A Data Fusion Service-Oriented Infrastructure for Production Line Monitoring

Sebastiano Gaiardelli*, Nicola Dall'Ora*, Francesco Ponzio†, Enrico Fraccaroli*,
Franco Fummi*, Santa Di Cataldo†, Sara Vinco†
*Department of Engineering for Innovation Medicine – University of Verona, Italy, name.surname@univr.it
†Department of Control and Computer Engineering – Politecnico di Torino, Italy, name.surname@polito.it

***Abstract*—The Industry 4.0 paradigm has deeply changed classical manufacturing by introducing data-based analytics and decision-support strategies. At the state of the art, data used for manufacturing monitoring is mostly originated by sensors, that undergo a fusion step to align different data sources. However, this data is only relative to the monitored process, and it does not include the corresponding operating conditions and parameters, that are known by the Manufacturing Execution System (MES). Such information is currently either not included or labeled by hand, thus incurring in errors and limiting the amount of available labeled data. To overcome this issue and go beyond the sole data fusion of sensor data, this paper proposes an infrastructure that automatically label time series generated by sensors with information extracted from the MES, to achieve enhanced monitoring of the production process. The relevance of the proposed solution and the possibilities opened by its application are stressed with the application to a robotic arm.**

***Index Terms*—Industry 4.0, Industrial IoT sensors, Data fusion, Process monitoring, Anomaly detection.**

I. Introduction

Manufacturing over the past decade has invested massive budgets in the digitalization of factory equipment [1]. Sensors supported by Industrial Internet of Things (IIoT) infrastructures can now collect and analyze large quantities of data rapidly and accurately, thus reducing human error, and enhancing production quality [2]. After collection, data originated by different sensors are typically combined and aligned in a process called *data fusion*, which aims to produce more consistent, accurate, and useful information reflecting the current status over time of the monitored equipment. The ultimate goal of this process is to provide a sound database for the application of data analytics techniques, and especially Machine Learning (ML) approaches, to extract relevant information and patterns, ensure lifelong process monitoring, identify and react to anomalous behaviors and unexpected conditions, and reduce machine downtime [3]. Nonetheless, the application of such techniques typically requires a *time-consuming manual annotation of time series* to relate measured quantities with information about production parameters or to distinguish regular from anomalous behaviors [4], [5]. This manual step introduces many potential pitfalls, as labels may be incorrect or misaligned in time or applied to few samples, thus not allowing effective knowledge extraction. This article proposes an important step forward in data management of manufacturing lines to go beyond the aforementioned limitations. The key idea is the development of a *Data Integration HUB (DIH)*, depicted in Figure 1, that automatically aligns sensor time series and correlates them with information about the production process, extracted from the MES. The generated data reconstructs a complete view of the system state over time, including sensed data and MES production configuration. To achieve this, the solution proposed in this article:

1) *merges data originated by sensors* transmitted at different sampling frequencies and with different protocols through data fusion and the adoption of a global timestep, thus achieving homogeneous data traces;
2) *automatically labels data with information extracted from the MES related to the production recipe*, i.e., machine settings used by equipment, performed actions (*i.e.*, pick), and relative parameters (*i.e.*, speed) [6].
3) applies both steps automatically to avoid manual intervention and potential errors, and to *easily produce extensive datasets*, more suited for an effective application of data analytics to extract relevant information.

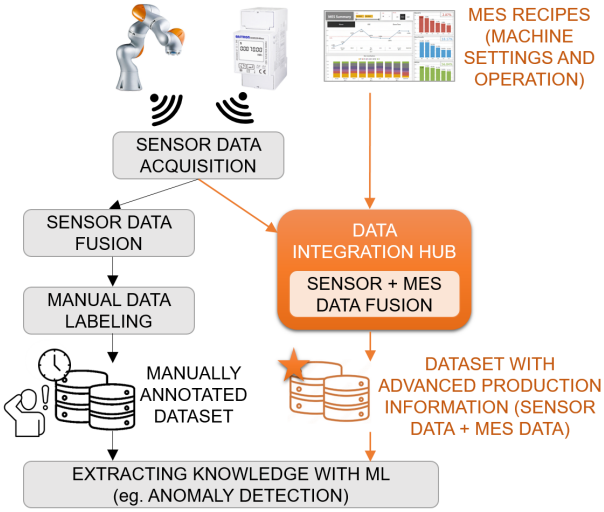

Figure 1. Overview of the proposed data fusion infrastructure that fuses sensor data with recipe data extracted from the MES inside the DIH module, avoiding time-consuming and error-prone manual annotation of the dataset, allowing enhanced extraction of knowledge with data analytics and ML.

The work has been partially supported by the PRIN 2022T7YSHJ SMART-IC - Next Generation EU project. Furthermore, this study has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 101109243.

