Computers in Industry 120 (2020) 103244

Contents lists available at ScienceDirect

Computers in Industry

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

Machine learning for predictive scheduling and resource allocation in large scale manufacturing systems

Cristina Morariu, Octavian Morariu*, Silviu Răileanu, Theodor Borangiu

Department of Automation and Applied Informatics, University Politehnica of Bucharest, Romania

ARTICLE INFO

Article history:
Received 18 December 2019
Received in revised form 5 March 2020
Accepted 14 April 2020

Keywords:
Large scale manufacturing system
Big data streaming
Cloud manufacturing
Machine learning
Long short-term memory neural network
Prediction
Classifier
Anomaly detection

ABSTRACT

The digitalization processes in manufacturing enterprises and the integration of increasingly smart shop floor devices and software control systems caused an explosion in the data points available in Manufacturing Execution Systems. The degree in which enterprises can capture value from big data processing and extract useful insights represents a differentiating factor in developing controls that optimize production and protect resources. Machine learning and Big Data technologies have gained increased traction being adopted in some critical areas of planning and control. Cloud manufacturing allows using these technologies in real time, lowering the cost of implementing and deployment. In this context, the paper offers a machine learning approach for reality awareness and optimization in cloud.

Specifically, the paper focuses on predictive production planning (operation scheduling, resource allocation) and predictive maintenance. The main contribution of this research consists in developing a hybrid control solution that uses Big Data techniques and machine learning algorithms to process in real time information streams in large scale manufacturing systems, focusing on energy consumptions that are aggregated at various layers. The control architecture is distributed at the edge of the shop floor for data collecting and format transformation, and then centralized at the cloud computing platform for data aggregation, machine learning and intelligent decisions. The information is aggregated in logical streams and consolidated based on relevant metadata; a neural network is trained and used to determine possible anomalies or variations relative to the normal patterns of energy consumption at each layer. This novel approach allows for accurate forecasting of energy consumption patterns during production by using Long Short-term Memory neural networks and deep learning in real time to re-assign resources (for batch cost optimization) and detect anomalies (for robustness) based on predicted energy data.

1. Introduction

The post-industrial era provided a stable environment that spanned several decades in which manufacturing companies were able to exist and evolve gradually, based on common competition criteria. One perspective is that the open competition defines success for these companies based on their ability and efficiency in delivering finished goods. The relatively stable economic climate and the small variance in the process of transforming raw materials in physical products fostered the innovation in directions that increased efficiency for specific areas of manufacturing, like advanced enterprise resource planning, supply chain optimization, ad-hoc automation solutions and many others. From a technological perspective, the last decade has brought a wide scale migration from legacy / custom software to cloud-based SaaS solutions, benefiting for even more integration options and therefore increased efficiency. Recent studies have shown that the current economic environment is changing and the term VUCA (volatile, uncertain, complex and ambiguous), used to describe it gains more attention (Bennett and Lemoine, 2014; Horney et al., 2010). In this context, manufacturing firms need to shift their focus from linearly improving efficiency towards real time learning from big data and contextual decision making. This approach reduces the uncertainty by allowing accurate predictions of relevant key performance indicators based on previous production data. At the same time, it reduces the complexity of human decision makers, by augmenting people with contextual insights at each point in the process. Data and the way it can be processed in real time become thus differencing success factors in these companies.

* Corresponding author.
E-mail addresses: cristina.morariu@cimr.pub.ro (C. Morariu), octavian.morariu@cimr.pub.ro (O. Morariu), silviu.raileanu@cimr.pub.ro (S. Răileanu), theodor.borangiu@cimr.pub.ro (T. Borangiu).

https://doi.org/10.1016/j.compind.2020.103244
0166-3615/

The digitalization processes of large manufacturing enterprises and the integration of increasingly smart shop floor devices and control software caused an explosion in the data points available at shop floor and Manufacturing Execution System (MES) layers. The degree in which enterprises can capture value from processing these data and extract useful insights from them represents a differentiating factor on short- and medium-term development of the processes that optimize production.

Machine learning (ML) and Big Data technologies have gained increased traction by being adopted initially for corner case scenarios and as more data and computation power became available, also in some critical areas of planning and control. Cloud manufacturing provides a robust platform for developing these solutions, lowering the cost of experimentation and implementation of various solutions.

In this context, this paper offers an ML-based approach for reality awareness and efficiency in cloud manufacturing and proposes two practical applications of machine learning algorithms for global optimization of manufacturing at batch level, robust behaviour at disturbances and safe utilisation of manufacturing resources. The paper focuses on batch planning (operations scheduling and resource allocation) based on the prediction of energy consumptions for the optimization of global energy cost, and safe resource usage (predictive resource maintenance and team reconfiguration) based on detecting anomalies and predicting unexpected faults from real time big data and ML models. The paper extends the authors' research in big shop floor data streaming (energy consumed by resources), described in (Morariu et al., 2019), for predictive, ML-based control.

There are three important dimensions when processing shop floor data:

- Aggregating at the right logical levels when data originates from multiple sources;
- Aligning the data streams in normalized time intervals;
- Extracting insights from real time data streams.

All these dimensions should be considered in the context of scale. In other words, the processing of these data streams must be scalable in a linear fashion, so that the overall processing time remains short. In practice, the insights that can be extracted from real time data are less accurate with the number of forward steps predicted. A short time spent in the data processing pipeline is critical for the system's accuracy.

The prediction functionality as presented in this paper is designed at operation time horizon for each individual shop floor resource. The ability to predict accurately, in real time, the instant power consumed by a resource in any given operation is the core functionality that is used as building block for predictive planning and maintenance, as well as real time fault detection. The solution presented in this paper works as a generic prediction engine, where the machine learning model tries to predict the next state of the system at every level. The actual state of the system is then compared to the predicted value. For normal behaviour of resources, the prediction should be close to the real system state; if that is not the case due to the degradation of resource performances, a real time corrective action is initiated that either weights down the resource allocation to jobs (updating the optimal production schedule) or triggers predictive resource maintenance operations.

The main contribution of this research consists in developing a hybrid control solution that uses Big Data techniques and machine learning algorithms to process in real time information streams in large scale manufacturing systems, focusing on energy consumptions that are aggregated at various layers. The control architecture is distributed at the edge of the shop floor for data collecting and format transformation, and then centralized at the cloud computing platform for data aggregation, machine learning and intelligent decisions. The information is aggregated in logical streams and consolidated based on relevant meta-data; a neural network is trained and used to detect deviations from the normal patterns of energy consumption at each layer and anomalies. This allows forecasting accurately energy consumption patterns during the production cycle using Long Short-term Memory neural networks (LSTM) and deep learning in real time, and optimizing resource allocation to jobs based on predictions.

The implementation presented in this paper combines the scalability of big data architectures with the power of real time machine learning, specifically recurrent neural networks, to build predictive capabilities for manufacturing structures. Specific prediction/reaction use cases are presented for batch planning (operations scheduling and resource allocation), maintenance and fault detection scenarios, along with future research directions.

Traditionally, machine learning is used in resource maintenance based on anomaly detection (Luo et al., 2018; Pereira and Silveira, 2018), but in this paper machine learning is used in manufacturing control, more specifically in real-time optimization of production plans (scheduling operations and allocating resources) at batch horizon based on forecasting energy consumption and execution times of resources.

The proposed ML-based control framework applies for job shop processing (parts may visit any machine more than once) in shop-floor networked and service-oriented cloud manufacturing (CMfg) models that:

- comprise a pool of reconfigurable and interchangeable shop floor resources (robots, machines) that can be shared to offer on-demand manufacturing services;
- enable pervasive, on-demand network access to a shared pool of configurable high-performance computing (HPC) resources (server, storage, ML software tools and applications) that can be rapidly provisioned and released as services to the higher MES, predictive System Scheduler and preventive maintenance managing layers.

This means that the proposed CMfg model for job shop processing uses Cloud Computing (CC) facilities for high performance purpose: optimization of production costs (energy consumption, makespan), safe resource utilisation and preventive maintenance through Machine learning.

The paper is structured as follows: section 2 presents recent work about Machine learning and Big Data technologies applied to manufacturing systems and real-life problems. Section 3 analyses Big Data streaming techniques for real-time processing of manufacturing data generated by shop floor resources and intelligent products. Section 4 presents some machine learning applications for manufacturing control: prediction, classification and clustering; a special recurrent neural network – the Long Short-term Memory is proposed for prediction tasks. An implementing solution is described in section 5 relative to: messaging and Big Data processing; using machine learning to create energy consumption models (case study: robot operations); using forecasted data for real time optimization of the production plan and anomaly detection for preventive resource maintenance. Section 6 shows experimental results, formulates conclusions and future work directions.

