A flexible alarm prediction system for smart manufacturing scenarios following a forecaster–analyzer approach

Article *in* Journal of Intelligent Manufacturing · June 2021

DOI: 10.1007/s10845-020-01614-w

CITATIONS
2

READS
142

3 authors, including:

Kevin Villalobos
Universidad del País Vasco / Euskal Herriko Unibertsitatea
5 PUBLICATIONS **9** CITATIONS

SEE PROFILE

Arantza Illarramendi
Universidad del País Vasco / Euskal Herriko Unibertsitatea
186 PUBLICATIONS **3,204** CITATIONS

SEE PROFILE

manuscript No.
(will be inserted by the editor)

A Flexible Alarm Prediction System for Smart Manufacturing Scenarios Following a Forecaster-Analyzer Approach

Kevin Villalobos · Johan Suykens · Arantza Illarramendi

Received: date / Accepted: date

Abstract The introduction of data-related information technologies in manufacturing allows to capture large volumes of data from the sensors monitoring the production processes and different alarms associated to them. An early prediction of those alarms can bring several benefits to manufacturing companies such as predictive maintenance of the equipment, or production optimization. This paper introduces a new system that allows to anticipate the activation of several alarms and thus, warns the operators in the plants about situations that could hamper the machines operation or stop the production process. The system follows a two-stage *forecaster-analyzer* approach on which first, a Long Short-Term Memory Recurrent Neural Network based *forecaster* predicts the future sensor's measurements and then, distinct *analyzers* based on Residual Neural Networks determine whether the predicted measurements will trigger an alarm or not. The system supports some features that make it particularly suitable for Smart Manufacturing scenarios: on the one hand, the *forecaster* is able to predict the future measurements of different types of time-series data captured by various sensors in non-stationary environments with dynamically changing processes. On the other hand, the analyzers are able to detect alarms that can be modeled with simple rules based on the activation condition, and also more complex alarms on which it is unknown when the activation condition will be fulfilled. Moreover, the followed approach for building the system makes it flexible and extensible for other predictive analysis tasks. The system has shown a great performance to predict three different types of alarms.

K. Villalobos · A. Illarramendi
Department of Computer Languages and Systems, University of the Basque Country UPV/EHU, Donostia - San Sebastián, Spain, 20018
E-mail: kevin.villalobos@ehu.eus, a.illarramendi@ehu.eus

J. Suykens
Katholieke Universiteit Leuven, Department of Electrical Engineering ESAT-SISTA, B-3001 Leuven, Belgium
E-mail: johan.suykens@esat.kuleuven.be

Keywords Alarm Prediction · Data-driven Predictive Maintenance · Long Short-Term Memory (LSTM) · Residual Neural Networks (ResNet) · Time Series Forecasting

1 Introduction

The introduction of data-driven [46] economy in the manufacturing industry has promoted the so called fourth industrial revolution or "Industry 4.0", also referred to as "Smart Manufacturing", which is defined in [8] upon two main concepts: the compilation of manufacturing records of products and the application of artificial intelligence techniques to analyze those records. Thus, the captured raw data (time series generated by the continuous operation of the manufacturing process or equipment to be analyzed) are usually stored in cloud computing infrastructures [59] for further analysis processes (product quality [11], fault detection [19], predictive maintenance of equipment [48], etc.).

In these smart manufacturing contexts, equipment maintenance plays an important role, and directly affects the service life of equipment and its production efficiency. Therefore, several analysis methods are appearing to address a proactive maintenance of the equipment; among which many of them manage different types of alarm systems to control the production process, and warn the operators in the plant about situations that could hamper the machine operation or incur in stops in the production process [48]. Overall, those alarm systems play a prominent role in maintaining plant safety and operation efficiency of modern industrial plants, by keeping the processes with normal operating ranges [51].

However, sometimes, in those systems the activation of the alarms is so close to the issue that there is no action-margin for the operators to manage the situation [25]. But, if the activation of the alarms is predicted early enough, the settings of the machine could be reconfigured in order to avoid stops in the production process or hampering the machine [51], [24]. In fact, the design of mechanisms to generate predictive alarms in order to forecast upcoming critical abnormal events has been stated as one of the open research problems in alarm systems [51], as it affects directly to the Overall Equipment Efficiency (OEE = Availability (A) * Performance (P) * Quality (Q)). For example, in the real context presented in Section 3, an early prediction of the *Plastic Temperature not Reached in the Die Entry Alarm* could lead to avoid bad quality (Q) products, by increasing the resistors' temperature, and a *Molten Resistor or Broken Thermocouple Cable in Die Zone 2 Alarm* could allow the operators in the plant to perform a proactive maintenance of the equipment to avoid possible damages in the machine or its components, by turning on the fans that cool down the resistors (A & P).