Copyright © 2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

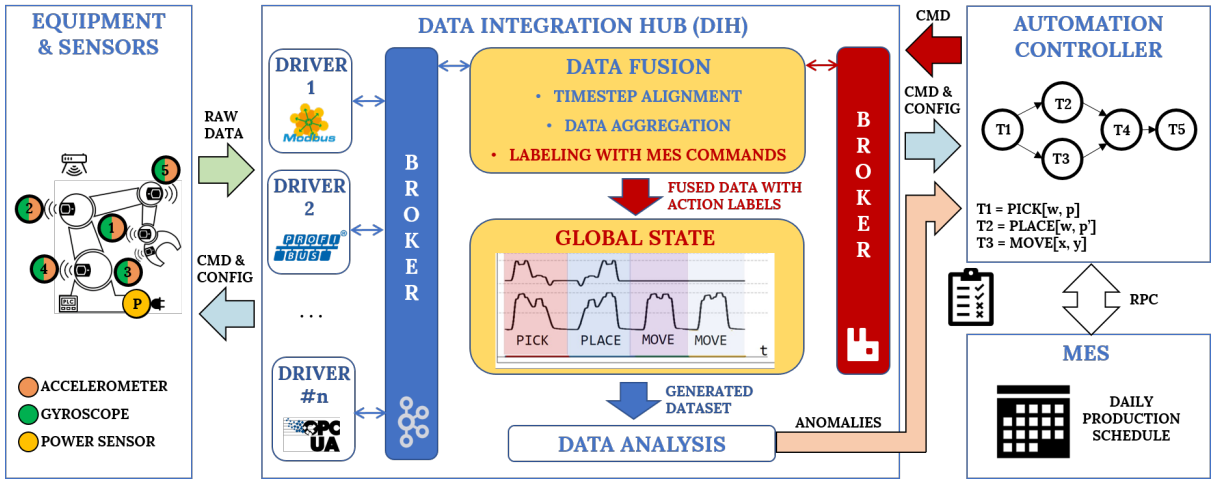

Figure 2. Role of the DIH. Given inputs data coming from sensors (potentially at different frequencies and through different protocols, left) and commands originating from the Automation Controller (with relative parameters, right top), the DIH fuses all data traces by aligning the timestamps and by automatically annotating sensor data with production information. The parts highlighted in red are the novelties introduced w.r.t. sensor data fusion.

II. Background and State of the Art

Advanced data fusion architectures are necessary in modern manufacturing plants to collect and exploit data to enhance the governance of the manufacturing system in all production phases [7]. In this scenario, data fusion focuses on aggregating data from different sensors, to generate a coherent state estimation of the manufacturing equipment. The industrial data fusion approaches can be categorized based on the main target of each specific architecture [8]. The most common targets are control and optimization of the manufacturing system schedule [9], [10], ensemble learning of production applications [11], [12], and the monitoring of machinery status: anomaly detection, chatter detection [13], wear detection [14], and prognosis applications [15].

In a typical industrial scenario, raw data is generated by IIoT sensors placed within the manufacturing system. Such data streams are called time series, as they sample given quantities (e.g., an acceleration, a pressure) over time. The sensed values are transmitted by the sensors through a dedicated industrial communication networks [16], and then redirected through specific local or distributed architectures responsible for storing and managing large data series [17], creating the so-called *data lakes*. As a subsequent step, the proper *data fusion* occurs, as the process that correlates and combines data originated by different sensors by abstracting from the specific data format and communication protocol, to build a coherent database for the subsequent application of knowledge-extraction algorithms [7]. This step is crucial to allow the correct estimation of the state of the industrial scenario over time, and a coherent and sound application of learning and monitoring algorithms. A limitation of such information flow is that the focus is on the sensor-generated data: all information relative to production management, stored in the company MES, are not fused with the sensor data. This is a limitation, as it does not allow aligning sensor data with the production parameters that generated such conditions.

III. IIoT and MES Data Fusion

This paper aims to go beyond the traditional idea of data fusion, where the MES is a mere receiver of sensor-generated data rather than an active actor in the data generation process. The data fusion infrastructure is thus extended to take into account not only sensor data but also data available in the production MES, that describes production settings. Such MES-generated data is used as a descriptor of the production steps, and aligned with the correspondingly generated sensor samples to derive a *global state* of the manufacturing system. The explanation of the different parts of the proposed methodology follows the schema depicted in Figure 2.

A. Communication with sensors and equipment

With the term *smart sensors*, we identify those that are characterized by high heterogeneity in terms of monitored phenomena (*i.e.*, accelerators, acoustic emissions), sampling frequency (*i.e.*, seconds, milliseconds), technology, and communication protocols (*i.e.*, cabled or wireless) [16], [18].

Let us focus on the sources of data coming from the sensors placed on a robotic arm, as shown in the left-hand side of Figure 2. From this piece of equipment, we can detect acceleration, rotation, and power values. This high degree of heterogeneity creates a big issue for data management, as each smart sensor produces data at different timescales and with different protocols: all these factors contribute to increasing the complexity of collecting data and deriving a global state of the manufacturing system. Additionally, monitoring a manufacturing system requires collecting information from *digitalized production machines* with built-in sensors and support machine-to-machine communication, typically using OPC UA protocols [19].