2. Related work

Stream processing for manufacturing systems is a well-studied topic. The authors of (Arasu et al., 2016) propose a Data Stream Management System that is equipped with a Continuous Query

Language (CQL). This approach is very useful because it eases the integration by using a SQL-like approach to data retrieval from live streams. However, it lacksP the distributed nature of publish-subscribe messaging systems, and thus will not scale beyond a certain point. Apache Kafka (Kreps et al., 2011) provides a good platform for building real- time streaming data pipelines that reliably get data between systems or applications and for the development of real time streaming applications that transform or react to data streams. It provides a distributed platform written in Java for publish-subscribe architectures at scale.

On the other hand, the emergence of Industry 4.0 paradigm has raised the interest in extracting insights from shop floor data for various monitoring and optimizations purposes. In (Wang et al., 2018), the authors explain the main differences between traditional machine learning and deep learning techniques applied to manufacturing shop floor data. There are several applications mentioned for manufacturing, specifically: continuously optimized production at batch horizon, quality inspection of products, diagnostic analysis, fault assessment and predictive operations for defect prognosis. All these examples are falling in the category of classification problems where a deep learning solution is proposed. In these cases, the classifier input can be the product being manufactured or a fault to be detected. In another study (Shao et al., 2017), the authors use deep belief networks in order to learn useful features from frequency distribution of vibration signals. These feature vectors are then used for fault detection in induction motors.

In (LeCun et al., 2015) the authors discuss the prediction properties of recurrent neural networks in general and some of their applications to real life problems (i.e. language translation, speech recognition, image recognition, automatic labelling, and so on). Some more advanced research related to long short-term memory (He and Wang, 2007; Sutskever et al., 2014; Mikolov et al., 2010) shows that these neural networks are particularly good at learning data patterns which have a longer temporal dependency. The authors of (Chandramitasari et al., 2018) use LSTM to predict energy consumption in a manufacturing setting. A study on LSTM in (Sundermeyer et al., 2012; Williams and Zweig, 2016) is focused on how a LSTM can be trained for different tasks and on how supervised learning (SL) performed by a subject matter expert can be coupled with reinforcement learning (RL) to train a LSTM for a specific problem domain. Combining SL with RL proves to be a very powerful technique especially in cases with a lack of labelled data. A novel approach to short-term load forecasting using a LSTM is described in (Liu et al., 2017).

Manufacturing execution systems are part of this category due to their changing and evolving nature. Shop floor robots are reconfigured, upgraded or replaced, while the products' specifications are changing at a rapid pace. In this context, fault detection has also been studied by some authors (Viswanadham and Johnson, 1988; Heshan and Surgenor, 2017); LSTM forecasting can be used in this scope as well (Chandramitasari et al., 2018). The problem of energy-aware production scheduling is studied in (Shin et al., 2019), while research on energy-saving batch scheduling based on clustering methods is reported in (Tong et al., 2016).

This paper proposes an architecture that uses Big Data concepts and map-reduce algorithms to process the information streams in large scale manufacturing systems, focusing on energy consumptions aggregated at various layers. Once the information is aggregated in logical streams and consolidated based on relevant meta-data, a neural network is trained and used to determine possible faults or variations from the normal patterns of energy consumption at each layer. This novel approach allows accurate forecasting of energy consumption patterns during the production cycle by using Long Short-term Memory neural networks and real time classifiers in two stages:

- *Batch planning* – on long term: real time optimization of operations scheduling / product and resource allocation / operation for the entire batch of products to be executed based on the prediction of energy consumed by resources for each type of operation; the minimization of the total energy consumed by all resources used to work out the batch is considered);
- *Resource health monitoring* – on short term: whenever an anomaly or a significant increase in the energy consumed by a resource is detected, the resource is isolated, and its maintenance program is launched.

In addition to existing research, the ML solution presented here is embedded in the dynamic, real-time reconfiguration of semi-heterarchical production control at global batch level for optimization, and in forecasting unexpected events: resource faults or breakdowns for reality awareness and robustness at disturbances. This solution is based on predicting processing times and energy consumption of resources.

3. Big Data streaming for real-time manufacturing data processing

Dealing with large amounts of real time data originating from various sources and having different formats requires a careful design of the messaging system architecture. Increased resource and product instrumenting and adoption of Edge computing paradigms leads to new types of data sources, that generate high quality information, but at the same time large volume. This information flow is only useful if it can be processed in real time or near real time, so decisions can be derived fast. Dividing the messaging platform in separate parameter domains (or topics) with established producer-consumer relations helps in logically dividing the load. This concludes how the distributed messaging platform is storing and pushing information.

However, topics must be seen at a logical level only, as overstructuring the topics at design time can reduce the flexibility of the platform on long term (Dean and Ghemawat, 2008). For example, it would make sense to use distinct topics for shop floor resource messages (including instant energy consumption data) and for intelligent product messages (comprising operation execution times). However, dedicating a topic to only one type of resource might cause issues as the shop floor capacity can change in time, and that will require changes in the map-reduce algorithms down the line. At the same time, the content of the messages sent can be different depending on the application, like instant energy consumption, motor temperature, axis vibration, and so on. Using a free format for payloads is essential but requires a complex meta-data definition for each attribute.

Traditionally the communication at shop floor level is established in a stack model, where the following data flows can be identified:

- *Upstream*: consisting in events generated by the lower level modules (e.g. resource breakdown, zero stock, quality assurance failures, product tracking).
- *Downstream*: dispatching of production orders, operations and resources selected to perform the operations, route calculation and optimization.
- *Flat*: this kind of data flow is used when the original schedule cannot be followed due to unexpected events, like resource breakdowns or rush orders.

The stack-based data flow is a tightly coupled model, as each layer is coupled with two other layers and thus reduces the flexibility of the system. The introduction of a messaging system, acting

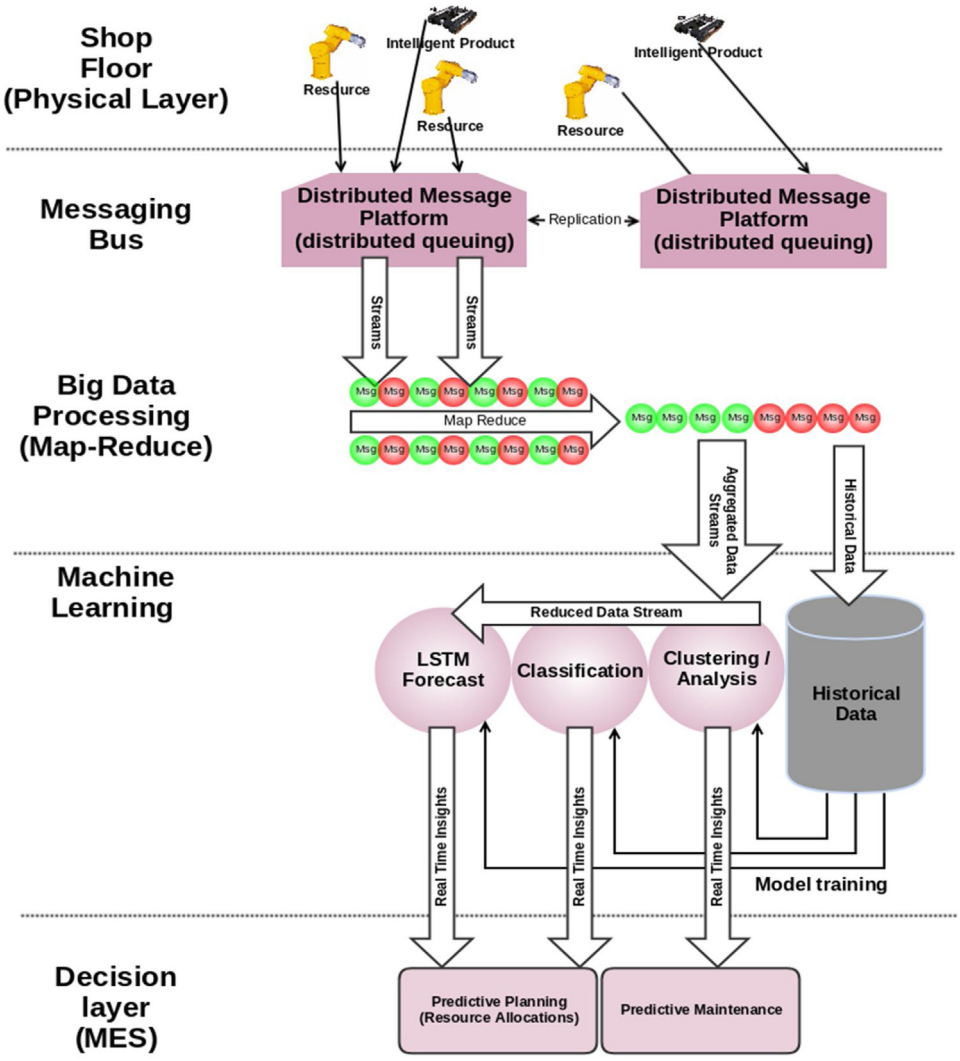

Fig. 1. Information flow in Big Data driven architecture with real time machine learning.

as a service bus for the MES, assures loose coupling between modules at shop floor level. In this context, the messaging system needs to support the following main characteristics:

1 **Event driven communication**: during the manufacturing process, there is a high number of events generated at shop floor level that need to be handled by specific components. For example, when a pallet arrives in a given position on the conveyor belt, a sensor detects the associated radio-frequency identification tag and generates an event. This event needs to be dispatched to the relevant resources in order to be processed by the scheduler or by the robot that performs an operation. The main role of the messaging system is to perform the event dispatch operation allowing shop floor components to exchange information in an event-driven mode.

