4. The concepts presented in the figure are linked to the case study that will be further introduced in Sect. 4. To describe Fig. 5, concepts linked to the semantic model are presented starting with capital letter whilst instances are in `typewriter` font. The instance `processMonitoring01` is of type ProcessHealthClassification which is a type of IndustrialAISolution. This solution is *usedBy* Resource `M01`. It can be solved with a time series classification (TS_Classification) approach with a DeepLearning Algorithm such as a Convolutional Neural Networks (CNN) [11, 50–52] and can also be solved with a Regression approach which according to the literature can be addressed with a DeepLearning Algorithm such as Long Short Term Memory (LSTM) approach [53–55]. The model `CNNM01Model` developed for Resource `M01` is linked to `ProcessMonitoring01` which *supports* Process *OP01* which is related to capability StepDrilling. The model has been produced by the `M01Pipeline` instance of type MLPipeline.

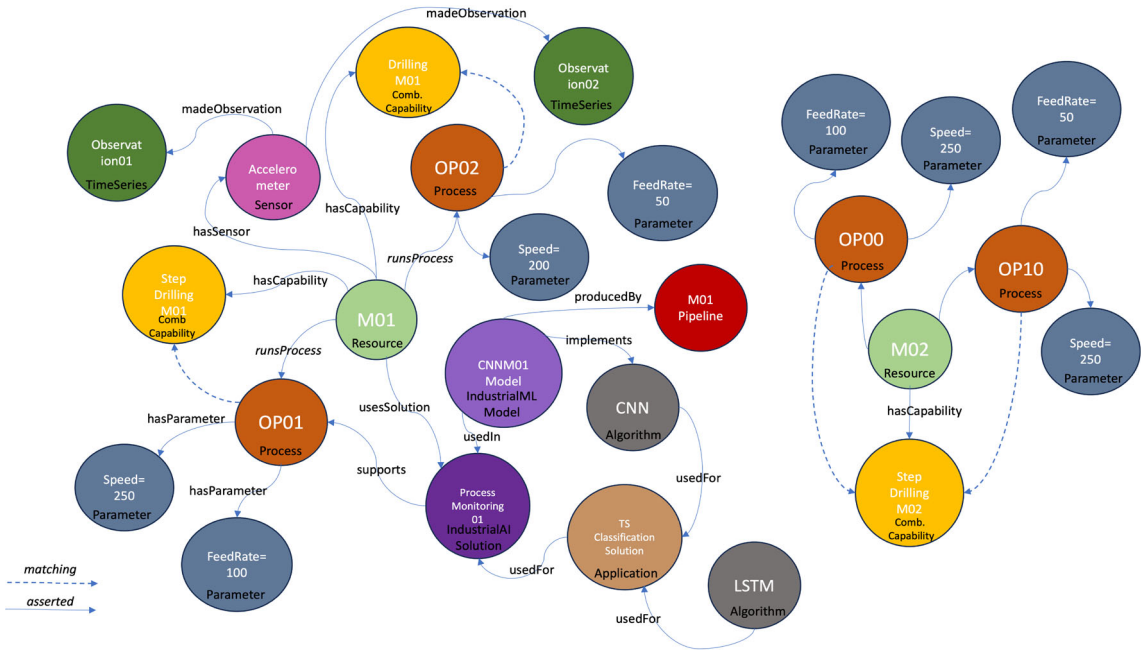

Fig. 5 Extract of instances from the semantic model. Colours differentiate the class to which the instances belong to

3.1.2 Similarity measure and inference

When a new model needs to be developed either for a new asset or when an existing model needs to be updated due to changes in the process running in the asset (e.g. change in process parameters), it is useful to identify models from a similar context scenario to use transfer learning as a way to speed up development, or identifying existing models from the same context scenario where applying continual learning strategies makes sense to build model robustness. Two models may be similar, either because they are used for the same Manufacturing Application, but there are other factors that might contribute to similarity such as algorithm type, architecture, input data and more importantly, process parameters. Semantic similarity is a challenging problem for which multiple metrics have been proposed in the literature and which typically aimed at word sense disambiguation, machine translation, and information retrieval [56]. According to [57], semantic similarity metrics are typically either path-based or information-content related and are usually aimed at word similarity. Edge or Path-based metrics such as the *Shortest Path* [58], the *Weighted Link* [59] and the *Hirst and St-Onge Measure* [60] algorithms calculate the similarity of two instances based on how closely connected they are or the topology of the graph to which the instances belong to. Information content may use hierarchy on the model structure to determine "ambiguity" or look at the attributes and neighbours. The *Resnik* metric [61], for example, defines that two concepts are more similar if they present a more shared information, and the information shared by two concepts C1 and C2 is indicated by the information content of the concepts that subsume them in the taxonomy (concepts they have a hierarchical link with). In the context of ML model or IndustrialAISolution similarity, there are several factors that need to be taken into account. High similarity between two concepts would mean that they share not only the same values in their attributes, but that they contain the higher number of similar neighbours through their object properties. Figure 6 shows a very simplified example of what this similarity means. From the three instances of Asset shown, two of them not only show to have a higher (3) number of similar object properties, but such properties define the asset as a machine tool or an asset with machining capabilities, making them semantically closer for the purpose of IndustrialAISolution search. This is particularly important, as although assets might be performing the "same process" (e.g. drilling) the type of drill bit would have an influence on the vibration patterns, and so on the potential similarity of the data which is important for transfer and continual learning. Formally, based on [62, 63] similarity between two concepts is then defined as:

$$sim(C_1, C_2) = f(C_1 \cap C_2, C_1 - C_2, C_2 - C_1) \quad (1)$$

where the similarity increases as the intersection of the features increases and the distinctive features (features that belong to one concept and not the other) decreases. To move into the range of 0 to 1, the ratio is used:

In the context of asset, process, AI model and data, similarity take into account the characteristics of the asset, of

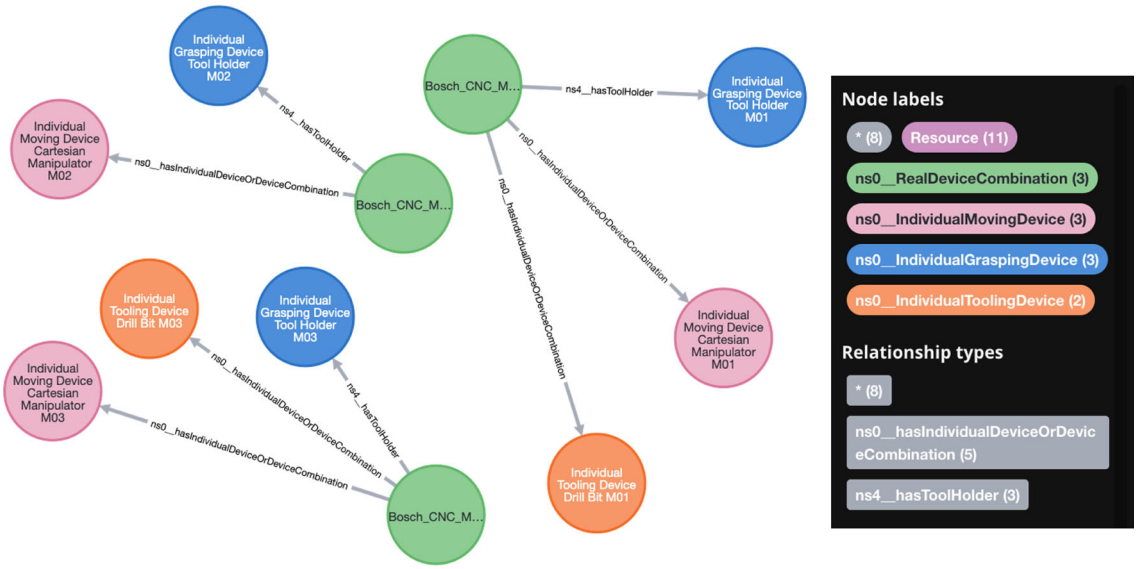

Fig. 6 Assets and related object properties that determine the assets' capabilities as well as the similarity of concepts

the process running in the asset, the type of sensor data collected from that asset, its feature distributions and the type of ML model/IndustrialAISolution that is implemented in such context.