The first challenge encountered by the DIH is thus the collection of data from heterogeneous data sources. This is realized with the typical data collection architecture, implemented as a centralized broker that communicates with a number of *drivers* that directly communicate with the sensors.

a) Protocol-specific drivers: The drivers communicate with the sensors with the suited sampling frequency, technology, and communication protocol. Their goal is to allow sensor- and machine-specific communication and to make it available to the subsequent blocks of the DIH infrastructure. It is important to note that new drivers can be added whenever necessary, thus guaranteeing the extensibility and flexibility of the infrastructure. Examples of protocols range from wired ones (*i.e.* Profibus, ModBus) or Ethernet-based ones (*i.e.* Modbus/TCP, Profinet) to lightweight wireless protocols for resource-constrained devices (*i.e.* Wi-Fi, ZigBee) [20].

b) Centralized broker: Sensor data is then collected in a centralized broker through a publish-subscribe approach: drivers act as publishers, that send messages (*i.e.* sensed data) to the centralized broker with no indication of specific receivers. This allows to easily aggregate data received from different sensors in a single repository, still guaranteeing high throughput and accessibility and low communication latency. A typical broker used for sensor data management is the *Kafka broker*, suited for environments with low latency and high throughput are primary concerns [21].

It is important to note that this approach enables bidirectional communication: sensors may indeed expose setup and reconfiguration services, allowing run-time changes of their data publishing frequency on demand, *i.e.*, to avoid bandwidth over-saturation by reducing the frequency of *idle* machines.

B. Communication with the MES

To allow the fusion also of data generated from the MES, it is necessary to build a similar communication infrastructure also towards the MES itself. Production information extracted from the MES, including machine state (*i.e.*, performed actions) and relative parameters (*i.e.*, speed, object weight), is indeed crucial to allow the construction of a global state of the manufacturing system.

The target MES is an extended MES, as proposed by [10], that extends the standard capabilities with advanced automated features (right of Figure 2), such as reconfiguration of the production line, autonomous execution of production orders, resource management, and advanced scheduling. This introduces, in fact, an intermediate software component between the MES and the DIH, called *Automation Controller* in Figure 2. The Automation Controller communicates with the MES through Remote Procedure Call (RPC) interfaces to navigate through the MES configuration and to notify any actions, such as execution of operations and reconfiguration. This allows the creation of custom logic, such as scheduling policies, automatic execution, and automatic feedback from the production plant.

The communication between the DIH, the Automation Controller, and the MES reflects the organization of the communication with the sensors. A broker connects the DIH to the Automation Controller. In this case, the broker is *RabbitMQ* that relies on a request-response message-based communication protocol where producers publish messages into a queue [22] that supports delivery acknowledgment and permission-based security, and it supports RPC.

The Automation Controller communicates with the production machines through the DIH by publishing to the RabbitMQ broker its parameters and all its *state* changes, identified by `CMD` in Figure 2. These states represent the *evolution* of a command, as seen from the Automation Controller perspective:

- `Init`: when the Automation Controller has sent the command to a machine, but the machine's response is still unknown;
- `Start`: corresponds to a positive response from the machine for the received command, and thus to the beginning of its execution;
- `End`: sent by the machine to identify the end of the received command;
- `Error`: notifies if the machine or a component monitoring the machine raises an error that requires human intervention. This state can only derive from external inputs, generated by the machine itself, a human operator, or anomaly detection components, and it identifies a specific condition in which a rescheduling policy can be activated as a reaction to an unexpected event;
- `Failed`: if the machine rejects the received command.

This information can be extremely useful when associated with the correspondingly sensed measurements (originated by sensors) and to other machine information (provided by production machines), as it allows to correlate sensed data with the production parameters that determine such evolution. Thus, this paper aims to collect, fuse, and handle such information that is normally not exploited for more than the simple activation of production machinery.

C. Data Fusion

The goal of this block is to implement *data fusion*, that is, to make a comprehensive merge of several sources of data extracted from multiple sources. This approach extends the classical definition of data fusion, which generally addresses data generated from sensors (either external or embedded in the most advanced production machines). The idea is to fuse both sensor data and data available in the MES to provide more reliable and accurate information and build a global state of the manufacturing system.

This problem entails three challenges: 1) deriving a common notion of time to associate data items referring to the same instant, 2) bringing sensor data to a common format, and 3) annotating sensory data and machine-generated data with information coming from the MES and the Automation Controller. It is important to note that applying such data transformations manually is not feasible due to the high volume of data, its frequent generation, and the inherent data conversion challenges.

1) Timestamp alignment: Distributed architectures, such as the proposed DIH, do not have a shared notion of time and, therefore, a unique global timestamp. We address this problem by exploiting the Precision Time Protocol (PTP) [23], a state-of-the-practice industrial clock synchronization protocol

that guarantees microsecond clock accuracy. Due to its low-computation requirements and software implementation, PTP is also supported by constrained devices, making it the best option for a manufacturing system. This allows to align data generated by different sources temporally, and to associate a timestep to the global state information, computed as a result of the data fusion steps.

2) Data aggregation: The next step is to aggregate sensor data in a common format to preserve only measurement data and remove protocol-specific information. Data is merged with data fusion approaches to build local status information, here, specific to a single device or production line [8]. The data is generated with the frequency of the highest frequency sensor, as not to miss any update; successive downsamplings can be applied at later stages during the application of data analytics. In each record, all data measurements recorded at the relative time step are listed together with the time step. If no new data is available for a data source (e.g., a lower frequency sensor), the relative sample is equal to the latest measurement available as still considered valid.

3) Labeling with MES information: The last step is to associate sensor-based data with the states generated by the Automation Controller and the MES. The goal is to associate the fused sensor information with the corresponding machine commands and related parameters as an automatically generated label. This goes beyond traditional data fusion approaches that focus only on sensor-generated data. Additionally, it allows to infer meaningful knowledge about the relationship between machine status and operation and the corresponding physical evolution, as detected by the sensors.