2 **Workflows**: along with event dispatching, the messaging system can launch and execute predefined workflows associated with specific events. Workflows consist in sets of successive operations, either automated or manual (human interventions). Workflows are typically required for exceptional events that require complex logic to handle them, like unexpected resource breakdowns or rush orders.

3 **High number of messages**: the messaging system is a very effective architecture providing that all the other components are using it to exchange messages with each other, or in other words, the messaging system is not bypassed by using direct point to point communication. This allows the messaging system to have a global view on the system. However, the disadvantage is that in complex manufacturing systems, the number of messages passed can grow exponentially based on the number of modules involved and the number of products. From a messaging bus implementation perspective, it is important to assure that high message throughput is possible.

4 **Message transformation**: the shop floor level integrates a wide range of modules, from software schedulers to various hardware devices (robots, sensors, etc.). From a communication perspective, the protocols and the message formats used can be simple +5 V dc signals, proprietary line protocols or even high level TCP based protocols. The messaging system's role is to transform these messages to and from these proprietary protocols in a common standardized format. This is done by allowing the development of message convertors at any entry point and exit point of the bus.

5 **Message Validation**: the messaging system needs to perform message validation at entry points. Malfunctions or defects in shop floor devices can generate messages that are invalid because they are either incomplete or obsolete or duplicated. The robustness of the system depends on the validity of the messages. Filtering invalid messages at the edge of the processing system, before they enter in the big data map-reduce, becomes very important.

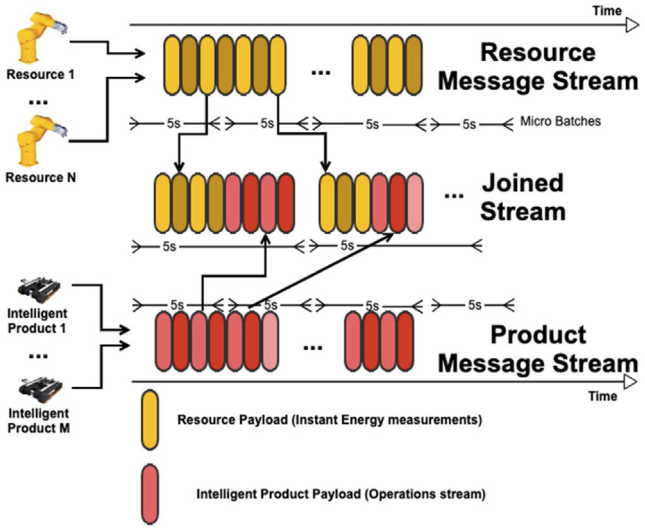

Fig. 2. Message flow and micro batch streaming.

The overall layout of the information flow for the proposed solution is shown in Fig. 1.

The information flow starts from the shop floor where different resources and intelligent products post information packages to the messaging system topics (Morariu et al., 2012). While these messages may contain any kind of information, ranging from self-monitoring to images for quality control, the topics used here are: energy consumption for resources, and energy consumption per operation for intelligent products. In practice the messages are encoded in JSON format and are posted via REST protocol (Lange, 2016). Considering two topics, one for resources and one for orders embedded on intelligent products, the message flow is illustrated in Fig. 2.

In the upper side, the resources are posting messages to a Resource topic, while the products are posting to the Product topic. The messages are ordered in time and this order must be guaranteed by the messaging system for each topic. These initial streams can be considered as raw streams of information.

The next step in the processing is represented by joining of the raw streams in application specific streams. In the example illustrated in Fig. 2, the join operation is done between the resource streams and intelligent product stream. It is important to notice that the joined stream contains at this point messages of two different types; red ones represent messages from resources while green ones represent messages from intelligent products. The *join / merge* big data operation is distributive in nature. This means that it can be executed in parallel on multiple worker nodes and thus will scale linearly with the number of messages and the number of worker nodes in the map-reduce cluster. Another important aspect is that the map-reduce operations are done in a series of micro-batches. In Fig. 2 the micro-batch (which can be considered as the time horizon of the *join* operation) is illustrated as a 5 s interval. In practice, the micro batch interval will be given by the nature and expected frequency of the messages in the stream. For example, if the correlation is between a slow changing flow like weather information and high pace change rate like instant traffic measurement, the aggregation can be done on micro-batches of different time length.

Once the information required for a given application is assembled in a joined stream, the next operation is a *reduce* type operation, as illustrated in Fig. 3.

The reduce operation is also working at micro-batch level and the goal is to process a set of messages in a logical fashion and create a reduced information stream. The operation implemented

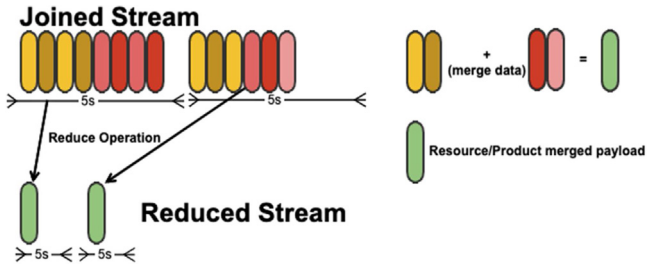

Fig. 3. Reduce operation on micro batches in a stream.

as a reduce operation must also be distributive in nature, so that it can be executed in parallel by the worker nodes. Some examples of operations can be: a *merge* operation if the messages are JSON objects, a *sum* or an *average*. While Fig. 3 presents only a theoretical abstraction, in real case applications a successive set of map-reduce operations are needed to obtain the required reduced streams that have the aggregated information available.

An important aspect of this approach is that the reduced stream now contains information from both resources and intelligent products that is aligned in time. In other words, as the messages are initiated by the shop floor actors and pushed to the messaging queue, they have different original timestamps. The merged message on the reduced stream has all the information, aligned in time at the micro-batch interval. There are two problems that can arise with this approach: firstly, the situation in which multiple messages from the same resource or intelligent product are in the same micro-batch and secondly, the scenario where a micro-batch does not have a message from an expected source. Both problems are usually resolved with an additional mapping operation that either merges the messages or picks only the last one depending on the specific application.

For most applications, the reduced stream will suffer more refining operations and enrichments with other data sources that can be static and proprietary to the manufacturing system like customer order management, or even external with the goal of forming aggregated data streams. These aggregated data streams contain the entire information set required for the next step of the data processing pipeline.

4. Machine learning applications in manufacturing

The aggregated data streams can be considered as an endless stream of feature vectors that can be used to extract insights in real time (ref. lower part of Fig. 1). This paper proposes basically three types of machine learning applications that can be used for intelligent manufacturing control:

- The first type tries to resolve the *prediction problem*, and the insights obtained can be used directly in the business layers for decision making or in fault detection systems. More specifically, the prediction problem is interpreted as the possibility to use a deep learning neural network to learn patterns and variations in numerical measurements, i.e. energy consumption, and then use the neural network to make forecasts for that measurement. This is especially useful when the data has temporal patterns that can be learned.
- The second type deals with *classification problems*, in which the system tries to determine a class for every feature vector received.
- Finally, *clustering* is represented by a set of algorithms that try to find similarities in non-labelled, multidimensional data and tag each feature vector accordingly. Examples of applications of the latter are quality assurance, image recognition for part picking, etc.