Different proposals have attempted to predict alarm activations in industrial scenarios with different approaches. For example, in [62], records of previous alarm activations are used to predict the most critical alarms; in [25], different features, extracted from measurements made by different detectors installed along rail tracks, are used to predict different alarms; and in [24], sensors data are used to predict the future measurements of the sensors and detect if an alarm will be triggered or not in the predicted values. The system presented in this paper also uses this last approach. However, the main difference between both works resides in the alarm detectors; while in [24], a binary classifier has been built based on

the activation condition of the alarm; the proposal presented in this paper, uses deep learning models to predict the alarms. In this regard, although binary classifiers based on the activation condition could be useful for some use cases, they are limited to predict alarms whose activation condition is based on simple rules known *a priori* (e.g., a threshold), while deep learning-based classifiers are able to predict these kind of alarms, but also more complex alarms whose activation condition cannot be modeled with simple rules or it still remains unknown.

The main contribution of this paper resides in the development of a flexible alarm prediction system that is able to predict different types of alarms that can be produced on a real smart manufacturing scenario. The system follows a two-stage *forecaster-analyzer* approach on which first, a *forecaster*, predicts the future measurements of the time-series data captured by various sensors; and then, distinct *analyzers* determine if the predicted measurements will trigger an alarm or not. Unlike other proposals, it supports some features that make it particularly suitable for Smart Manufacturing scenarios: on the one hand, the built system is suitable for multi-sensor time-series data forecasting in non-stationary environments such as smart manufacturing scenarios with dynamically changing processes. On the other hand, it is able to detect alarms that can be modeled with simple rules based on the activation condition, and also more complex alarms (see Section 3.2) on which it is unknown when the activation condition will be fulfilled, and it has shown the possibility of dealing with unseen situations that can emerge unexpectedly. Furthermore, the followed approach to build the system makes it easily extensible to other predictive analysis tasks, since the predicted measurements of the sensors could be used for other processes such as anomaly detection, prediction of other types of alarms, etc.

Concerning the used deep learning techniques for building the system, the forecaster has been built by using a Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) based model that predicts the next values of the time-series data captured from 11 different sensors implanted in a real extruder machine, for three different time horizons (5, 10 and 15 minutes) with an average Root Mean Squared Error (RMSE) of 0.00852, 0.01215 and 0.01737 (respectively). The *analyzers* have been built by using Residual Neural Networks (ResNet) based classifiers and have shown a great performance to predict three different types of alarms (with an area under the Relative Operating Characteristic (ROC) curve value of 0.99937, 0.99270 and 0.97381 respectively). Moreover, authors would like to notice that even though time series are a very common data type, most of the available systems cannot inherently accommodate and support the data sizes and analytics required by smart manufacturing scenarios, where fulfilling the strict requirements of such scenarios is a challenging goal, involving many interesting research problems [33].

The paper is structured as follows: Section 2 presents a review of related work; Section 3 presents the context of the alarm prediction system; Section 4 presents the built model for predicting multivariate time-series data; Section 5 presents the built models for detecting alarm activations in the predicted data; Section 6 shows the performance of the system for predicting three different types of alarms, and finally; Section 7 shows the conclusions of the realized work and some further research directions.

2 Related Work

The increasing interest among manufacturers in exploiting the potential of large volumes of manufacturing data for diverse purposes [7] such as the control of product quality, the predictive maintenance of equipment, fault detection, etc., has lead to the introduction of data-driven artificial intelligence techniques in these scenarios, to conduct different types of analysis over the captured data. For example, in a similar scenario to the one considered in this paper (the particular context of the manufacturing based on extrusion processes), in [11], different regression models are used for predicting the product quality, based on the predicted diameters of the extruded tubes, in order to optimize their production process.

Moreover, in those smart manufacturing scenarios in which the widespread deployment of sensors and Industrial Internet of Things (IIoT) devices [4] allow to capture huge volumes of data (which is essential for approaches that use artificial intelligence techniques [26]), the automatic feature learning and high-volume modelling capabilities of deep learning, provide advanced analytics tools [50]. This, together with the increasing popularity of deep learning, has promoted the application of deep learning techniques to manufacturing data and many researchers are advocating for their use to boost data-driven applications in smart manufacturing scenarios [50].