The first step in this direction is to associate to each machine the corresponding sensors. This requires the knowledge of the *hierarchical structure* of the production plant. Such information is stored in an ISA-95 hierarchy model of the manufacturing site, which is stored in the MES and contains a view of the manufacturing site as equipment in each working unit, plus correspondingly installed sensors.

The second step is to detect the time window in which each command issued for the machine is active. As explained in the former section, the boundaries of this time window for a given command are identified by the `Init` state (starting the command) and either the `End` state or the `Failed` state, as the corresponding end event.

Then, all rows in such time window are labeled with the command plus command-specific information, *i.e.*, speed, weight. As a result, all data related to a piece of equipment are aggregated into *a single row, identifying a local state* described as: global timestamp, current machine commands with relative state and parameters, and for each sensor, its last value received.

D. Generated fused dataset

The resulting generated data is a dataset where each record lists the state of all monitored equipment, reporting the global timestamp, and for each monitored piece of equipment: the value of sensors and machine information, plus any commands (with relative parameters) issued by the Automation Controller and the MES. This creates a dataset where measured data is automatically labeled with machine commands and configuration, avoiding manual intervention and guaranteeing correct temporal correlation of the aggregated information. The dataset can be saved on a timeseries database, like InfluxDB, suitable for storing time series of data coming from heterogeneous sources and for performing real-time analytics.

E. Data analytics

The characteristics of the generated dataset make it very promising for the application of subsequent data analysis techniques, *i.e.*, based on ML techniques that are widely adopted for dealing with a variety of production line problems [3]. The Data Analytics block of Figure 2 is not novel per se, but it can rather be filled with any data analytics technique available at the state-of-the-art, *i.e.*, for detecting product failure and outliers w.r.t. the normal operation, for filtering noisy or repetitive data points and highly correlated or irrelevant features, or for reducing the size of the data collected over time for the manufacturing system. The novelty here lies on the dataset, that provides sensor data automatically labeled with MES-generated information. Given that the results obtained with data analytics strictly depend on the quality of the information that can be extracted from the collected data [11], this opens interesting scenarios, as a large amount of data is made available with reliable and meaningful labels.

F. Implementation

The proposed DIH is implemented following the *microservices* paradigm, *i.e.*, as a suite of small services, each running in its own process and covering one of the various blocks of the DIH architecture. Each component of the proposed DIH is containerized and deployed in a cluster managed by Kubernetes, a portable, extensible, open-source platform for managing containerized services [17]. This approach allows for enhancing modularity, scalability, future extensibility, and integration of new sensors and devices, as new protocols can be simply supported by adding the respective drivers. This allows taking full benefit from a parallel architecture, both for storage and performance matters.

IV. Application to an industrial robotic arm

The proposed infrastructure enabling data fusion has been implemented within an industrial research laboratory consisting of a full-fledged production line in which all different types of industrial processing are present, *i.e.*, additive manufacturing, subtractive manufacturing, collaborative robots, and different types of material transport between different processing stations. The whole architecture is applicable to different industrial machinery; in this work, it is exemplified on a collaborative anthropomorphic manipulator, a Kuka Lightweight Robot (LR), with seven degrees of freedom.