From an architectural perspective, the utilisation of machine learning approaches discussed in this paper requires historical data storage that serves to train the machine learning models. Depending on the algorithm, being a supervised or non-supervised approach, the training and re-training would be done as a continuous process.

While all three machine learning approaches are relevant for manufacturing systems in the context of big data processing, this paper focuses on the first problem - that of prediction using energy consumption samples. The prediction problem is a time-based problem, where the numerical measurement changes value at each time interval. This makes it a logical candidate for big data stream processing, where the micro-batch interval is the main driver.

The forecasting of energy consumption based on ML techniques will be applied to manufacturing systems in two ways:

- Firstly: the information flow of the energy-related messages of *intelligent products* (IP) at operation granularity; intelligence is embedded on the processors placed on the pallet carriers that transfer the products on the conveyor belt between the resources assigned to perform the scheduled material processing and assembly operations; the prediction addresses the cost of products and can be used to globally optimize batch production.
- Secondly: the energy consumption data continuously measured from *shop floor resources* performing the assigned operations on products; the prediction addresses the resources' health and Quality of Services (QoS) performed and can be used for safety assurance.

4.1. Learning patterns from energy consumption data streams

The pilot implementation uses two message types, based on their origin: resource message and IP message. The resource message contains information about the instant power and current usages and is generated periodically by each resource. The design does not require any synchronization between resources in terms of messages and is also tolerant to missing values. The alignment in time of all these messages is achieved by the map-reduce operations in the message streams. The product-based messages contain information about the current operation that is to be executed on the product.

The messages are encoded in Java Script Object Notation format due to the wide adoption of this encoding in modern distributed systems and the large availability of APIs for JSON processing in programming languages supported by Spark: Python, Java and many others. The actual message types have the following JSON formats:

Resource Message JSON format

```
{
 resourceID: robot_1, //unique ID for the robot
 instantPower: 200, // power in W
 instantCurrent: 0.91, //current in A
 targetID: 2103, //ID of the target pallet
 timestamp: 1521044131128 //timestamp in ms
}
```

Intelligent Product Message JSON format

```
{
productID: 2103, //ID of the pallet
currentOperation: Op_pick_place_2, //op id
productBatchID: 1, //product batch ID
timestamp: 1521044133128 //timestamp in ms
}
```

The time interval at which the resource is pushing energy consumption messages through the system containing the instant power values is 250 ms, while the intelligent product is pushing messages every 5 s sampling interval. The goal for this is therefore to map-reduce these messages at operation level, in order to have a time line for energy consumption.

The following map-reduce operations were used to achieve this function, which is explained by the comments inserted in the code:

```
ResourceStream = createStreamFromTopic(resource_topic)
ProductStream = createStreamFromTopic(product_topic)
//merge the two streams in a unique stream
MergedStream = ResourceStream.merge(ProductStream)
//map on product id extracted from both messages
//depending on the msg type, the targetID or the
//productID will be used
MappedStream = MergedStream.map(productID, msg)
//reduce based on productID by appending
ReducedStream = MappedStream.reduceByKey(append(msg1,msg2))
//at this point for each product ID we have an array
//of resource messages with energy consumptions
```

The resulting stream will contain messages with the following structure:

Mapped reduced stream JSON message structure

```
{
productID: 2103, //ID of the pallet
operations:
[{ op:  Op_pick_place_2,
  resourceID: robot_1,
  energy_consumption:[
    { instantPower: 200, // power in W
      instantCurrent: 0.91, //current in A
      timestamp: 152104413112 //timestamp in ms
    },
    … //multiple entries for operation (e.g.: start operation, stop operation, etc.)
 ]
 },
 …  //multiple operations per product
]
}
```

The resulting operation data stream contains useful information aggregated at the product level, with all operations and costs associated and aligned in chronological time order. Therefore, it can be used for on-line (automatic) training of the ML models. Specifically, in the scope of implementation, the map-reduce stream is used as an input for operation/resource.

The software implementation of the prototype uses a 4 node Apache Kafka cluster for messaging system, with a Gunicorn REST-based web server acting as an API endpoint for various shop floor devices; it uses Python Web Server Gateway Interface (WSGI) HTTP server. This implementation is linearly scalable and can accommodate many shop floor devices and messages (Anon, 2020). At the same time, the Python-based REST API provides a flexible point of integration for message filters and validators. The map-reduce code is executed on Apache Spark streaming engine running on an 8- worker cluster configuration.

The processed streams are published back to Kafka on a separate topic, from where the downstream ML models are consuming the data. Fig. 4 illustrates the high-level architecture of the messaging system and big data design of the prototype. In practice, the services used for the pilot implementation are generally available in modern cloud implementations as micro-services.

Processing large amount of manufacturing data requires machine learning techniques in order to derive useful decisions in real time for process control. A classical, "threshold based" approach to data processing is not feasible nor scalable for large number of resources and product types. The self-adaptability and self-learning of the system are key factors for successful implementations and economic feasibility. Unsupervised ML techniques are therefore more appropriate for large scale manufacturing systems, in this context. Another argument in this direction is that some covariance between signals cannot be predicted easily, especially in environments that are often reconfigured. These covariances, that can have a high impact on the system's efficiency, can be observed and reported using machine learning techniques.

From a technical perspective, the prediction can be done using recurrent neural networks (RNNs). These networks are different

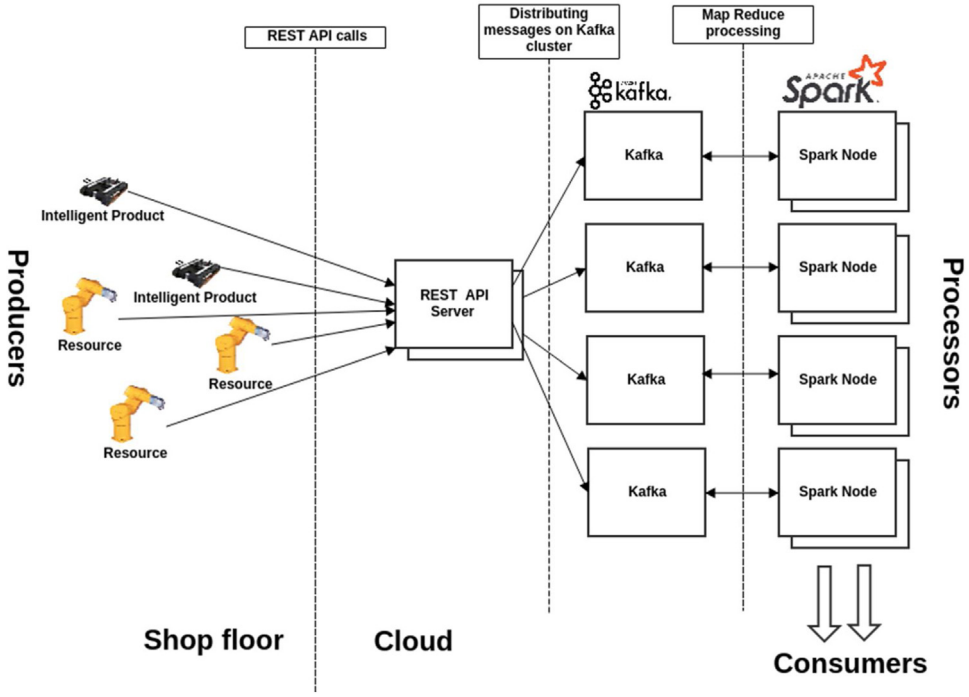

Fig. 4. The high-level architecture of the messaging system with big data design.

from normal neural networks in the sense that they hold an own time-based representation and use it for determining the current outcome. Because of this architecture they can be used to predict future values of learned patterns.

A sub-class of RNN, called long short-term memory (LSTM) is especially good at detecting and modelling patterns that occur in a time sequence. A LSTM has some additional set of gates for each instance, determining what should be learned and what should be forgotten at each step. The high-level view of a LSTM network architecture used for energy pattern prediction is illustrated in Fig. 5. The layout consists in t chained cells, where each cell is responsible for one step in the time series pattern. The inputs consist on instant energy measurements (X is the input feature vector) at each step, while the intermediate outputs (Y is the output feature vector) represent the predicted energy consumption at the next step in the sequence. The output and the state of each cell is feed to the next cell for all t steps in the time series sequence. One additional feature of the proposed LSTM architecture is the comparison of intermediate outputs with the next measured input. If the difference between the intermediate output and the next input is significant, an anomaly is triggered.

The prediction capability is the core functionality used to drive smart planning and maintenance decisions in manufacturing.