One of the most interesting applications of deep learning techniques to manufacturing data is the predictive maintenance of equipment, since it directly affects the service life of equipment and its production efficiency [48]. Thus, different methods are appearing to address a proactive maintenance of the equipment; for example, in [56], a data-driven bearing performance degradation assessment method based on Long Short Term Memory (LSTM) Recurrent Neural Networks (RNNs) is proposed; in [54], an approach for fault prognosis with the degradation sequence of equipment based on LSTM-RNNs is proposed; and in [29], a LSTM Encoder-Decoder model is used for multi-sensor prognostics using an unsupervised health index.

Besides those methods, different types of alarm-systems [51] have been also used in order to conduct a predictive maintenance of equipment. These systems control the production process and warn the operators in the plant about situations that could hamper the machine operation or incur in stops in the production process [48]. However, sometimes, in these systems the activation of the alarms is so close to the issue that there is no action-margin for the operators to perform a proactive maintenance of the equipment [25]. In such cases, an early prediction of the alarms' activation, will grant an extra time for the reconfiguration of the settings, controlling the production process in order to avoid production stops or hampering the machine. Therefore, the design of early alarm prediction systems has been stated as one of the open research problems in alarm systems [51].

Regarding the early prediction of alarms, different works can be found in smart manufacturing scenarios. For example, in [62], a dynamic alarm prediction algorithm is applied to an industrial case study to predict critical alarms by using a probabilistic model based on a *n-gram* model and sequences of previous alarm activations. In [24], an alarm prediction system has been built by using autoregressive Least Squares Support Vector Machines (LS-SVM) models, to predict the activation of a temperature alarm associated to the bearings of a steel production machine, and in [25], a customized SVM model has been built for alarm prediction

in a large-scale railroad network. Finally, in [5], an alarm prediction method based on word embedding and LSTM neural networks is presented to predict the next alarm in a process setting. The system presented in this paper follows a *forecaster-analyzer* approach that combines LSTM neural networks [17] to forecast the future measurements of various sensors, with residual neural networks (ResNet [15]) to analyze (or classify) the alarms in the predicted values.

Regarding the approach used to build the alarm prediction system; in [24], the captured data by the sensors are forecasted and used to predict alarm activations following a similar approach to the *forecaster-analyzer* proposed in this paper. Nevertheless, there are significant differences between both works: on the one hand, in [24], an autoregressive LS-SVM model is used to predict a unique sensor's future measurements, while in this work, a LSTM-based model has been used to predict multivariate time series captured by multiple sensors. The use of a multivariate time series forecaster avoids having a specific model for each sensor, and also allows to capture interdependencies between different time series and predicting alarms on which various sensors could be involved (e.g., *Incorrect Temperature Alarm* in Section 3.2). On the other hand, in [24], the used analyzer is based on a rule on which an alarm is predicted if in the forecasted temperature values, the maximum temperature is reached at least once, while the system proposed in this paper uses residual neural networks-based classifiers. The use of these classifiers leads to more general purpose analyzers that are able to detect different alarms, including alarms which can be detected thanks to a rule based on the activation condition (e.g., *Plastic Temperature not Reached in the Die Entry Alarm*) but also more complex alarms that cannot be detected by this kind of rules (e.g., *Molten Resistor or Broken Thermocouple Cable in Die Zone 2 Alarm*).

Concerning the neural networks used to build the alarm prediction system, it can be seen in the literature, that on the one hand, LSTM recurrent neural networks have been already successfully used for forecasting time series of sensor data. For example, in [18], LSTM recurrent neural networks are used for forecasting time-series data coming from different sensors monitoring environment variables in a farm-monitoring context, and in [60], LSTM recurrent neural networks are used for forecasting the time-series data from 33 sensors of a cooling pump in a power station. On the other hand, neural networks have also been already used for time series classification purposes. For example, in [53], 44 different time series databases of different nature are used to compare the performance of 9 time series classifiers including three deep learning classifiers based on neural networks. Furthermore, [20] extends the benchmark to 85 time series databases including also multivariate time series databases to compare 9 deep learning classifiers based on neural networks, on which the ResNet classifier achieves the best performance. In both benchmarks, the ResNet classifier has been demonstrated to perform well on classifying time series datasets of different nature, an interesting property for smart manufacturing scenarios where multiple heterogeneous sensors produce different types of time series.