3.2 Model retrieval and ML pipeline instantiation

The semantic similarity as defined in the previous subsection is then used when performing the search for existing models or pipelines. Using Protégé and TopBraid, a series of SPARQL and SPIN rules were implemented to define a search logic. This search logic is based in the following general steps:

1. In scenarios where an IndustrialAISolution exists for the desired asset but the process has changed (e.g. machine carrying out a new operation which leads to concept drift), an existing model can be directly retrieved from the semantic model.
   (a) Continual learning can be used with the existing model when it is important for the model to be accurate both for the old process conditions as well as the new ones. However, continual learning only makes more sense when knowledge from the previous learnt task is useful for the new task [64]. This is currently a challenge as there is no effective way to anticipate if the new task will be closely related in terms of data patterns; however, here is where the process contextual information and similarity is useful to support the decision of using the same model or create a new one for the new process conditions.
   (b) If data from the already existing model exists and there is no computational constraint, then instead of applying continual learning, a new model can be developed from scratch using previous and new data using the same pipeline from the existing model to speed up model development.
   (c) In the case of a completely new task and not having the possibility of training from scratch, a new model can be created using the previous model as a starting point for transfer learning. These can be useful to reduce training time and potentially not having to rely on large amounts of data being available for the new process conditions.
2. In scenarios where no model exists for the given asset, then it is useful to identify existing solutions in other assets. A good candidate asset would be such that can provide the same capabilities, particularly if they share the same physical features and if the sensor/data that is collected from it is *similar*.
   (a) Given the identified assets (and potential data found), a search on models related to the IndustrialAISolution of interest is performed on the ontology.
   (b) Models identified would be trained under specific process conditions. And so, *similar* process type and parameters would be preferred.
   (c) If process parameters are not the same but the type of data available has the same characteristics (sensor type, frequency), then transfer learning is a good option for speeding up development.
3. In scenarios where no model exists for similar assets but a different asset has a solution to the IndustrialAISolu-

tion of interest, the pipeline for such model can be used and AutoML architecture and hyperparameter optimisation can be used to automate the search for the optimal model. The pipeline concepts in the semantic model provide a link to the snippets of reusable code, which can be executed using MLFlow Projects. MLFLow Projects provides a way to execute in python different pipeline snippets of code once a main file is defined which defines different *entry points*, each of these entry points being the steps in the pipeline that need to be executed. An example of a MLFlow structure is presented in Listing 1.

```
name: mlflow_pipeline

conda_env: conda.yaml

entry_points:

  load_raw_data:
    command: ''python preprocess.py''

  train:
    command: ''python trainHPTuning.py''

  validate:
     command: ''python test.py''

  main:
    command: ''python main.py''
```

Listing 1 Example of MLFlow Project file

4. If none of the previous steps could be followed, then a *template* pipeline can be used, provided the data available meet the characteristics needed for the data preprocessing steps of the pipeline.

3.3 Model deployment

Once a new model is created or re-trained, semantic information needs to be updated before preparing it for deployment. The steps for model deployment include the following:

1. A new model requires the creation of a new instance in the ontology, and add all the necessary semantic links to it (e.g. process, asset, data).
2. Based on the process metric captured in the semantic model, as well as statistical features of the data used to trained the model, a ML model metric can be defined within the IndustrialAISolution to be used in the user interface. The user then would have a way to feedback into the solution the accuracy of the model as well as the data shift, and this way perform continuous monitoring once deployed.
3. An AAS file (if it does not exist yet for the intended asset) for the IndustrialAISolution is created with the corresponding semanticID of the solution.

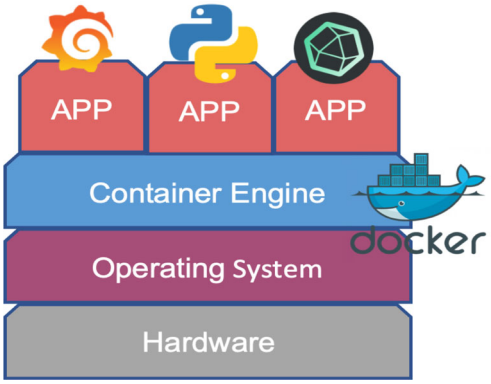

Fig. 7 Industrial AI Solution Deployment Architecture

4. Although out of the scope of this work, the ontology model can also contain concepts regarding the visualisation preferences for each IndustrialAISolution and for each user. The trained model can the be containerised and coupled with a containerised visualisation solution and a data storage container (see Fig. 7).
5. Once the solution is packaged, it can be published to a device running a container platform like Docker.

A summary of the methodology of how the proposed framework is implemented in this work is shown in Fig. 8.

4 Industrial case study

In this section, a proof of concept of the framework is implemented and presented. Particular focus is given to the ontology models and how they enable robust model development using continual learning in a limited data scenario. In this case study, new data becomes available in an online manner and learning happens incrementally. Only new incoming data is used for training, delivering a more computationally efficient approach as well as taking advantage of existing knowledge to overcome the need for large amounts of data being available at a single time. This is particularly important, as real flexible and reconfigurable manufacturing environments will be dealing with unexpected changes in process type, process parameters as well as natural wear, all which cause concept drift. Using a continual learning approach it is possible to assure the delivery of more robust models in the longer term, which is key for scalability of the technology. The experimental work is divided into two parts. First, given an industrial scenario, the ability to query and identify solutions in the form of trained models or existing pipelines based on process context is presented. Then it is demonstrated how the contextual information is key to develop effective continual learning solutions for model robustness.

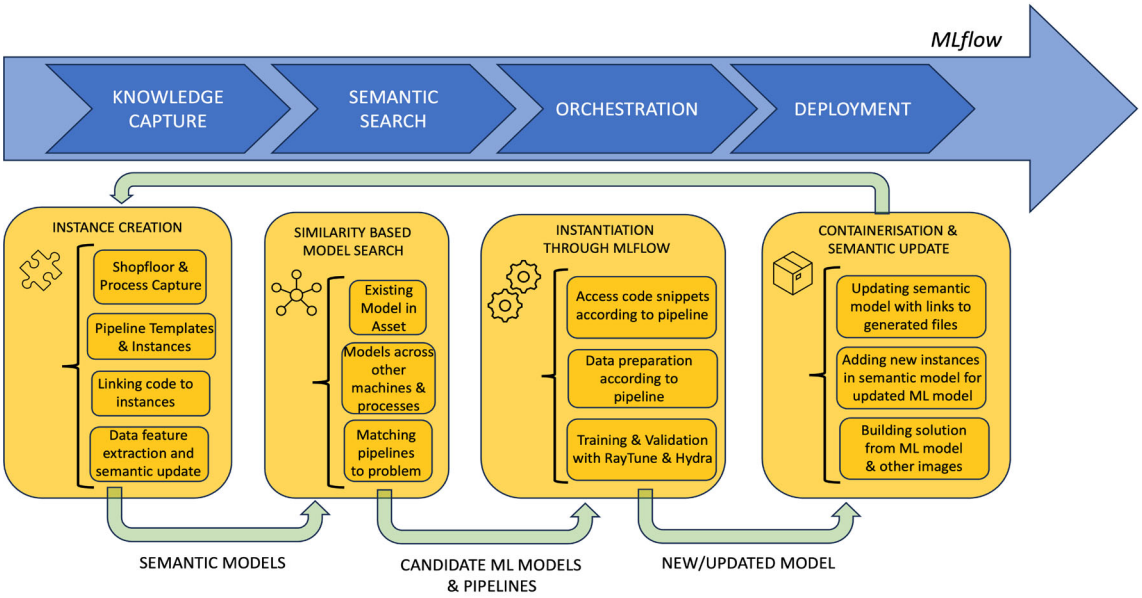

Fig. 8 Summary of the methodology that is implemented based on the proposed framework

To achieve this, the Bosch process monitoring through spindle vibrations case study presented in [8] is used. This data set is the first of its kind which has been gathered from machining centres in a real manufacturing environment over a long period of time (3 years). Due to its nature, it presents the following complexities: there is high process variation in terms of types of operations, process parameters and types of tools, as well as part type variation in terms of geometries. This level of change introduces multiple environmental challenges as frequent mounting and unmounting of tools may lead to different process failures such as tool misalignment, chip clamping, chip in chuck and tool breakage in addition to natural ageing of components. This last factor is particularly challenging as slight deterioration would not affect the process quality initially, but would certainly affect the sensor data by causing additional noise. The Bosch data set consists of three brownfield CNC machining centres (M01, M02 and M03) which perform 15 different operations (i.e. OP00 - OP14) in different order according to the part, different duration and with different tools using different combinations of feeds and speeds. These details can be seen in Table 1. All machines have a low-cost tri-axial accelerometer mounted on the spindle that samples at 2KHz. The set up is shown in Fig. 9.