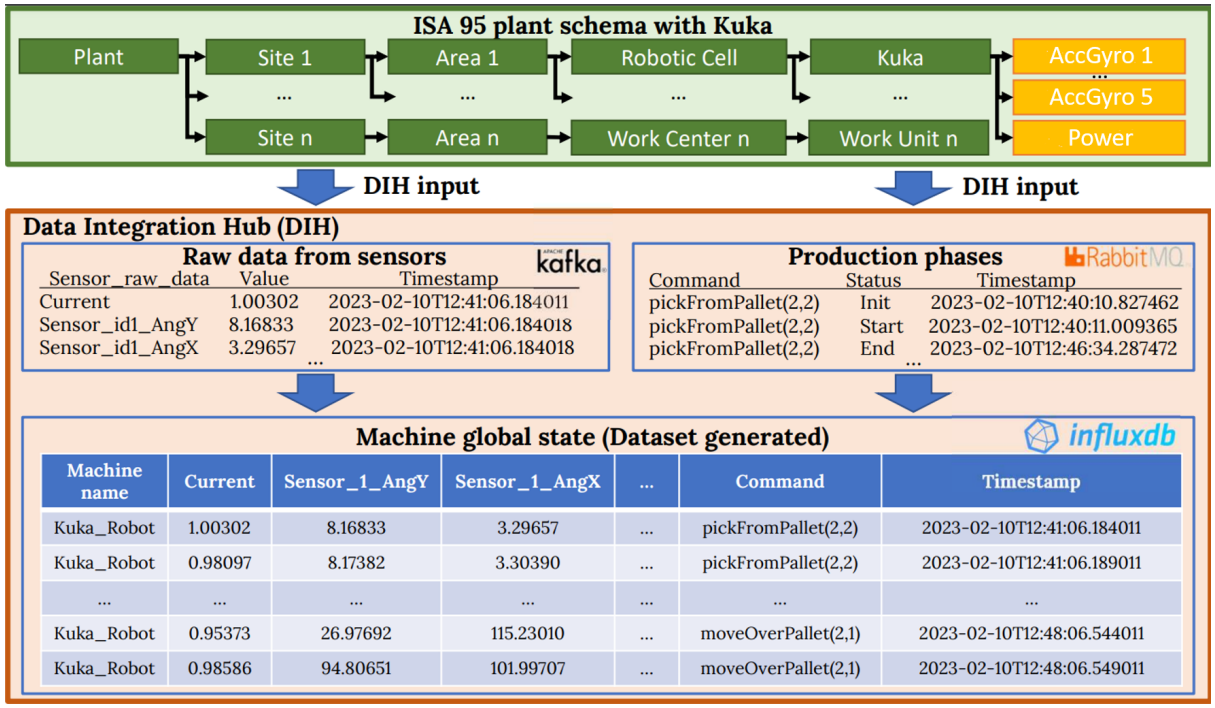

Figure 3. Pictorial representation of the data fusion process exemplified on the Kuka robotic arm. The ISA-95 equipment hierarchy model (top) is used to derive the hierarchical structure of the plant to accurately correlate MES commands for the Kuka's arm with the connected sensors (yellow nodes). Sensed data (generated by the energy monitor and two accelerometers, middle left) and MES commands for the Kuka robot arm (middle right) are temporally aligned. Data relative to the same timestamp are then fused in one record containing: the machine name, sensed data measurements, current machine commands with relative state and parameters, and global timestamp (bottom).

A. Communication with sensors and the Kuka's arm

The Kuka has been augmented with a number of sensors, depicted on the left-hand side of Figure 2:

- five 6-axis motion sensors equipped with an accelerometer and a gyroscope, which are installed in correspondence to the joints and collect acceleration, orientation, and angular velocity at 200 Hz;
- a single-phase energy meter collecting current, power, and energy at 1 Hz.

The chosen sensor types monitoring the Kuka manipulator belong to the different classes of sensors that can be exploited for monitoring an electric motor defined in the ISO 17359 specification [24]; as such, their installation is compliant with the guidelines defined by the ISO 13373-1 standard [25].

The accelerometers transmit data wirelessly by publishing it on the MQTT broker, while the energy meter transmits via Ethernet with Modbus. The Kuka robot hosts an OPC Unified Architecture (OPC UA) server, which is used to communicate with the DIH and convert MES high-level commands into specific arm movements. A Kafka broker is used to collect all data and make it available to the DIH infrastructure.

B. Communication with the MES

The production line is managed by a Siemens OPCenter Execution Discrete MES, extended with the Automation Controller proposed in [10]. The Automation Controller controls the Kuka manipulator by sending high-level commands corresponding to arm movements. The commands are sent to the RabbitMQ broker, while communication between the MES and the Automation Controller occurs via RPCs.

C. Data Fusion

The data fusion process for the Kuka robot arm is exemplified in Figure 3. The ISA-95 equipment hierarchy model (top of the figure) is used to identify which sensors are on board the Kuka robot arm (right-hand side of the hierarchy, in yellow) and to filter Kuka-related commands. Then, data traces relative to Kuka sensors (middle left) and the MES commands (middle right) are temporally aligned by using the global timestamp. The complete dataset is then constructed by creating one record every 5ms (corresponding to the highest sensor frequency, 200 Hz) by reporting: the machine name (*i.e.*, Kuka), the data measurements sensed at the current global timestamp (or latest measured value, if the data has a lower generation frequency), current machine command with relative state and parameters, and global timestamp (bottom). This allows reconstructing the global state of the Kuka robot arm in all its aspects, *i.e.*, to automatically correlate measured data originated from different sensors with MES commands.

As an example, the first rows of the global dataset are labeled with `pickFromPallet(2,2)` as the time step of such sensor measurements is between the time at which Kuka status changes to `Init` and the time at which it moves to `End` for the `pickFromPallet(2,2)` command issued by the MES. This allows to correlate sensor measurements with the pick operation.

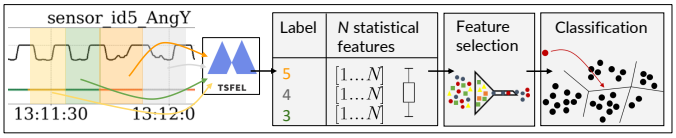

Figure 4. Overview of the data analytics pipeline used for dataset validation. The raw signals (*i.e.*, *sensor_id5_AngY*) are windowed based on the robot's actions and associated with corresponding labels. Then, signals are fed into the TSFEL library to extract statistical features, which are reduced to a 50-dimensional set by feature selection. The final feature set is fed into a classifier.

Once the global state of the Kuka robot is reconstructed, multiple actors can exploit the enclosed information by extracting additional knowledge and implementing additional data analysis. The Automation Controller can query the database at runtime through the RabbitMQ broker to apply data-aware policies. At the same time, the data analytics module can benefit both from runtime data generation and from the collection of historical data, to train the algorithms.

In the following sections, all analytics are built on an initial dataset containing 3 hours of data collection.

D. Analysis of the fused global dataset

To prove the relevance and robustness of the generated dataset, we exploit data analytics approaches, that prove whether the dataset allows the extraction of any knowledge about the operation of the Kuka robot arm.

1) From sensors to features: Data analytics are not applied to raw signals but rather to numerical features that can be processed while preserving the information in the original dataset. As visually represented in Figure 4, each raw signal is appropriately windowed over time according to the corresponding action of the robot. Then, we extract a set of 1,980 statistical features exploiting the TSFEL Python package.