Two main aspects distinguish the proposed system from those mentioned before. Firstly, the use of a LSTM neural networks-based forecaster to predict multivariate IIoT devices time-series data in a real smart manufacturing scenario with dynamically changing processes (the system presented in [5] also uses LSTM neural networks for alarm prediction; however, that system predicts the activation of the alarms by using previous alarm activation data instead of the time-series data cap-

tured by the sensors); and secondly, the use of deep learning-based analyzers that are able to predict those kind of alarms on which the activation condition could be modeled by a rule (as the system presented in [24] does), but also more complex alarms that cannot be modeled with rules based on the activation condition. Moreover, it has shown the possibility of adapting the used analyzers to deal with unseen situations which can emerge unexpectedly. Finally, although the mentioned deep learning-based models have been independently used in smart manufacturing scenarios, to the best of the authors' of this paper knowledge, these systems have not been already combined for predictive maintenance tasks in manufacturing scenarios.

3 Context of the Alarm Prediction System

This section provides details about the main elements involved in the setting of the alarm prediction system; in particular, details about the main features of the captured time series and alarm data, the accomplished tasks for pre-processing the data, and the followed approach for implementing the system are given.

3.1 Time-Series and Alarm Data

The access to real-world data was facilitated by the collaboration with a Capital Equipment Manufacturer (CEM) that has installed several sensors in the machines that it manufactures. Those sensors register time-series data with a continuous measurement at 1Hz frequency (i.e., one measurement per second) of a variety of equipment setting parameters and physical magnitudes (temperatures, pressures, etc.) related to the raw materials, production processes and industrial equipment from a plastic bottles production plant based on an extrusion process. Associated to those sensors, the CEM has also defined some alarms that are triggered under different conditions established over the measurements taken by the sensors. Those alarms can allow the operators in the plant to conduct a proactive management of the different controls in the machine for a predictive maintenance of the equipment. The data from the sensors implanted in an extruder machine of a real production plant and their associated alarm data (i.e., time series of alarm activation events registered in a log mode with a register per event) have been captured by using a REST API provided by the CEM.

Fig. 1 shows the scheme of an extruder machine on which the CEM has implanted several sensors, and Table 1 shows the type of captured time series, the type of sensor and their associated alarms with their activation condition given by the domain experts from the CEM. The captured data from the 01-12-2018 to the 28-02-2019 have been used to train and test the models, and the captured data from the 01-03-2019 to the 31-03-2019 have been used to evaluate the models. Fig. 2 shows as an example a four-hour sub-sequence of the *Melting Temperature* time series on which an alarm (vertical red line) has been triggered.

High-frequency sensors capturing data during long periods of time, lead to large-scale raw time-series data that hamper the performance of machine learning models which usually scale poorly to high dimensional data [27]. Thus, in order to reduce the dimensionality of the data, the time series have been aggregated by

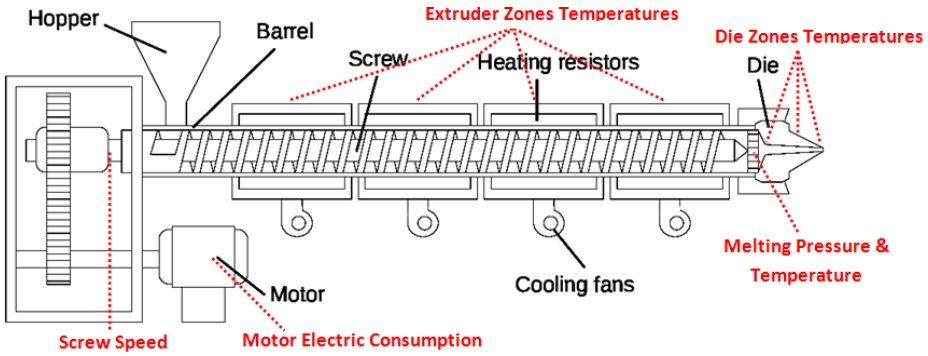

Fig. 1 Different Sensors (in red) Implanted on an Extruder Machine

Table 1 Properties of Captured Time-series Data and the Associated Alarms

Time series	Sensor type	Associated Alarm	Activation Condition
Melting Temperature	Thermal (°C)	Plastic temperature not reached in the die entry	Melting Temperature $<$ 170 °C
Extruder Temperatures Zones [1-4] (Extruder) Zone 5 (Union) Zone 6 (Filter) Zones [1-4] (Die)	Thermal (°C) (×10)	Incorrect temperature	Temperature $>$ (set-temperature $+$ error-margin) or Temperature $<$ (set-temperature - error-margin) in any of the zones
Extruder Temperatures Zone 2 (Die)	Thermal (°C)	Molten resistor or broken thermocouple cable in die zone 2	The heat resistor in the second zone of the die is molten, or the thermocouple cable is broken