Table 1 Operations and cutting parameters performed by the three CNC machines in [8]

Tool Operation	Description	Speed $[Hz]$	Feed $[mms^{-}1]$	Duration $[s]$
OP00	Step drill	250	≈ 100	≈ 132
OP01	Step drill	250	≈ 100	≈ 29
OP02	Drill	200	≈ 50	≈ 42
OP03	Step drill	250	≈ 330	≈ 77
OP04	Step drill	250	≈ 100	≈ 64
OP05	Step drill	200	≈ 50	≈ 18
OP06	Step drill	250	≈ 50	≈ 91
OP07	Step drill	200	≈ 50	≈ 24
OP08	Step drill	250	≈ 50	≈ 37
OP09	Straight flute	250	≈ 50	≈ 102
OP10	Step drill	250	≈ 50	≈ 45
OP11	Step drill	250	≈ 50	≈ 59
OP12	Step drill	250	≈ 50	≈ 46
OP13	T-slot cutter	75	≈ 25	≈ 32
OP14	step drill	250	≈ 100	≈ 34

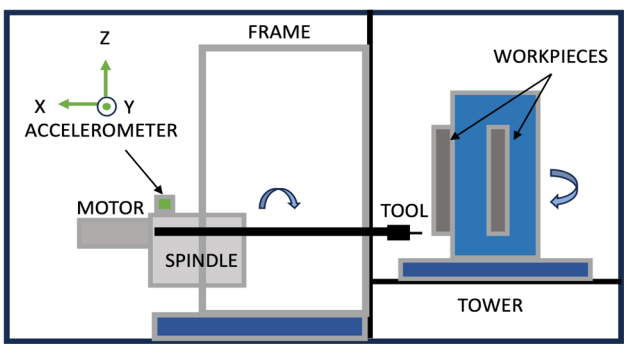

Fig. 9 Example of brownfield machining centre with tri-axial accelerometer mounted on the spindle to monitor the health of the processes running in such machining centre

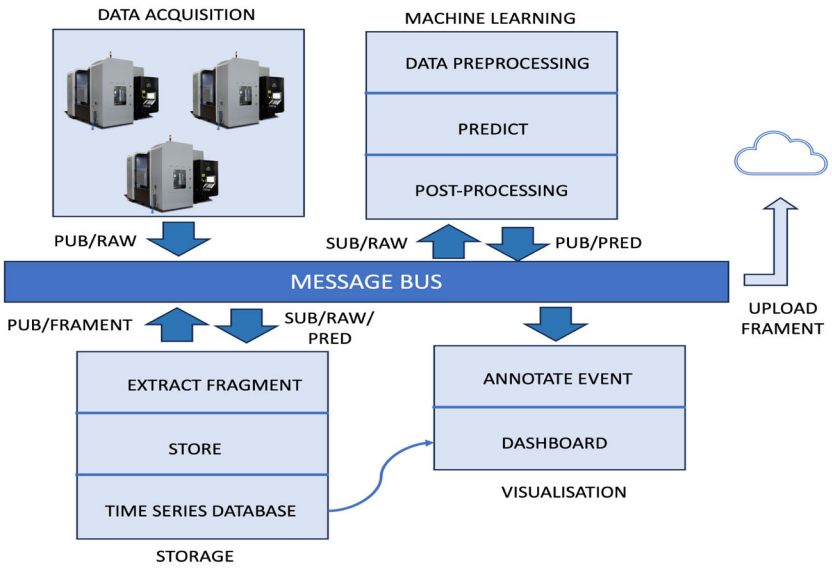

Fig. 10 Infrastructure at Bosch to collect data from the machining centres in a publish/subscribe approach

Data is collected from October 2018 to August 2021 in periods of 6 months. The data collection architecture implemented at Bosch is presented in Fig. 10. The data collected from the sensor is published to a message bus (enabled via MQTT) to which a machine learning and storage modules are subscribed to. The machine learning module provides a preliminary automated label (e.g."OK" and "NO OK" corresponding to the health of the process) to the raw signal which is later on checked manually after product quality inspection. Once it is checked by the expert, then the fragment is sent for final storage at the cloud.

This use case is particularly complex because of the difference in patterns (1) across operations in the same machine (see Fig. 11), (2) across two machines for the same operation (Fig. 12), as well (3) across the years for the same machine and operation (Fig. 13). In the case of (1), although a model could be built using all the operations in one time, in a real scenario it is not possible to ensure all possible operations and parameters are available. So ideally algorithms

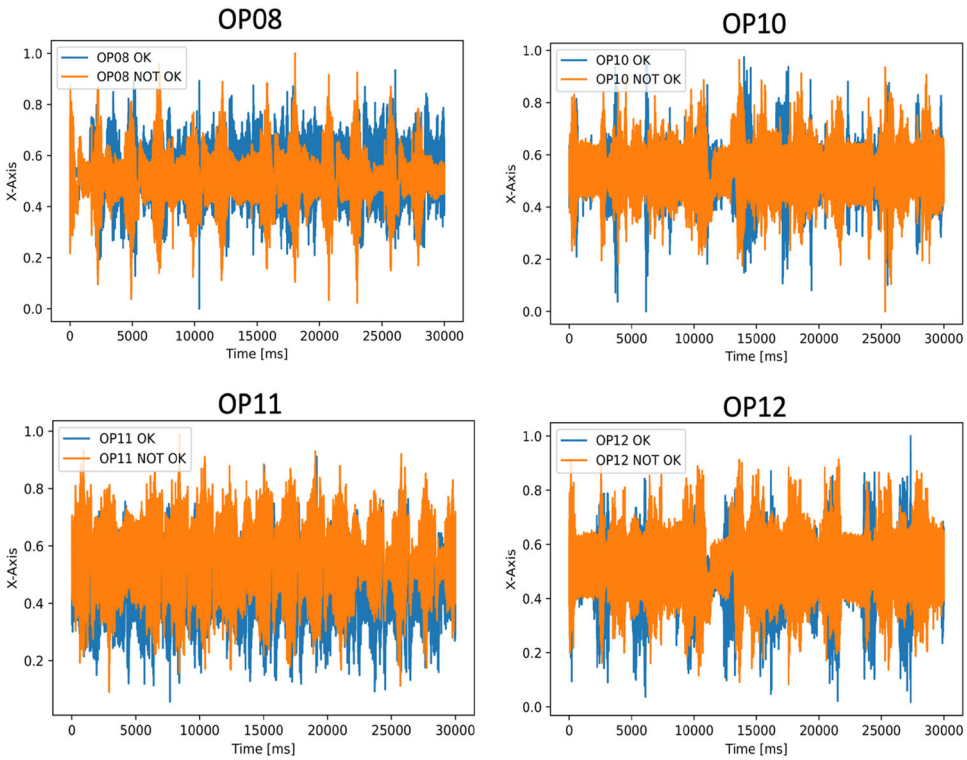

Fig. 11 Signal signatures of acceleration in x across four different operations 8,9,11 and 12 in machine M02. It is shown how different operations on the same time period have very different signatures