To ease the following data analytics, the extracted features undergo feature standardizationand feature selection, through the Minimum Redundancy Maximum Relevance (mRMR) algorithm [26], that reduces the features to a set of 50 features mutually and maximally dissimilar, thus representing more effectively the target class.

2) Machine states classification: The subsequent step is to *demonstrate the consistency of the relation between the sensor data and the corresponding MES commands*. To achieve this result, we reduced it to a classification problem, where the commands represent the class labels. The question is, is it possible to infer the MES command given the sensed data? If so, then we can reasonably consider the relation solid.

In our experiments, we put into effect five different state-of-the-art classifiers: RandomForest (RF), Support Vector Machine (SVM), and AdaBoost (AB), implemented with the Scikit-learn library, plus Multilayer Perceptron (MLP) and Bayesian Multilayer Perceptron (MLP-B), built on top of the Keras framework. Hyper-parameters optimization is performed through randomized search [27], and we implement a standard 5-fold cross-validation strategy.

Figure 5 provides the mean cross-validation accuracy of the classifiers at different sampling rates of the sensed data

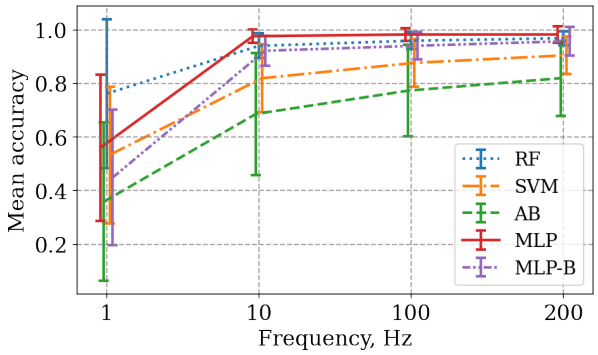

Figure 5. Mean cross-validation accuracy of different machine state classifiers at different sampling rates of the sensed data. Error bars indicate the standard deviation of accuracy among different classes.

(respectively: 1, 10, 100, 200 Hz), by reproducing different data collection settings of the overall infrastructure. The error bars in the graph indicate the standard deviation of the accuracy among the different classes (that is: the shorter the bars, the more balanced the classification accuracy).

As can be seen from Figure 5, all the tested classifiers are successful, with increasing accuracy values at increasing sampling frequency. MLP is the one that provides the highest and most balanced mean accuracy values (about $97.6 \pm 2.5\%$ at 10 Hz). The most important result of this experiment is that *the relation between sensor data and MES labels is verified*, and the dataset annotated with information extracted from the MES constructs a solid base for data analytics approaches.

Interestingly, the accuracy curve presents a clear elbow point at 10 Hz, which provides a very useful indication of the best compromise between sampling rate and classification performance: at this sampling rate, most of the tested classifiers have mean accuracy above 80%. This indicates that the data collection infrastructure can be reconfigured to operate at 10Hz, thus *reducing sampling frequency*. This allows applying the proposed solution also to older production lines, where the pre-existing sensors work at lower frequencies, and to moderate traffic for production control, not affecting the normal and safety-critical operations of the production line.

Given the obtained results, we will focus the downstream analysis on raw signals always sampled at 10 Hz. Obtaining data at this frequency is possible by configuring the sensors and the OPC UA servers accordingly, as enabled by the bidirectional communication of the Kafka broker.

3) From features to sensor: The last step is to understand which sensors and/or specific physical properties had a higher impact on the classifier decisions, to provide better insights on the relevance of the sensed data to machine state prediction. This is important to guide optimization of the sensing infrastructure (*i.e.*, by removing/replacing irrelevant sensors).

We adopt the following strategy to circumvent the intrinsic *black-box* nature of ML techniques. We start from the top 50 features obtained after the feature selection step, and we reconstruct the association to their raw input signal and the corresponding sensor and physical property. Thus, we group

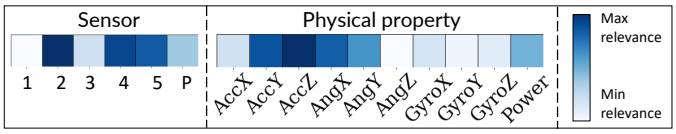

Figure 6. Sensors/physical properties color-mapped by their relevance. The sensor positioning is depicted in the bottom-left of Figure 2.

feature occurrences by the sensor/physical property. The ratio is: the higher the occurrence of features derived from a given sensor/physical property, the higher its relevance for the machine states classification task.

The obtained results, after scaling, are represented in Figure 6, where darker color means higher feature occurrence (and hence, higher relevance) of the corresponding sensor. As can be observed, accelerometers 2, 4, and 5 (position visible in Figure 2) provide most of the information to the classifier, while the most relevant physical properties appear to be the acceleration and angle of the machine arms. Vice versa, accelerometer 1, as well as all gyroscopes data, seems to have a much lower relevance to the task. This gives relevant feedback for constructing the production line, as sensors may be configured to extract only relevant data, thus reducing bandwidth occupation. Less relevant sensors can be removed and installed on other parts of the line, without losing essential information. While the specific outcome of this analysis is not generalizable, because it is tied to the classification task and commands that were analyzed, it indeed provides a relevant example of how the data fusion infrastructure can be exploited with data analytics to support the configuration and optimization of the production line.