The building block of the predictive functionality is the LSTM model in univariate mode. In other words, the LSTM model is used to learn the pattern of a single parameter and predict in time the value, one or more steps ahead. Concretely, in the pilot implementation two types of machine learning techniques are being used:

- *Resource-based predictors*: bound to each shop floor resource and distinct for each operation the resource is performing. These are implemented using sklearn (Pedregosa et al., 2011) LSTM implementation and are running centralized in cluster of cloud VMs;
- *Product-based classifiers*: bound to each (distinct) product type, the classifiers rate the overall efficiency of each product completed using a multivariate feature vector against a pre-trained model.

Resource based predictors are identified by the resource they belong to and the operation type. For example, for a specific shop floor resource R1, capable of executing four distinct operations O1, O2, O3 and O4 there would be four individual and independent LSTM neural networks allocated. Each neural network learns a time- based pattern of energy consumption for the given operation in an unsupervised mode by processing the relevant stream resulting from the map-reduce operation.

Traditionally, the algorithm that produces the scheduling of operation execution and the resource allocation for these operations uses a generic, predefined estimation of energy consumption and job duration (and other parameters that are relevant for a certain implementation). This approach might be enough for small scale systems / batches, where an accurate estimation can be determined.

For large scale systems, the scheduling algorithm can be implemented in two stages. In the first stage the scheduling and allocation are computed based on historical data and initial estimations; in a second stage a LSTM is trained on each resource/operation pair, during live execution of scheduled operations. The LSTM models are then used as input for real-time rescheduling and/or reallocation, whenever the initial estimation starts to differ significantly from the actual predictions learned on each resource. This approach is detailed in the following section.

4.2. Optimization of production with forecasted energy data

The LTSM module was used for the prediction of instant power consumption of resources to provide a more accurate input for the cloud-based System Scheduler - an optimization module that performs mixed product and operation scheduling plus resource allocation: instead of the last recorded energy consumption values, forecasted values at batch execution horizon will be used as input for optimization (Fig. 6).

The parameters that are used as input data for the optimization engine are *operation execution time* and *operation energy consumption* for each resource (machine, robot); these parameters can be extended with other measurable data of interest. The ML tech-

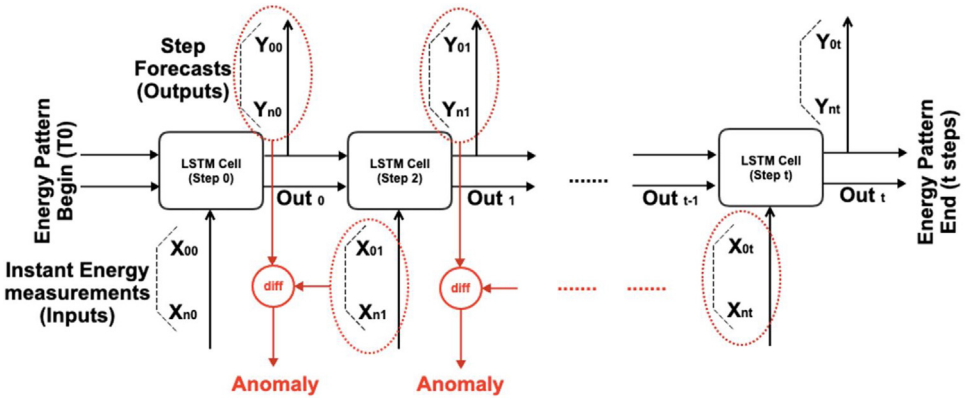

Fig. 5. LSTM time sequence energy prediction with intermediate step comparison.

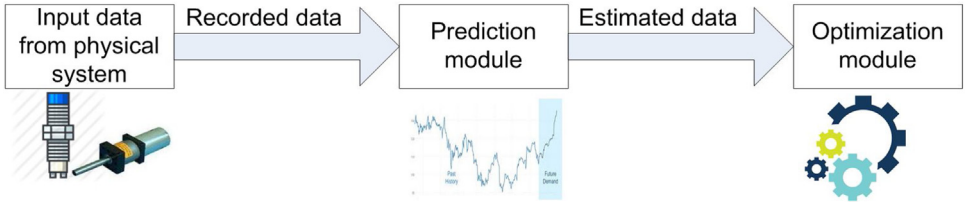

Fig. 6. Proposed input for the optimization engine – predictions of resource energy consumptions.

niques presented above forecast these parameters from measured data (timers and digital power meters) the necessary number of steps ahead until batch completion; the predicted values are then used as input data for the optimization module. In this architecture, the ML prediction module and the optimization module work in parallel being interconnected by a multi-dimensional buffer which is:

- updated by the prediction module each time the execution of an operation is recorded, and
- read by the optimization module each time a new schedule needs to be computed.

The multi-dimensional buffer represents the forecast of the parameters over a set of time intervals, operations and resources, see Fig. 7. This 3-dimensional buffer has the size $n_{max} \times m \times k$, $n_{max} = max\{n_1, \ldots, n_m\}$, where m is the number of types of operations on products each operation being performed n_i times and k is the number of resources, and is used to transfer the input data (the vectors of power consumption forecasts) from the prediction module to the optimization module. It results that n_{max} is the maximal dimension of the forecast horizon.

The prediction module works as follows for a set of m operations on a product, each one to be executed n_i times, $1 \leq i \leq m$ on a subset of k_{n_i} resources from the k-resource team (n_i [steps] is the forecast horizon for power consumption predictions):

Step 1: Initialization of the prediction module's training data set (the parameter "power consumption" and "execution time") for the resource team selected to execute the batch of products, from:

- historical records available for resources having performed (some of) the O_j, $1 \leq j \leq m$ operations of interest, in previous production sessions;
- initial power estimations for new machines and / or operations not done before.

Step 2: Set the forecast horizon to n_i, $1 \leq i \leq m$ (the farthest ones) for all operations i on all k_{n_i} resources in the selected team, execute the prediction algorithm with initialized training data set and make available the obtained parameter prediction values for the first run of the optimization module.

Step 3: Await the signals 'Start operation execution (SOC)' from the resources

Step 4: Loop:

- Upon receiving SOC signals from resources, read every 250 msec the instant power consumption values from the IoT gateway devices which are connected to these resources and use the data to learn the temporal pattern of the parameters.
- Run the anomaly detection algorithm and check the offset between the instant prediction and the value read from the IoT gateway
- If the instant prediction is significantly off compared to the real value read from the sensors, take the actions: signal 'resource fault', send report (machine ID, operation, time, parameter evolution) and add a description in the parameter's training data set for that resource) (Valckenaers et al., 2011).

Step 5: Extend the training data set (energy consumption and execution time) for an operation previously assigned to a resource that acknowledges termination of the operation by issuing the signal 'End operation execution (EOC)' with data acquired from sensors and processed on line by the IoT device of the resource.

Step 6: Update the forecast horizon: $n_i \longleftarrow n_i - 1$, execute the prediction algorithm to obtain the new inputs (predicted parameter values) for the optimization task.

Step 7: Goto Step 3.

The optimization module, which assigns resources to operations by minimizing the total power consumption for completing the entire batch works as follows:

Step 1: Initialize the optimization module's data for the resource team (resource ID, capabilities for operation execution) and run the off-line optimization module using history-based prediction values to obtain an initial resource allocation for the farthest forecast horizon n_i (no operation was yet performed) for all m operations on all k resources.

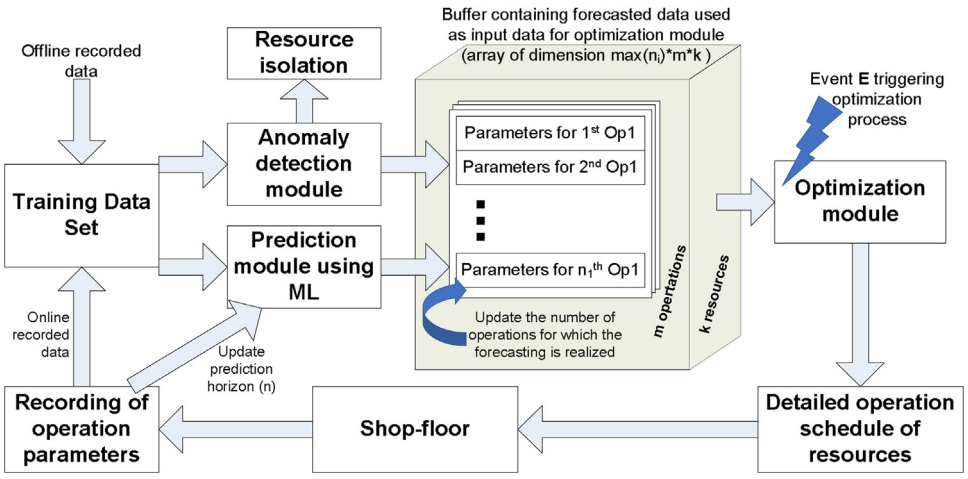

Fig. 7. Architecture interconnecting the ML-based prediction module with the optimization module.