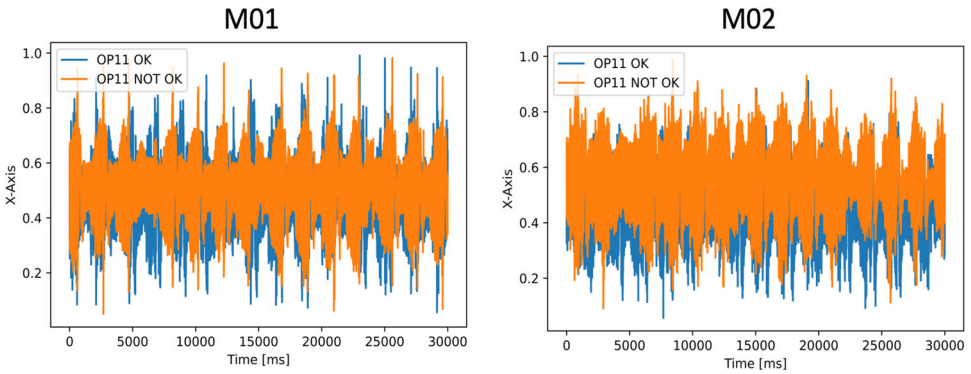

Fig. 12 Signal signatures of acceleration in x across machines M01 and M02 carrying out the same operation. The blue colour corresponds to the "OK" process and the orange colour corresponds to a "NOT OK"

need to be able to incrementally learn signal signatures from new processes. In the case of (2), as the same operation shows different signatures across machines, a model cannot be used straight away in the new machine as it will inherently decrease its accuracy. The model, though, could still be used for transfer learning, requiring less data to be used in the new machine. In the case of (3), there is an evident concept drift in both operations 10 and 11 when looking at a signal from February 2019 (blue) to a signal from August 2019 (orange). Although a model could be re-trained with regular backpropagation as the new data becomes available, this would decrease its accuracy for old data, for which signatures could still appear later in the future once changes such as maintenance are done to the machine (i.e. decreasing model robustness). In this case either the model is trained with all the data, old and new, or an strategy such as continual learning is implemented to retrain the model in the most effective way to ensure model robustness.

4.1 Initial semantic model and pipeline code snippets setup

Given the scenario described above, instances for assets, processes, sensors and data were created in the ontology (some of these shown in Fig. 14). Each of the 15 operations have different duration and are executed multiple times during the 3 year period. Due to this difference in duration, data needed several pre-processing steps before it could be used for model training. For this, a pre-processing pipeline code snippet was written in python which takes each run of each operation and segments it into equally sized samples (4096 data points per sample), which roughly corresponds to two seconds according to the sampling rate. This ensures a sample contains at least one complete cycle. As this information is known from the process, it is possible to write this pipeline code snippet that can be reused for different data sampling rates.

Other essential code building blocks for the pipeline are feature extraction methods. Statistical features as well as Fast Fourier Transform (FFT) blocks were created. Although the implemented classification algorithms in the following subsections work with the raw data directly, these features are used for semantic similarity, to help understand what model or data is potentially more useful in a given scenario. To show this, Fig. 15 presents FFT results of samples from four different operations OP08, OP10, OP11, OP12. Although they have the same process parameters (see Table 1), some additional frequencies appear in some operations. For example, Operations 10 and 12 show a particular peak before the 200

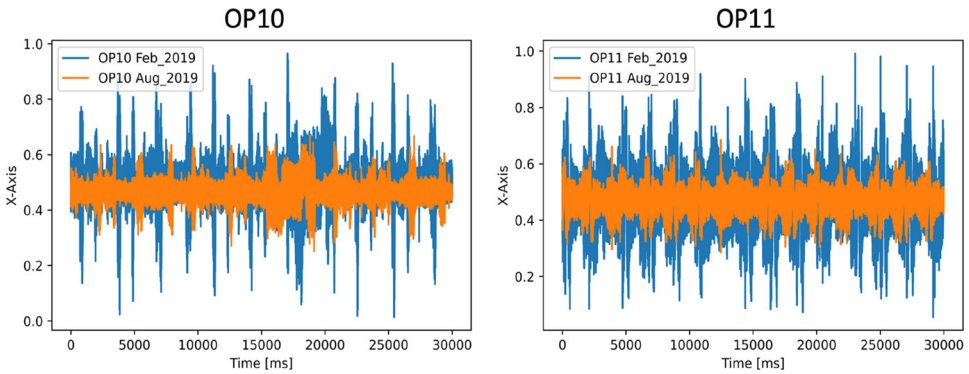

Fig. 13 Accelerometer signals acquired at different time periods for an "OK" process. There is a clear drift on both operations 10 and 11 despite the process still rendering a good quality part

Imports | Instances | Domain | Relevant Pro... | Change Hist... | Error Log | SPARQL

[Resource]	rdf:type
Bosch_CNC_M01	resourceModel:RealDeviceCombination
Bosch_CNC_M02	resourceModel:RealDeviceCombination
Bosch_CNC_M03	resourceModel:RealDeviceCombination
IndividualGraspingDevice_toolHolder_M01	resourceModel:IndividualGraspingDevice
IndividualGraspingDevice_toolHolder_M02	resourceModel:IndividualGraspingDevice
IndividualGraspingDevice_toolHolder_M03	resourceModel:IndividualGraspingDevice
IndividualMovingDevice_CartesianManipulator_M01	resourceModel:IndividualMovingDevice
IndividualMovingDevice_CartesianManipulator_M02	resourceModel:IndividualMovingDevice
IndividualMovingDevice_CartesianManipulator_M03	resourceModel:IndividualMovingDevice
IndividualToolingDevice_DrillBit_M01	resourceModel:IndividualToolingDevice
IndividualToolingDevice_DrillBit_M03	resourceModel:IndividualToolingDevice
MovingTool_CapabilityM01	capabilityModel:MovingTool
ProcessHealthClassification_M02	ProcessHealthClassification, owl:NamedI...
SpinningTool_Capability_M01	capabilityModel:SpinningTool
ToolHolding_Capability_M01	capabilityModel:ToolHolding
cnnModel_ProcessHealth_M02_Op08	OneClass_CNN, owl:NamedIndividual
cnn_Model_ProcessHelath_M02_op04	OneClass_CNN, owl:NamedIndividual

Fig. 14 Instances of three CNC machines with their corresponding low-level capabilities, which are used to infer if the machines have drilling capabilities

Hz which is not that obvious on OP08 and OP11. Although the nature of that frequency might be different in both operations, it can be helpful for knowledge transfer during training, speeding it up but also making the learning process more data efficient.

Another important building block of the pipeline is data augmentation. In the case of the Bosch data set, due to the imbalanced nature of the data, with a higher number of "OK" samples, which is common on the manufacturing shop floor, the Soft Dynamic Time Warping (DTW) method was implemented using the tslearn python package as another pre-processing pipeline module. This allows to produce an equivalent number of samples across classes for training. DTW is a method for measuring the similarity of two time series signals. This method is used for data augmentation in the following way [65]: starting with a random initial time series chosen from a data set, a weight equal to 0.5 is assigned to it. From the selected time series, its 5 nearest neighbors are identified using the DTW distance. From these neighbour signals 2 are randomly selected and assigned a weight of 0.15 each, making thus the total sum of assigned weights till now equal to $0.5 + 2 \times 0.15 = 0.8$. To have a normalized sum of weights (equal to 1), the rest of the time series of the neighbourhood will share the rest of the weight 0.2. This weighting allows to derive a new time series that follows a similar distribution. The soft DTW method has been a very successful time series augmentation method; however, other methods could be used and added to the semantic model as a different instance of the Data Augmentation concept.