E. Anomaly detection

Finally, we propose an example of implementation of the data analytics box in Figure 2. A typical task applied for manufacturing monitoring is anomaly detection, *i.e.*, the *early detection of anomalous behaviors of the production line*, that is crucial to avoid severe deterioration in productivity and prevent dangerous accidents [28]. Anomaly detection is still a challenging open problem because anomaly data records are (i) rare; (ii) irregular; (iii) typically lacking prior information, and (iv) hardly predictable [29], [30]. In this regard, training a supervised ML algorithm to recognize anomalies would require a huge amount of training data, with many pre-annotated anomalies as the target classes. Such a dataset is extremely cumbersome to collect and more often unfeasible in the early life of the production line when the equipment is still behaving correctly most of the time [28].

On top of these considerations, we chose a different approach in our work, which does not require specific supervision of the anomalies. We start from the machine state classifier described in the previous sections, which is built on top of data without anomalies. Then, we exploit this model to indirectly identify anomalous behaviors at inference time. Such a learning paradigm, where a supervised model learns a *pretext task* (in this case, machine states classification) to solve a different *downstream task*, is generally known in the literature as Self-Supervised Learning (SSL) [31].

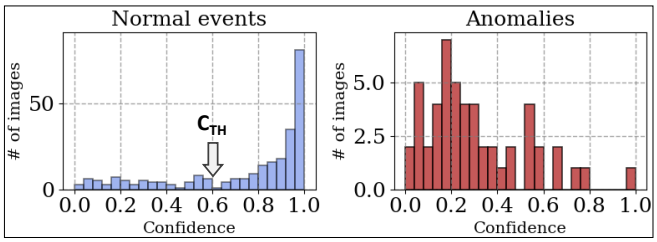

Figure 7. Distribution of the confidence values (MSR) of MLP-B predictions for the normal events (left) and for the anomalies (right). The threshold C_{TH} is indicated by an arrow.

In our case, anomalies can be indirectly identified by the machine states classifier, in two scenarios: (1) the action predicted by the classifier is different from the one specified by the MES; (2) the predicted action is correct, but the decision has relatively low confidence. This is reasonable to assume, given that the classifier is asked to perform a prediction on data that are *anomalous*, thus significantly different from the ones it was trained on. While the first scenario is straightforward, the second requires a statistically sound measure of prediction confidence, generally not provided by deterministic ML models. For this purpose, we exploit MLP-B, a probabilistic implementation of MLP based on Bayesian theory. Differently from conventional training methods that attempt to find a single set of optimal values for the neural network weights, the Bayesian approach estimates the posterior distribution of the weights and makes a prediction by integrating over this distribution [32]. To produce a distribution of weights (and hence, of predictions), in our work, we use the Monte Carlo Dropout method, in the popular implementation by [32]: random dropouts (*i.e.*, random switching-off of neurons) are used to produce many different networks at inference time, that can be treated as Monte Carlo samples from the space of all available models. This allows inferring a distribution of predictions per each given input, providing the mathematical grounds to estimate a confidence level: the so-called Maximum Softmax Response (MSR) (*i.e.*, the maximum value of Softmax activations in the last MLP layer).

To assess the goodness of our approach, we extend the available dataset, already described in the previous sections, by artificially adding 56 anomalies created by manually hampering the robot motion at random instants during data acquisition. The so-obtained dataset is fed into the MLP-B, and the distribution of MSR confidence values obtained during machine states classification are shown in Figure 7, separately for the normal events (on the left) and the anomalies (on the right). From these plots, we can see how MLP-B confidence is very high with *normal* events (peak of distribution in the left plot is at MSR=1.0) and significantly drops in the presence of anomalies (peak of the right plot is at MSR=0.2). This demonstrates our initial hypothesis that the classifier confidence can be indirectly used to identify anomalies.

Starting from these considerations, it is reasonable to as-

sume that the overall distribution of the predictive confidence is bi-modal, where the highest mode is associated with high-confidence values, which are the majority, and the second to the low-confidence ones (*i.e.*, anomalies). Hence, to automatically identify an anomaly, we just need to establish a threshold C_{TH} on the confidence level. The easiest way to do so is to compute the value that splits the confidence values of the training dataset into two groups with maximum inter-group variance (see the arrow in Figure 7). By applying this approach to our dataset with artificial anomalies, with a 5-fold cross-validation strategy, we obtain that anomalous events are recognized with a mean accuracy of 79% (sensitivity 84%, specificity 75%). These results are remarkable for two reasons: i) *the anomaly detection is fairly accurate*, even without having received specific training on the anomalies to be detected, which would be difficult to implement in real industrial applications; ii) *the training only relies on data that is automatically collected, integrated and annotated during a normal production process*. This further demonstrates the usefulness of our proposed data fusion infrastructure.

V. Conclusions

This work presents an enhanced solution for effectively implementing a data fusion approach in the context of Industry 4.0. The proposed DIH merges data traces produced by different sensors with recipe data extracted from the MES, generating sound datasets with advanced production information. The proposed infrastructure is showcased on an anthropomorphic manipulator located in an industrial research laboratory and is demonstrated applicable to different industrial machinery. The relevance and robustness of the fused data is proven by the application of state-of-the-art data analytics algorithms. Future works will try to i) extend the infrastructure with more data analytics functionalities, and ii) map these algorithms on embedded platforms, enabling computation on the edge and real-time monitoring.