Step 2: Wait the event E (shown in Fig. 7): termination of a product's execution cycle (TPC)

Step3: Update the input data of the optimization module with values forecasted by the prediction module and execute the optimization module.

- If the new global cost value obtained for the predicted power consumption is significantly smaller than the one computed in the previous Step 3, then re-assign resources and send the new schedule to shop-floor execution.

Note 1: This decision is taken only when the resource for which the power consumption of the most recent finished operation was computed and reported to the prediction module: i) shows a small change in the last prediction steps, and ii) the change is in the same direction.

Note 2: The optimization criterion might be a weighted combination of 'batch execution time' with 'energy consumption for batch completion'.

Note 3: Any time (at sampling periods of 250 msec) the high priority signal 'resource fault' and its associated report are received from the prediction module, the resource will be removed from the list of k team resources and from the optimization program.

The scheme in Fig. 7 also shows the information flow between the higher cloud layer of the MES hosting the prediction, anomaly detection, classifying and clustering engines based on machine learning and the system scheduler using optimization techniques, and the lower level, distributed MES with IoT gateways and sensors.

The collaborative process between the prediction module and the optimization module starts with the computation of the bill of materials (BOM) that specifies the set of operations needed to execute the ordered batch of products. The BOM includes the description of the set of m distinct operations O_i, $1 \leq i \leq m$, and their precedencies; at batch level, O_i is executed n_i times, hence the prediction module starts generating power consumption forecasts at the farthest forecast horizon of dimension n_{max}.

Although the total number n_i, $1, \forall 1 \leq i \leq m$ of operations O_i to be performed for all products in the batch is partitioned in k_{n_i} numbers $n_{i,j}; 1 \leq i \leq m, 1 \leq j \leq k$ by the optimization task which progressively (re)-assigns resources R_j to operations O_i after each product is completed, the forecast horizon of predicted energy consumption values (the input data to the optimization algorithm) should have the same maximal dimension of $n_i - n_{compl_i}$ (n_{compl_i} being the current number of finished products just before the current execution of the optimization module), for all the k_{n_i} resources candidates to the execution of operation O_i.

After each scheduled operation O_i is executed and its parameters are recorded, its associated training data set is updated using the online recorded data computed by the IoT gateway from smart meter data collected every 250 msec, and the forecasting horizon decreases by 1. This data represents the input for the prediction module which re-computes online the new parameter estimations and offers them as input data to the optimization algorithm. If a significant offset is detected in the new value of the optimization criterion relative to the one computed at the previous step, rescheduling is triggered each time a finished product exits the shop floor and a new one starts to be worked out.

The operating of the prediction and optimization modules is organized in two nested cycles:

i) Extending by 1 the training data set of the parameter of interest (e.g., resource power consumption) with the value computed by the IoT gateway device of the resource R_j that has just finished executing an operation O_i. Then, using the training data set increased in time by one value, the power consumption of resource R_j is computed according to the ML model for the forecast horizon decreased by 1 (see Fig. 8); this first prediction-related cycle is nested in a second cycle ii):
ii) Executing the optimization program with input data represented by the parameter forecasts (e.g., power consumption predictions) computed in cycle i) by the prediction module, each time a product is finished. Depending on the result of the global batch criterion computation and on the time evolution of the changes in the computed forecasts, resource re-allocation may be initiated.

So-called 'hard changes' may be initiated by detecting anomalies in the most recent parameter values (e.g. power consumed by resources) relative to learned patterns at the sampling period of 250 msec; they invalidate the existing resource operation scheduling since the affected resource is isolated and removed from the team. In this case it becomes mandatory to reschedule all operations on the remaining valid resources.

5. Experiments

5.1. Optimal resource allocation based on forecasted energy data

The testing scenario consists of *a pick and place* operation (the basic operation performed by industrial robots and used in assembly) executed using the same speed (70 %) and acceleration (100

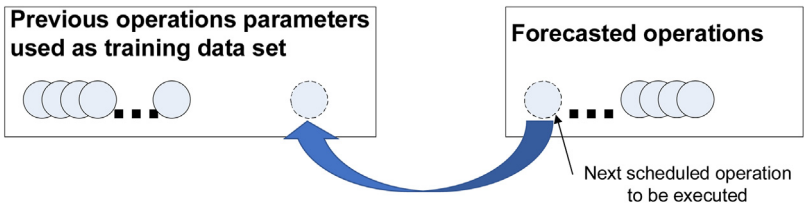

Fig. 8. The updating process of the training data set with parameter values measured from operations scheduled and executed, and the prediction process for the rest of operations to be executed.

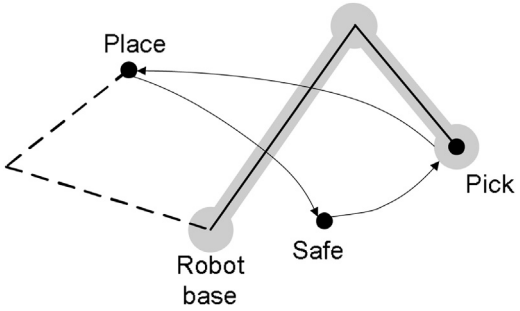

Fig. 9. Motion sequence for the pick and place robot operation.

%) limitations between two fixed points. The sequence of pick and place robot motions is shown in Fig. 9.

The distances between the points are as follows: d(safe, pick) =590 mm, d(safe, place) =302 mm, and d(place, pick) =590 mm. The operation's basic power profile is depicted in Fig. 10 for a set of 10 repetitions, from where it can be observed that the energy consumption rises three times corresponding to the approach pick, approach place and move safe executed motion commands. The spikes in the energy and instant current indicate an acceleration/deceleration phase of the robotic arm. The ability to learn such patterns, combined with the real time big data aggregation, allows accurate forecasting and immediate detection of performance degradation. The LSTM prediction is presented in Fig. 10 along with the actual instant power for comparison. The computed squared summed prediction error for this time interval is 66.466. As this is an unsupervised learning model where there is no labelled data as such, the error is considered by comparing with an operation measured in ideal conditions after several iterations through the LSTM. In this case, the error was measured after 100 iterations.

The algorithm for optimal resource allocation from predicted energy consumption values has been tested on a set of three industrial SCARA robots (R1, R2, R3) that can execute the same operation on a product, but with different costs (processing times and energy consumptions). The tests have been done on a batch of 1000 products with global objective function of minimizing energy consumption at batch level in due delivery time of 8 h (modelled as a constraint). The results are presented in Table 1 for three cases: a) optimization using static data, i.e. the last recorded values of energy consumption (6.39/6.4/6.41 KWsec) of R1/R2/R3; b) computation of the real energy consumption for the allocation generated in a) in the case when an increase of the energy consumed by R1 is registered; c) optimization using forecasted energy data for R1, R2, R3.

Table 2 presents the evolution of the total energy consumed and the resource allocation for a linear increase (4.8 %) of the medium energy consumption of resource 1. If this increase is not considered, the total energy consumed for the 1000-product batch increases with 1.75 %; if forecasted data is used then the increase of the total energy is reduced to 1.42 %. Resource allocation and operation scheduling were computed with the IBM ILOG OPL optimization engine in Constrained Programming approach (IBM, 2019).

5.2. Prediction-based classifiers for the cost of resource activities

While the described research focuses on resources' power consumption, alternative LSTM predictors can be used for other time-series signals like vibrations amplitude and speed, heat or supply voltage measurements, etc. The predictions and their related errors can be further combined in a feature vector where a second stage classifier can be used to evaluate the resource state at a different level: across operations, integrated in time, etc.

At resource level, predictive maintenance of resources may be triggered in two situations: 1) the instant prediction is significantly off compared to the real value read from the sensors; this indicates a forthcoming breakdown requiring urgent stop and a new resource allocation 2) the resource shows a small prediction error at each step, the error being in the same direction; this would indicate a moderate but consistent performance degradation (e.g. the accelerated wear of a robot joint causing an increase in instant power). In both cases, the remaining valid resources must be re-allocated to operations on products to be further manufactured; as a general rule, the predicted increase of the cost (energy consumed, execution time) of an operation performed by a resource causes weighting down the resource's allocation to that type of operation.