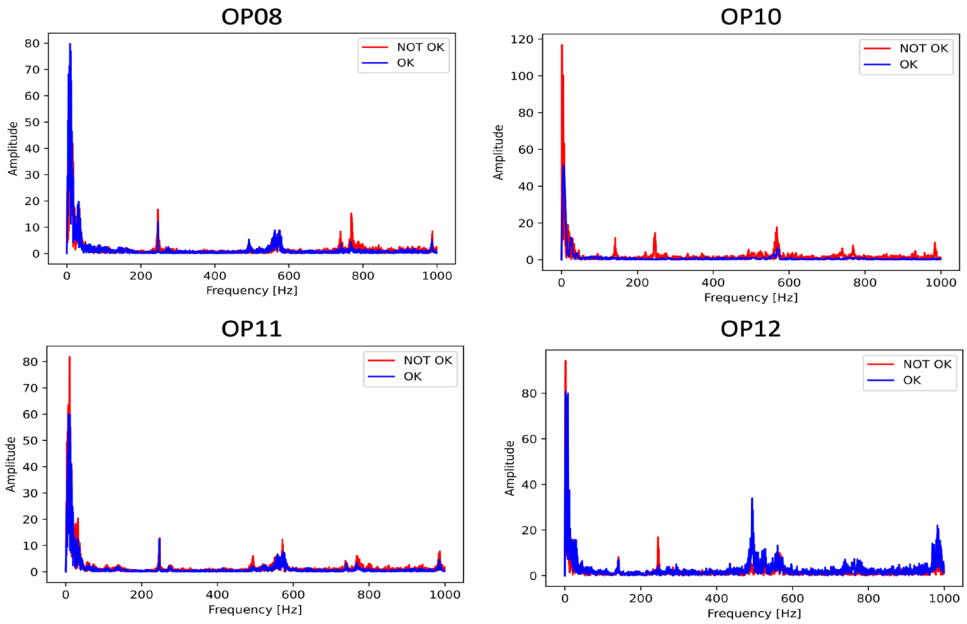

Fig. 15 Fast Fourier Transform on samples from four operations in machine M02

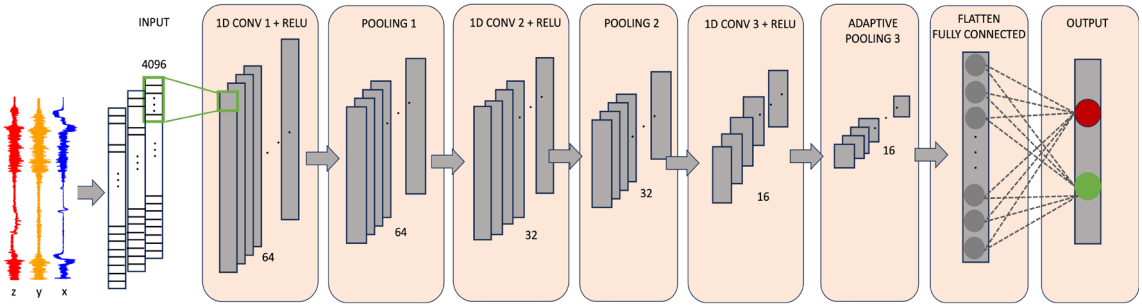

Fig. 16 CNN architecture used for Process Health Monitoring (Time Series Classification)

As an initial pipeline to be stored on the semantic model, a typical solution for process health classification using a CNN architecture is modeled using the well-known LeNet5, as used in [11]. This model architecture is shown in Fig. 16.

A CNN is a deep learning approach with two main building blocks: the *convolution layer* and the *pooling layer*. A convolution layer is formed by a series of neurons that are connected to neurons of a previous layer based on the their *receptive field*. For example, in the first convolution layer for time series classification, each neuron is not connected to each individual sample point of the input sensor signal, but to only those data points within a receptive field (highlighted in green in Fig. 16. The first convolution layer learns to detect the lower-level features of the signal, and further convolutions assemble these features into higher-level ones. The process of learning at each convolution layer allows the model to tune, or find the best weights (i.e. filter) of a neuron to be able to identify those features that are useful for classification. The *pooling layer* downscales the output of the convolution, thus reducing dimensionality, the local sensitivity of the network and computational complexity.

The architecture shown in Fig. 16 receives as input a 3-channel vector, where each channel is the vibration signal in each axis. The architecture then has stacks blocks of convolution + ReLU (layer used to speed up training) + Pooling to then flatten the output before doing the final classification. There are two output nodes for "OK" and "NOT OK". Once again code snippets for the architecture, defined as a python class tsCNN, as well as for the training process using Ray Tune for hyperparameter optimisation, Avalanche for continual learning algorithms and Hydra for training monitoring, logging and management of output files where implemented and semantic model concepts were updated with the generated snippets.

To create an starting point for the experiments presented in the next section, two process health classification models for Machine 02 following the aforementioned pipeline building blocks were created, one for OP08 (M02_OP08) and one for OP10 (M02_OP10). In both cases, all the data available for that operation was splitted in 70% for training and 30% for testing. For model M02_OP08 that meant using a total of 822 sample signals, from which 761 correspond to "OK" process and 61 to "NOT OK". As it is a very imbalanced data set, once it was split on train and test sets, the training set was augmented to achieve same number of samples for each class. Hyper parameters used for these models are shown in Table 2.

As this is an imbalanced classification problem, several metrics are used to measure the accuracy of the models. First, Precision, Recall and Accuracy which are three common metrics of classification problems were implemented. These are calculated based on the number of True Positive (correct classification of the Positive class), True Negatives (correct classification of the Negative class), False Positive (number of incorrect classification to the Positive class) and False Negatives (number of incorrect classifications of belonging to the Negative class). Precision quantifies the number of positive class predictions that actually belong to the positive class. Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made. Accuracy is the proportion of correct classifications out of the complete number of predictions made. Finally, the F-Measure is also used as this is a better reflection of accuracy when having the Positive class as a minority class. It provides a way to combine both precision

Table 2 Hyperparameters used for developing models M02_OP08 and M02_OP10

Hyperparameter	Value
Optimiser	Stochastic Gradient Descent (SGD)
Learning Rate	0.0001
Momentum	0.9
Batch Size	10
Number of Epochs	20
Training Set Size OP08	575 sample signals
Training Set Size OP10	742 sample signals
Length of Input Signal	4096

Table 3 Results of initial models from M02 on their test sets for operations OP08 and OP10

Model	test size	TN	TP	FN	FP	Precision	Recall	Accuracy	F-Measure
M02_OP08	247	216	24	1	6	0.8	0.96	0.9716	0.8727
M02_OP10	318	307	7	2	2	0.77	0.77	0.9874	0.77

TP: True Positives ("NOT OK", TN: True Negatives ("OK"), FP: False Positives, FN: False Negatives

and recall into a single measure that captures both properties. In the context of process health classification, True Negatives correspond to "OK" cases and True Positives to "NOT OK" cases. Accuracy results for both M02 models trained are shown in Table 3. As it can be observed both models are generally good at classifying the "NOT OK" class despite the highly imbalanced data set. In further experiments, a special attention will be put to understanding if this performance can be maintained once these models are used for further incremental learning.

At this stage, an initial semantic model with instances that captures the Bosch use case from an asset, data and ML solution point of view has been created. Two initial models have been created for two operations in machine M02. The following sections present the use of this semantic model for solution identification and for effective model learning.

Algorithm 1 ML Model Matching.

Input: Solution Type
Output: Models

```
SELECT {
?asset ?solutiontype ?model ?process ?pt .

WHERE {
?asset resourceModel:usesSolution ?application .
?application rdf:type ?solutiontype .
?application :supportsProcess ?process .
?process rdf:type ?pt .
?pt rdfs:subClassOf* pt:Drilling .
?solutiontype rdfs:subClassOf*:ManufacturingAIApp .
?application :canBeSolvedBy ?aiapplication .
?aiapplication :canBeSolvedBy ?model
}
```

4.2 Identifying models according to similarity using SPARQL

There are two scenarios that will be demonstrated through querying the ontology: (a) the case of a new operation being carried by a currently monitored machine and (b) the case where no model exists for a machine but a *similar* machine is searched for.

(a) *Existing model* — assuming a new process called OP11 starts in M02, the ontology is queried to retrieve existing models of M02. The query to obtain the models is shown in Algorithm 1, and the result is shown in Fig. 17. In the case of M02, the two aforementioned models for OP08 and one for OP10, are found.

Looking at the process parameters, both are operations that run at the same speeds and feeds. However, other similarities need to be taken into account for an effective selection of the model for "reuse". There are several distance metrics in the literature with regard to data features that can be used to establish how different is the distribution of new incoming data [66]. Distance of features such as frequency are incorporated into the semantic similarity measure of the proposed approach. This allows to identify OP10 as a better candidate as its frequency features are closer to OP11. OP10 shows higher amplitudes for "NOT OK" classes compared to OP08 which does not have any notably differences in frequencies [8]. Results to demonstrate the difference in the effectiveness of carrying out continual learning from OP08 than from OP11 will be presented in the next subsection.