References

[1] H. R. Chi, C. K. Wu, N.-F. Huang *et al.*, "A survey of network automation for industrial internet-of-things toward industry 5.0," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 2065–2077, 2023.

[2] A. T. Rosário and J. C. Dias, "How Industry 4.0 and sensors can leverage product design: Opportunities and challenges," *Sensors*, vol. 23, no. 3, 2023.

[3] Z. Kang, C. Catal, and B. Tekinerdogan, "Machine learning applications in production lines: A systematic literature review," *Computers & Industrial Engineering*, vol. 149, p. 106773, 2020.

[4] M. Mestre, "Avoiding top pitfalls in annotation projects," https://towardsdatascience.com, 2021.

[5] S. Enderes, "The impact of annotation errors on neural networks," https://understand.ai, 2021.

[6] Sepasoft, "What is a recipe?" https://help.sepasoft.com/docs/pages/viewpage.action?pageId=7313378, 2023.

[7] D. Ursino, Y. Takama, and F. Castanedo, "A review of data fusion techniques," *Hindawi Scientific World Journal*, 2013.

[8] A. Tsanousa, E. Bektsis, C. Kyriakopoulos *et al.*, "A review of multisensor data fusion solutions in smart manufacturing: Systems and trends," *Sensors*, vol. 22, no. 5, 2022.

[9] A. Argyrou, C. Giannoulis, A. Sardelis *et al.*, "A data fusion system for controlling the execution status in human-robot collaborative cells," *Procedia CIRP*, vol. 76, pp. 193–198, 2018.

[10] S. Gaiardelli, S. Spellini, M. Panato *et al.*, "A software architecture to control service-oriented manufacturing systems," in *Proc. of IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 1–4.

[11] Y. Roh, G. Heo, and S. E. Whang, "A survey on data collection for machine learning: A big data - ai integration perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, 2021.

[12] Y.-H. Hung, "Improved ensemble-learning algorithm for predictive maintenance in the manufacturing process," *Applied Sciences*, vol. 11, no. 15, p. 6832, Jul. 2021.

[13] M.-Q. Tran, M.-K. Liu, and M. Elsisi, "Effective multi-sensor data fusion for chatter detection in milling process," *ISA Transactions*, vol. 125, pp. 514–527, Jun. 2022.

[14] A. Y. Jaen-Cuellar, M. Trejo-Hernández, R. A. Osornio-Rios *et al.*, "Gear wear detection based on statistic features and heuristic scheme by using data fusion of current and vibration signals," *Energies*, vol. 16, no. 2, p. 948, jan 2023.

[15] A. Diez-Olivan, J. D. Ser, D. Galar *et al.*, "Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0," *Information Fusion*, vol. 50, pp. 92–111, Oct. 2019.

[16] J. Wan, J. Yang, S. Wang *et al.*, "Cross-network fusion and scheduling for heterogeneous networks in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6059–6068, 2020.

[17] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe *et al.*, "Microservice based architecture: Towards high-availability for stateful applications with kubernetes," in *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, 2019, pp. 176–185.

[18] M. Urbina, T. Acosta, J. Lázaro *et al.*, "Smart sensor: SoC architecture for the industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6567–6577, 2019.

[19] M. V. García *et al.*, "From ISA 88/95 meta-models to an OPC UA-based development tool for CPPS under IEC 61499," in *Proc. of IEEE WFCS*, 2018, pp. 1–9.

[20] A. Hazra, M. Adhikari, T. Amgoth *et al.*, "A comprehensive survey on interoperability for IIoT: Taxonomy, standards, and future directions," *ACM Computing Surveys*, vol. 55, no. 1, 2021.

[21] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, 2011.

[22] P. Dobbelaere and K. S. Esmaili, "Kafka versus RabbitMQ: A Comparative Study of Two Industry Reference Publish/Subscribe Implementations: Industry Paper," in *Proc. of ACM DEBS*, 2017, p. 227–238.

[23] IEEE 1588-2019, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," IEEE, Standard, 2019.

[24] ISO17359, "Condition monitoring and diagnostics of machines," International Organization for Standardization, Geneva, CH, Standard, jan 2018.

[25] ISO13373-1, "Condition monitoring and diagnostics of machines — vibration condition monitoring," International Organization for Standardization, Geneva, CH, Standard, feb 2002.

[26] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[27] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of Machine Learning Research*, vol. 13, no. 2, 2012.

[28] I. Bozcan, C. Korndorfer, M. W. Madsen *et al.*, "Score-based anomaly detection for smart manufacturing systems," *IEEE/ASME Transactions on Mechatronics*, 2022.

[29] J. S. L. Senanayaka, H. Van Khang, and K. G. Robbersmyr, "Multiple classifiers and data fusion for robust diagnosis of gearbox mixed faults," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 8, pp. 4569–4579, 2019.

[30] T. Xie, X. Huang, and S.-K. Choi, "Intelligent mechanical fault diagnosis using multisensor fusion and convolution neural network," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 5, pp. 3213–3223, 2022.

[31] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 11, pp. 4037–4058, 2020.

[32] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1050–1059.