Product-based classifiers are linked with aggregated information originating from both product and resource logs (Intelligent Product Message JSON and Resource Message JSON) and make possible two types of classifications: a classification for each individual operation and a classification score across all operations required for manufacturing of a given product instance from the time it enters the production line to completion. The feature vector considered in the pilot implementation for each product type is presented in Table 2.

The classifications can be done in two modes. A simple way to classify both at operation level and at product level is by using measured signals. This is a post-execution classification that can score an operation and a product on the execution efficiency. A second mode is by using LSTM predicted values. This approach is useful when the MES system needs to forecast / evaluate the various scheduling solutions. For this experiment, the Keras LSTM implementation was used in state-full mode, so that the last state for each input will be used as initial state for the new input (Brownlee, 2016).

The classifier implementation used in the development of both operation level and product level classifications is One Class Support Vector Machine (SVM) (Pedregosa et al., 2011) with radial basis function non-linear kernel. This approach allows training the models on normal data and then using the trained model for abnormal operations and products, either based on LSTM predictions or on real time measurements. The decision border is determined automatically based on initial observations on normal operations. Therefore the algorithm is used to determine if a new operation is

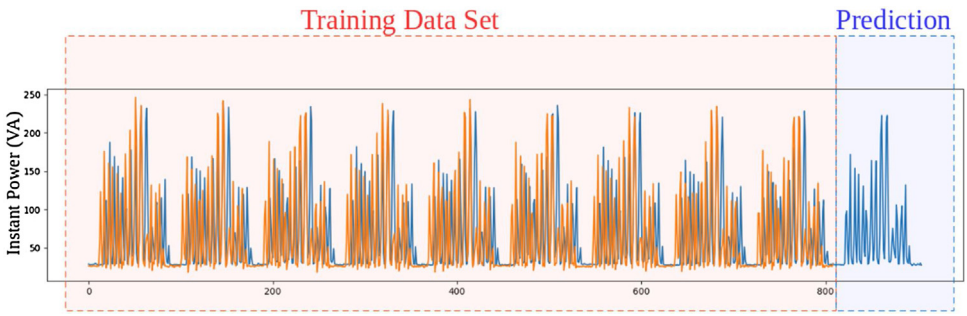

Fig. 10. LSTM training / prediction of instant power consumption.

Table 1
Comparison of the effects of resource allocation with static (historical) and forecasted energy data.

Production data and Key performance indicators	**Computed** (resource scheduling using static data) R1/R2/R3	**Registered** (allocation after energy consumption increase of R1) R1/R2/R3	**Forecasted** (resource scheduling using forecasted data) R1/R2/R3
Processing time (sec)	78/80/83	78/80/83	78/80/83
Number of products and resource allocation	369/360/271	369/360/271	294/360/346
Energy consumption/ product (KWsec)	**6.39**/6.4/6.41	**6.695**/6.4/6.41	**6.695**/6.4/6.41
Utilization time (sec)	28,782/28,800/22,493	28,782/28,800/22,493	22,932/28,800/28,718
Total energy consumed (KWh)	6399	6512	6490

Table 2
Feature vector considered for prototype classifier.

Signal Measured	Characteristic	Signal Source	Details
Total energy consumption	Variable due to machine usage	Measurement or LSTM prediction	The signal can either be read in real time or a forward LSTM prediction can be used instead
Speed limitation of each operation	Fixed through program	Scheduler	Speed is set by the predefined scheduling method for each operation
Time to complete	Variable due to machine usage	Measured	Measured in real time and available in the map-reduced stream
Travel distance	Fixed due to technological constraints	Scheduler	Defined by the scheduler when operations are assigned for resources. The travel distance is defined as in Fig.9 and will be always the same.

part of the same distribution of normal operations, or is an outliner, representing an abnormal observation.

Depending mostly on the dimensionality of the feature vector considered, other machine learning algorithms can be used; isolation forest and local outliner factor being appropriate for high and medium dimensions respectively. Another important decision factor for the classification algorithm selection is how clean the training data set is. Considering these criteria, the OneClassSVM SVM was chosen for this prototype, while the others are good candidates as well.

Fig. 11 depicts a set of 100 pick and place robot operations, measured at two different motion speed settings (70 % and 100 %). The illustration uses the Principal Component Analysis technique to reduce the feature space to 2D (Pedregosa et al., 2011). There were four abnormal operations included in the dataset, characterized by either higher energy consumption or longer execution time. The training data for the model is represented by the green dots, while the yellow dots represent live executions of the operations.

As can be observed in the graph representation, the SVM has identified two distinct clusters representing intuitively the two speed settings for the operations executed, as considered in the data set. Abnormal executions are pointed in red. Another useful feature of the classification approach is that a score can be computed from the location of the operation in relation to the Decision Border (represented by the red line). Therefore, a threshold can be applied and used as decision point for on-line re-scheduling of operations.

To test the product level classification, a set of 20 products were considered for training data, each one comprising six distinct operations as represented by green dots in Fig. 12. A further set of 10 products were evaluated at runtime against the trained model (yellow dots). Two abnormal products were identified in the data set, where the energy consumption had increased value.

The results show in this case a single cluster as the products were done with mixed speed operations. This is consistent with the fact that there was no specific variation on the product type or configuration. Like in operations classification, a scoring mechanism can be used to evaluate the efficiency of the assembly process aggregated at product level, or even at product batch level.

6. Conclusions

The results obtained confirm the benefit of predicting resource performances (timeliness, energy consumption, precision, a.o.) and using these values for production planning (operation scheduling and resource allocation). Detecting anomalies and predicting resource faults would also prevent production stops and confer reality awareness to resource maintenance.

Large scale manufacturing systems with multiple interchangeable resources can benefit from new big data architectures by allowing dynamic reconfiguring of operations scheduling and resource allocation based on real time aggregation and processing of large amounts of shop floor data. The combination between a publish-subscribe messaging system that aligns the data in time

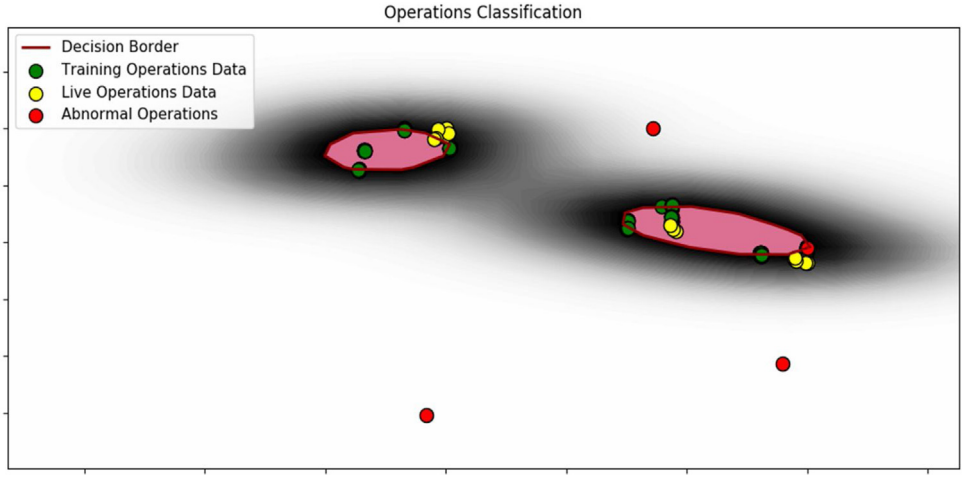

Fig. 11. Classification visualization (PCA and OneClassSVM) of manufacturing operations.

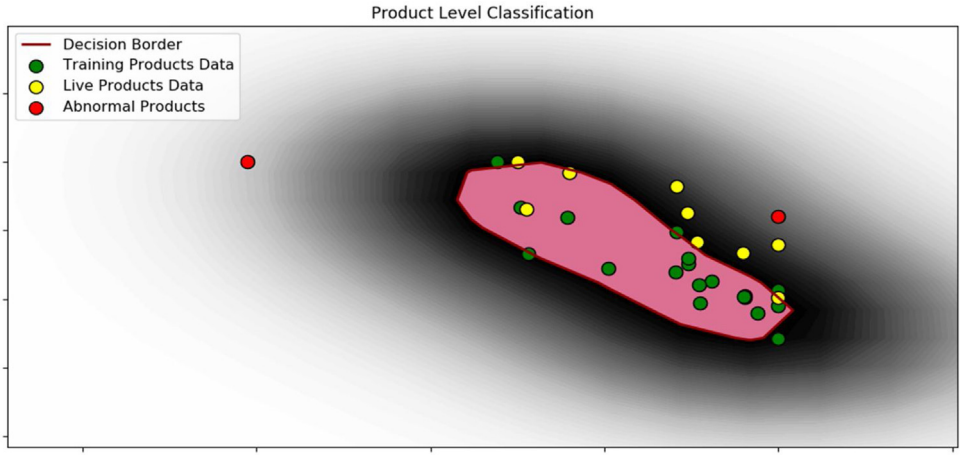

Fig. 12. Classification visualization (PCA and OneClassSVM) of data aggregated at product level.

and a map-reduce layer that can aggregate it along meta-data targets is a powerful approach to produce useful feature vectors that can describe the manufacturing system. Coupling this with machine learning techniques, insights can be extracted from information in real time.