(b) *Models from similar machines* — in the case where there are no existing models, for example case of M03, as this asset has a drilling capability, other assets with the same capability are searched for. To do this, inference runs the query in Algorithm 2 to obtain all assets with a similar capability. This is useful when there are other device combinations on the shop floor that might be using the same IndustrialAISolution despite not having the same physical machine features. Figure 18 shows the result of the inference. From this result, another query running the similarity metric will search for the best candidate assets which run a similar type of process and parameters. From these either a linked ML model or data available can be used for developing a new model instantiating one of the existing pipelines. These results will be presented in the following subsection.

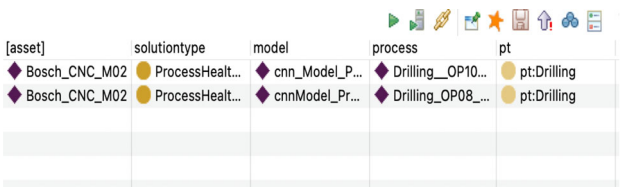

Fig. 17 Available models for M02 with their corresponding Industrial Application and process

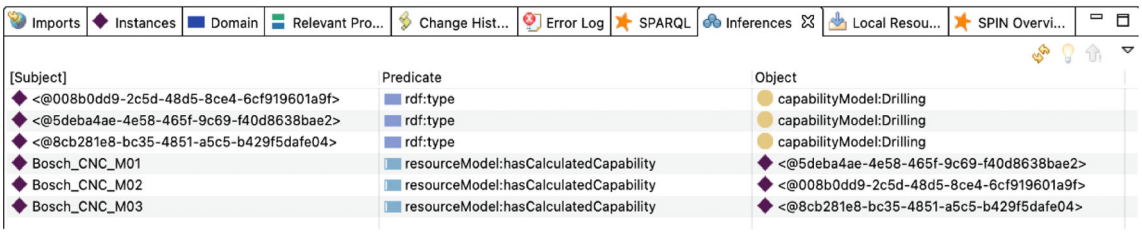

Fig. 18 Inferred drilling capabilities on the three machines

4.3 Continual learning on similar process contexts

Following from the model in OP10 identified in Subsection 4.1 (a), a continual learning strategy is used and the cnn_Model_OP10 was further trained on OP11 and then in a second experience it was trained on OP12. Each experience represents a future batch of new incoming data which has a new unseen data distribution. The continual learning framework was implemented in Avalanche python library using Elastic Weight Consolidation (EWC), with an optimised (Ray Tune) hyperparameter $\lambda = 102$. EWC is a recent continual learning approach for neural architectures that constraints how weights that are relevant for the accuracy of the current task are changed in future experiences [15]. These constraints allow to "retain" old knowledge, whilst at the same time, accommodate for new knowledge given the new incoming data. The results of applying this technique for incrementally learning from OP10 to OP11 and OP12 are presented in Table 4, where experience 1 corresponds to the initial training of the model done with OP10, the second experience corresponds to only presenting samples from OP11 to the trained model, and finally the third experience corresponds to only presenting samples from OP12. The accuracy at each experience learning of the model on the three operations is shown to demonstrate how the accuracy of the model is generally well kept despite not seeing old samples during later training experiences.

Looking at the percentage of "NOT OK" correctly classified during testing after the three experiences, OP10 obtained 9 out of 9 correct, for OP11 13 out of 15 and for OP12 12 out of 12. Results are shown in Fig. 19 as a confusion matrix, which is another way to visualise the total number of True Positives, True Negatives, False Positives and False Negatives.

To validate the selection of OP10 as a starting model rather than model OP08, another continual learning experiment was done using OP08 as starting model. The results are shown in Table 5.

As it can be seen, although generally new tasks perform well in the last experience, accuracy decreases for OP08. Figure 20 shows the confusion matrices for the three operations throughout the continual learning experiment. Although the overall accuracy of models is high, it is important to look at how it performs particularly on the minority class, which are the cases where the process is NOT OK and for which there are less samples. Compared to the results from OP10, OP08 decreases its performance for the minority class. Although from a practical point of view a larger data set containing the three operations, or even the four of them could be developed and possibly achieve a 100% accuracy, what it is important to note here is that when accuracy matters to a high degree and it is not possible to be re-training from scratch every single time, then it is important to understand to what extent data available can be useful for ensuring the robustness of the model.

Algorithm 2 Drilling Capability Inference SPIN Rule.

Input: Individual Assets and Capabilities
Output: Inferred Drilling Capabilities

```
CONSTRUCT {
?this resourceModel:hasCalculatedCapability _:b0 .
_:b0 a capabilityModel:Drilling .

WHERE {
?this a resourceModel:RealDeviceCombination .
?this resourceModel:hasIndividualDeviceOrDeviceCombination
?spindle .
?this resourceModel:hasCapability ?drillBit .
?drillBit a capabilityModel:DrillBitFunction .
?spindle resourceModel:hasCapability ?spinning .
?spinning a capabilityModel:SpinningTool .
}
```

Table 4 Accuracy of model after incrementally learning to classify the process for OP10, OP11 and OP12

Training Experience	Accuracy in OP10	Accuracy in OP11	Accuracy in OP12
EXP1	98.74%	–	–
EXP2	97.79%	99.83%	–
EXP3	**99.37%**	**99.27%**	**100%**

The bolded values indicate the best accuracy obtained, which coincides with the last learning experience

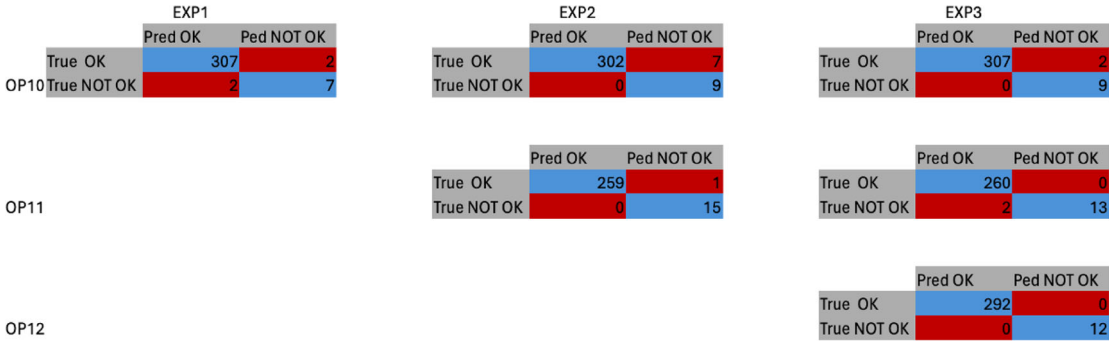

Fig. 19 Confusion matrices on test sets when learning operations incrementally through a continual learning approach. During experience one, the model learns to classify the health of the process from only using Operation 10. Experience two takes the existing trained model which in trained further only with data coming from Operation 11. Finally, during experience three, the existing model is only trained on data from Operation 12

To further demonstrate the relevance of context to effectiveness of training, a second continual learning experiment was done using a different set of operations. This time, it was trained using M01 and M02 and tested in M03, to not only verify the effectiveness of the continual approach within the same machine, but across machines. For this experiment, experience one corresponded to all samples from OP01, experience two corresponded to all samples from OP02, experience three from OP04 and experience four from OP07. Accuracy results can be seen in Table 6.

These four operations have different combinations of cutting parameters. OP01 and OP04 share the same speed=250 and feed=100, and OP02 and OP07 have speed=200 and feed=50. It can be seen in the results table that after experience 3 (OP4) the accuracy in OP01 increases significantly, and the same for OP02 which is noticeably improved after experience 4. This indicates the highly correlation of process parameters with the distribution of the data, and so improving through time the accuracy of the model. In this way it can be seen how using this information can be important to ensure adequate training and ensuring the model can improve in time. It is worth noting that although accuracy results are not as high as in the first experiment, after the four experiences, the true positives for OP02 and OP04 are 100% correctly classified, for OP01 61% and for OP07 62%. Both OP01 and OP07 generally through the four learning experiences had difficulties with true positives, but were able to achieved the highest result in the last experience. Compared to the first experiment where all operations had the same parameters, here it is demonstrated how indeed process parameters play a role in effecting knowledge acquisition. However, going back to the results from experiment one, where the expectation is that because operations have the same process parameters, all should be equally useful for robustness is not always true. And that is because signals signatures represent all sorts of different issues that may be occurring in a machine, creating a range of noise and changes to signals even when a process is healthy. For this reason, it is important that then a combined approach of process knowledge as well as data characteristics is used to make the best possible selection of training data or model to do continual learning on and deliver robust models.

Table 5 Accuracy of model after incrementally learning to classify the process for OP08, OP11 and OP12

Training Experience	Accuracy in OP08	Accuracy in OP11	Accuracy in OP12
EXP1	97.16%	–	–
EXP2	97.16%	99.63%	–
EXP3	**90.28%**	**96%**	**100%**

The bolded values indicate the best accuracy obtained, which coincides with the last learning experience

These results are critical, as the major challenge in a manufacturing shop floor is to build robustness over time without compromising accuracy in "old conditions" as well as avoiding having to retrain an algorithm with continuously growing data set. The latter is an unfeasible approach, and so it is imperative to start moving toward less complex more flexible learning approaches rather than continuing developing large complex models on limited data sets that cannot adapt.

Once of the challenges of building robust models is that it is difficult to anticipate all the different ways in which a process may change. On the one hand, new approaches in the literature need to be able to deal incrementally with these unexpected changes. On the other hand, new methods need to help the decision maker on how deployed models should be further trained to maintain the expected robustness. That decision needs to be an informed decision, to ensure the best possible outcome. That means understanding better the conditions in which existing and new data are generated, to then

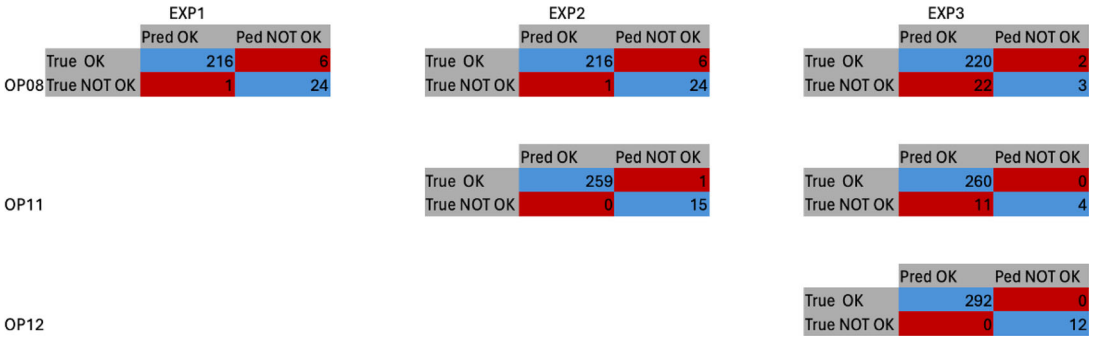

Fig. 20 Confusion matrices on test sets when learning operations incrementally through a continual learning approach. During experience one, the model learns to classify the health of the process from only using Operation 08. Experience two takes the existing trained model which in trained further only with data coming from Operation 11. Finally, during experience three, the existing model is only trained on data from Operation 12

decide when re-training an existing model is the best option of when creating a completely new one is best. The more these two processes can be automated, the closer industry can be from operationalising ML on the shop floor.

5 Conclusions and future work

This paper presents a framework and methodology to develop and reuse existing AI Industrial solutions to ensure their robustness to various types of concept drift. By capturing knowledge about the asset, the process and the ML model development, it is possible to identify approaches that work well for certain type of IndustrialAISolutions and use that to automate to some extent the development of such solutions. It was demonstrated in two experiments how having context information of process, asset and data is relevant to the quality of the results of the model during training as well as for building robustness over time in an efficient way by using a continual learning approach. Making the link between the manufacturing process and the AI solution is key and critical to AI scalability. A conceptual approach of how then to deploy and monitor these models is introduced. Future work will include the full integration with the AAS to deploy, monitor and re-deploy the models. Further experiments will include other continual learning techniques such as synaptic intelligence to determine the capabilities of such approaches when learning data that is more further apart in terms of distribution. Additional work will also look at model reusability when data quality changes (e.g. frequency) as well as studying transfer learning when changing the type of asset or completely different manufacturing operation.

Table 6 Accuracy of model after incrementally learning to classify the process for OP01, OP02, OP04 and OP07

Training Experience	Accuracy in OP01	Accuracy in OP02	Accuracy in OP04	Accuracy in OP07
EXP1	84.04%	–	–	–
EXP2	88.9%	88.8%	–	–
EXP3	87.20%	88.95%	71.99%	–
EXP4	**89.57%**	**93.14%**	**84.16%**	**97.8%**

The bolded values indicate the best accuracy obtained, which coincides with the last learning experience

Acknowledgements The authors gratefully acknowledge the support provided by the University of Nottingham through the Anne McLaren Fellowship and by UK EPSRC Elastic Manufacturing Systems (EP/T024429/1).

Author contribution All authors contributed to the study conception and design. Material preparation, experiments and analysis of results were performed by Giovanna Martínez Arellano. The first draft of the manuscript was written by Giovanna Martínez Arellano and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This research was funded by the University of Nottingham, under the Anne McLaren Fellowship program.

Data Availability This study was a re-analysis of existing data that is publicly available in the GitHub Repository https://github.com/boschresearch/CNC_Machining and originally published in https://doi.org/10.1016/j.procir.2022.04.022