There are two types of problems that machine learning can resolve in this context: the prediction of various performance indicators available as time series data, and the classification of aggregated data at operation level, product level or even batch level. These insights can be used for triggering real time decisions that can alter the execution of the manufacturing operations and the allocation of resources to reach a global goal such as: makespan, energy consumption at batch level, etc.

The approach presented in this paper can be implemented in shop floors with large number of resources due to scalability of its big data processing architecture and associated machine learning models that use unsupervised learning in the cloud. This second aspect is relevant, as the number of models depends on the shop floor resources and the number of operations supported.

Another very important benefit derived from applying the present approach is the increase of reality awareness in large-scale manufacturing systems, which is obtained by: a) mirroring the shop floor reality through the prediction of behaviours and properties of highly abstracted entities – operations, products and resources – based on realistic machine learning models, and b) predicting unexpected events through classifying, clustering and anomaly detection techniques.

LSTM pattern learning proves to be an important improvement that shifts the focus from configuration driven architectures towards data driven auto-configuration and auto-adjustments in real time. The prototype architecture introduced in this paper uses single variate techniques to model energy consumption patterns; however, multi variate models can be derived, where co-variance of performance indicators can be mixed. The main advantage of LSTMs is the ability for unsupervised learning form time series data, which directly applies to the manufacturing systems requirements.

The prediction-based solution provides better results than the traditional production planning based either on static, historical data or on measurements from resources, because it offers a global view at batch horizon. The solution, designed for large scale manufacturing systems, uses a cloud infrastructure that offers high processing power (needed for big shop floor data streaming, machine learning and optimization in real time), scalability and fault tolerance. The execution time for predictive production re-scheduling ranges from 0.38 s for a 100 product-batch to 4.1 s for a 1000 product-batch, with a 2-core cloud machine running at 2.6 GHz with 4GB of RAM.

Future work is focused on further developing the architecture presented in this paper to allow the aggregation and prediction of multiple signals in parallel, with several map-reduce parallel jobs. This extension from univariate problem to a multivariate solution

has the potential to reveal some interesting correlations between the system's performance forecast on multiple dimensions.

Another direction for future research is the development of Digital Twin models of shop floor resources and processes (manufacturing operations) and embedding them in the machine learning models for control and maintenance.

CRediT authorship contribution statement

Cristina Morariu: Formal analysis, Supervision, Project administration. **Octavian Morariu:** Writing - review & editing, Software. **Silviu Răileanu:** Resources, Data curation. **Theodor Borangiu:** Conceptualization, Methodology.

Declaration of Competing Interest

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

References

Bennett, N., Lemoine, J., 2014. What VUCA really means for you. Harv. Bus. Rev. 92 (January (1/2)), 2014.

Horney, N., Pasmore, B., O'Shea, T., 2010. Leadership agility: a business imperative for a VUCA world. Human Resource Plann. 33 (4), 34.

Morariu, C., Răileanu, S., Borangiu, T., Anton, F., 2019. A Distributed Approach for Machine Learning in Large Scale Manufacturing Systems, Service Orientation in Holonic and Multi-Agent Manufacturing, Studies in Computational Intelligence, Vol. 803., pp. 41–52, Springer.

Luo, J., Hong, T., Meng, Y.U.E., 2018. Real-time anomaly detection for very short-term load forecasting. J. Mod. Power Syst. Clean Energy 6 (2), 235–243.

Pereira, J., Silveira, M., 2018. Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In: 17th IEEE Int. Conference on Machine Learning and Applications (ICMLA), IEEE Xplore Digital Library, pp. 1275–1282.

Arasu, A., et al., 2016. Stream: the Stanford Data Stream Management System, Data Stream Management. Springer, Berlin, Heidelberg, pp. 317–336.

Kreps, J., Narkhede, N., Rao, J., 2011. Kafka: a distributed messaging system for log processing. In: Proceedings of the NetDB, June 12, 2011, Athens, Greece, pp. 1–7, ACM 978-1-4503-0652-2/11/06 http://notes.stephenholiday.com/Kafka.pdf.

Wang, J., Ma, Y., Zhang, L., Gao, R.X., Wu, D., 2018. Deep learning for smart manufacturing: methods and applications. Int. J. Ind. Manuf. Syst. Eng. 48 (July Part C), 144–156.

Shao, S.Y., Sun, W.J., Yan, R.Q., Wang, P., Gao, R.X., 2017. A deep learning approach for fault diagnosis of induction motors in manufacturing. Chinese J. Mech. Eng. 30 (6), 1347–1356.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.

He, Q.P., Wang, J., 2007. Fault detection using the k-nearest neighbour rule for semiconductor manufacturing processes. IEEE Trans. on Semiconductor Manufacturing 20 (4), 345–354 (2007).

Sutskever, O., Vinyals, Q., Le, V., 2014. Sequence to sequence learning with neural networks. Adv. Neural Inf. Process. Syst., 3104–3112.

Mikolov, T., Karafiat, M., Burget, L., Cernock, J., Khudanpur, S., 2010. Recurrent neural network-based language model. In: INTERSPEECH 2010, 11th Annual Conference of the Int. Speech Communication Association, Makuhari, Chiba, Japan, pp. 1045–1048, Sept. 26-30, 2010.

Chandramitasari, W., Kurniawan, B., Fujimura, S., 2018. Building deep neural network model for short term electricity consumption forecasting. In: 2018 Symposium on Advanced Intelligent Informatics (SAIN), IEEE, pp. 43–48.

Sundermeyer, M., Schluter, R., Ney, H., 2012. LSTM Neural Networks for Language Modeling. INTERSPEECH, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.248.4448&rep=rep1&type=pdf.

Williams, J.D., Zweig, G., 2016. End-to-end Lstm-based Dialog Control Optimized With Supervised and Reinforcement Learning, arXiv Preprint arXiv:1606.01269. https://arxiv.org/pdf/1606.01269v1.pdf.

Liu, C., C, Jin, Z., Gu, J., Qiu, C., 2017. Short-term load forecasting using a long short-term memory network. In: 2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe), Torino, pp. 1–6, 2017.

Viswanadham, N., Johnson, T.L., 1988. Fault detection and diagnosis of automated manufacturing systems. Decision and Control, Proceedings of the 27th IEEE Conference on. IEEE, IFAC Proceedings Volumes 21 (15), 95–102.

Heshan, F., Surgenor, B., 2017. An unsupervised artificial neural network versus a rule-based approach for fault detection and identification in an automated assembly machine. Robot. Comput. Integr. Manuf. 43 (2017), 79–88.

Shin, S.Y., Kim, Y.M., Meilanitasari, P., 2019. A holonic-based self-learning mechanism for energy-predictive planning in machining processes. Processes 7 (10), 739.

Tong, Y., Li, J., Li, S., Li, D., 2016. Research on energy-saving production scheduling based on a clustering algorithm for a forging enterprise. Sustainability 8 (2), 136.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51.1 (2008), 107–113.

Morariu, C., Morariu, O., Borangiu, T., 2012. Manufacturing service bus integration model for implementing highly flexible and scalable manufacturing systems. IFAC 45.6, 1850–1855.

Lange, K., Retrieved 24 November 2019 2016. The Little Book on REST Services. Copenhagen. https://www.kennethlange.com/the-little-book-on-rest-services/.

Anon, 2020. Gunicorn-Python WsgiSGI HTTP Server for UNIX. https://gunicorn.org/.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., 2011. Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12 (2011), 2825–2830.

Valckenaers, P., Van Brussel, H., Bruyninckx, H., Saint Germain, B., 2011. Predicting the unexpected. J. Comput. Ind. 62 (August 6), 623–637, Elsevier.

IBM, 2019. IBM ILOG CPLEX Optimization Studio. https://www.ibm.com/products/ilog-cplex-optimization-studio.

Brownlee, J., 2016. Sequence classification with LSTM recurrent neural networks in python with keras. In Deep Learning for Natural Language Processing, Machine Learning Mastery.