Declarations

Conflict of interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your

intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Koren Y, Gu X, Guo W (2018) Reconfigurable manufacturing systems: principles, design, and future trends. Front Mech Eng 13:121–136
2. Brunoe TD, Soerensen DG, Nielsen K (2021) Modular design method for reconfigurable manufacturing systems. Procedia CIRP, vol 104, pp 1275–1279. 54th CIRP CMS 2021 - Towards Digitalized Manufacturing 4.0
3. Bortolini M, Galizia FG, Mora C (2018) Reconfigurable manufacturing systems: literature review and research trend. J Manuf Syst 49:93–106
4. Huang S, Xu Z, Wang G, Zeng C, Yan Y (2019) Reconfigurable machine tools design for multi-part families. Int J Adv Manuf Technol 105:813–829
5. Arinez JF, Chang Q, Gao RX, Xu C, Zhang J (2020) Artificial intelligence in advanced manufacturing: current status and future outlook. J Manuf Sci Eng 142(11):110804
6. Arents J, Greitans M (2022) Smart industrial robot control trends, challenges and opportunities within manufacturing. Appl Sci 12(2)
7. Morgan J, Halton M, Qiao Y, Breslin JG (2021) Industry 4.0 smart reconfigurable manufacturing machines. J Manuf Syst 59:481–506
8. Tnani M-A, Feil M, Diepold K (2022) Smart data collection system for brownfield CNC milling machines: a new benchmark dataset for data-driven machine monitoring. Proc CIRP 107:131–136
9. Maier J (2017) Made smarter review. https://www.gov.uk/government/publications/made-smarter-review
10. Lee J, Singh J, Azamfar M (2019) Industrial artificial intelligence. arXiv:1908.02150
11. Martínez-Arellano G, Terrazas G, Ratchev S (2019) Tool wear classification using time series imaging and deep learning. Int J Adv Manuf Technol 3647–3662
12. Zhou Y et al (2022) A new tool wear condition monitoring method based on deep learning under small samples. Measurement 189:110622
13. Zhou Y, Sun W (2020) Tool wear condition monitoring in milling process based on current sensors. IEEE Access 8:95491–95502
14. Pratama M, Dimla E, Lai CY, Lughofer E (2019) Metacognitive learning approach for online tool condition monitoring. J Intell Manuf 30:1717–1737
15. Kirkpatrick J et al (2017) Overcoming catastrophic forgetting in neural networks. Proc Nat Acad Sci 114(13):3521–3526
16. Lesort T et al (2020) Continual learning for robotics: definition, framework, learning strategies, opportunities and challenges. Inf Fusion 58:52–68
17. Huyen C (2022) Designing machine learning systems (" O'Reilly Media, Inc.")
18. Zhou B, Pychynski T, Reischl M, Kharlamov E, Mikut R (2022) Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding. J Intell Manuf 33:1139–1163
19. Gerling A et al (2022) Comparison of algorithms for error prediction in manufacturing with automl and a cost-based metric. J Intell Manuf 33:555–573
20. Nasir V, Sassani F (2021) A review on deep learning in machining and tool monitoring: methods, opportunities, and challenges. Int J Adv Manuf Technol 115(9–10):2683–2709
21. Marei M, Zaatari SE, Li W (2021) Transfer learning enabled convolutional neural networks for estimating health state of cutting tools. Robot Comput Integr Manuf 71:102145
22. Garza A, Mergenthaler-Canseco M (2023) Timegpt-1. arXiv:2310.03589
23. Zhang H et al (2023) Large scale foundation models for intelligent manufacturing applications: a survey. arXiv:2312.06718
24. Frye M, Krauß J, Schmitt R (2021) Expert system for the machine learning pipeline in manufacturing. IFAC-PapersOnLine 54(1):128–133. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021
25. Zhou B et al (2021) Scaling usability of ml analytics with knowledge graphs: exemplified with a bosch welding case
26. Zhou B et al (2021) Semml: facilitating development of ml models for condition monitoring with semantics. J Web Semant 71:100664
27. Mourtzis D, Doukas M (2014) Knowledge capturing and reuse to support manufacturing of customised products: a case study from the mould making industry. Proc CIRP 21:123–128. 24th CIRP Design Conference
28. Järvenpää E, Siltala N, Hylli O, Lanz M (2019) The development of an ontology for describing the capabilities of manufacturing resources. J Intell Manuf 30(2):959–978
29. Järvenpää E, Siltala N, Hylli O, Lanz M (2018) Product model ontology and its use in capability-based matchmaking. Proc CIRP 72:1094–1099
30. Mo F, Chaplin JC, Sanderson D, Martínez-Arellano G, Ratchev S (2024) Semantic models and knowledge graphs as manufacturing system reconfiguration enablers. Robot Comput Integr Manuf 86:102625
31. Braga J, Dias JLR, Regateiro F (2021) A machine learning ontology. Preprint, Frenxiv 5:2–10
32. Sacha D, Kraus M, Keim DA, Chen M (2019) Vis4ml: an ontology for visual analytics assisted machine learning. IEEE Trans Vis Comput Graph 25(1):385–395
33. Mäkinen S, Skogström H, Laaksonen E, Mikkonen T (2021) Who needs mlops: what data scientists seek to accomplish and how can mlops help?. arXiv:2103.08942
34. Zhao Y (2021) MLOps scaling ML in an industrial setting. Ph.D. thesis, PhD thesis, University of Amsterdam
35. Raffin T, Reichenstein T, Werner J, Kühl A, Franke J (2022) A reference architecture for the operationalization of machine learning models in manufacturing. Proc CIRP 115:130–135
36. John MM, Olsson HH, Bosch J (2021) Towards mlops: a framework and maturity model
37. Faubel L, Schmid K, Eichelberger H (2023) Mlops challenges in industry 4.0. SN Comput Sci 4(6):828
38. Rauh L et al (2022) Towards ai lifecycle management in manufacturing using the asset administration shell (AAS). Proc CIRP 107:576–581
39. Ruediger-Flore P, Glatt M, Hussong M, Aurich JC (2023) Cad-based data augmentation and transfer learning empowers part classification in manufacturing. Int J Adv Manuf Technol 125(11):5605–5618
40. Lyu P, Zhang H, Yu W, Liu C (2022) A novel model-independent data augmentation method for fault diagnosis in smart manufacturing. Proc CIRP 107:949–954
41. Li Y et al (2021) Augmented time regularized generative adversarial network (ATR-GAN) for data augmentation in online process anomaly detection. IEEE Trans Autom Sci Eng 19(4):3338–3355
42. Bui V, Pham TL, Nguyen H, Jang YM (2021) Data augmentation using generative adversarial network for automatic machine fault detection based on vibration signals. Appl Sci 11(5):2166
43. Wang P, Gao RX (2020) Transfer learning for enhanced machine fault diagnosis in manufacturing. CIRP Ann 69(1):413–416

44. Giannetti C, Essien A (2022) Towards scalable and reusable predictive models for cyber twins in manufacturing systems. J Intell Manuf 1–15
45. Mendez JA, Eaton E (2022) How to reuse and compose knowledge for a lifetime of tasks: a survey on continual learning and functional composition. arXiv:2207.07730
46. Lin C-C, Deng D-J, Kuo C-H, Chen L (2019) Concept drift detection and adaption in big imbalance industrial IoT data using an ensemble learning method of offline classifiers. IEEE Access 7:56198–56207
47. Fisher OJ, Watson NJ, Escrig JE, Witt R, Porcu L, Bacon D, Rigley M, Gomes RL (2020) Considerations, challenges and opportunities when developing data-driven models for process manufacturing systems. Comput Chem Eng 140
48. Janowicz K, Haller A, Cox SJ, Le Phuoc D, Lefrançois M (2019) Sosa: a lightweight ontology for sensors, observations, samples, and actuators. J Web Semant 56:1–10
49. Munaro R, Attanasio A, Del Prete A (2023) Tool wear monitoring with artificial intelligence methods: a review. J Manuf Mater Process 7
50. Ross NS et al (2024) A novel approach of tool condition monitoring in sustainable machining of NI alloy with transfer learning models. J Intell Manuf 35(2):757–775
51. Aghazadeh F, Tahan A, Thomas M (2018) Tool condition monitoring using spectral subtraction and convolutional neural networks in milling process. Int J Adv Manuf Technol 98:3217–3227
52. Kothuru A, Nooka SP, Liu R (2018) Audio-based tool condition monitoring in milling of the workpiece material with the hardness variation using support vector machines and convolutional neural networks. J Manuf Sci Eng 140(11):111006
53. Marani M, Zeinali M, Songmene V, Mechefske CK (2021) Tool wear prediction in high-speed turning of a steel alloy using long short-term memory modelling. Measurement 177:109329
54. Shah M et al (2022) Tool wear prediction in face milling of stainless steel using singular generative adversarial network and LSTM deep learning models. Int J Adv Manuf Technol 121(1):723–736
55. Wang M, Zhou J, Gao J, Li Z, Li E (2020) Milling tool wear prediction method based on deep learning under variable working conditions. IEEE Access 8:140726–140735
56. Rathee P, Malik SK, Dutta P, Bhattacharya A, Dutta S, Lai W-C (eds) (2023) An analysis of semantic similarity measures for information retrieval. Dutta P, Bhattacharya A, Dutta S, Lai W-C (eds) Emerging technologies in data mining and information security
57. Ichise R (2008) Machine learning approach for ontology mapping using multiple concept similarity measures
58. Rada R, Mili H, Bicknell E, Blettner M (1989) Development and application of a metric on semantic nets. IEEE Trans Syst Man Cybern 19(1):17–30
59. Richardson R, Smeaton A, Murphy J (1994) Using wordnet as a knowledge base for measuring semantic similarity between words
60. Hirst G, St-Onge D et al (1998) Lexical chains as representations of context for the detection and correction of malapropisms. WordNet: Electron Lexical Database 305:305–332
61. Resnik P (1999) Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. J Artif Intell Res 11:95–130
62. Tversky A (1977) Features of similarity. Psycol Rev Am Psycol Assoc Inc 84(4):104–106
63. Harispe S, Ranwez S, Janaqi S, Montmain J (2015) Semantic similarity from natural language and ontology analysis. Springer
64. Chen Z, Liu B (2018) Lifelong machine learning vol 1. Springer
65. Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller P-A (2018) Data augmentation using synthetic data for time series classification with deep residual networks. arXiv:1808.02455
66. Zhuang F et al (2020) A comprehensive survey on transfer learning. Proc IEEE 109(1):43–76

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.