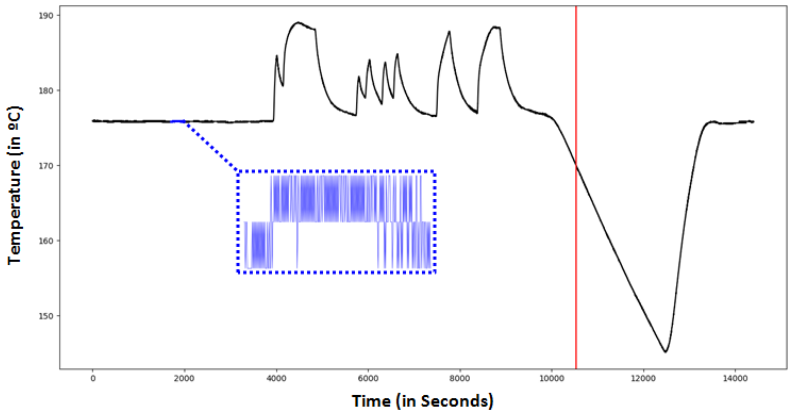

Fig. 2 Sub-sequence of Melting Temperature Time Series on which an Alarm has been Activated

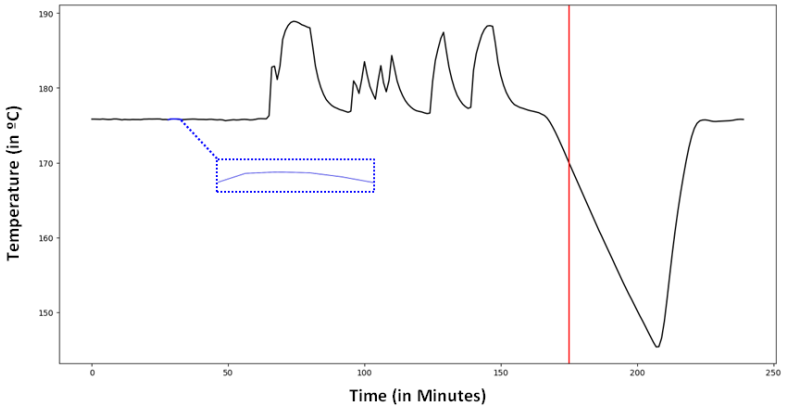

Fig. 3 Sub-sequence of Melting Temperature Time Series on which an Alarm has been Activated (after Pre-processing)

minute, using the Piecewise Aggregate Approximation technique described in [21]. This aggregation also allows to reduce the complexity of the time series forecasting problem, as it reduces the number of steps to predict, and as can be seen in [24], the performance of the models for predicting future values decreases as the number of steps to predict increases. For example, without any aggregation, in order to build a model that predicts the measurements of the following 5 minutes, the model would need to predict 300 steps-ahead, while aggregating the data by minute it will only need to predict 5 steps-ahead.

Furthermore, the measurements of the implanted sensors present some inaccuracies (i.e., noise) due to the precision of the sensors which introduces an additional complexity into the ability of the models to predict the under-laying behaviour of the time series. Thus, in order to remove the noise that hampers the performance of the models, data has been filtered by using the Discrete Fourier Transform [2]. Fig. 3 shows the same time series presented in Fig. 2 after aggregating the data by minute and removing the noise. Missing values and outliers have also been removed from the raw data.

Finally, the pre-processed time-series data have been integrated in a dataset on which each timestamp is associated to the measurements of all the sensors (i.e., a dataset with a structure of ($timestamps \times num\_sensors$)). This dataset has been the one used to generate the input data for the models. However, these data do not meet the specific necessities of the selected deep learning models, and thus, before building and training the models, first, data have been normalized in the range [-1, 1], and then, some transformations have been applied in order to meet the requirements of the models (see sections 4.1 and 5.1 respectively).

3.2 Alarm Types

Three different alarm types have been considered for building the alarm prediction system (see Table 1). For each alarm an analyzer has been built to determine if in a given time-series sub-sequence a particular type of alarm will be triggered or not.

- *Plastic Temperature not Reached in the Die Entry:* This alarm is associated to the thermal sensor implanted in the entry of the die, to measure the *melting temperature* of the plastic. This temperature directly affects the viscosity of the melted plastic that at the same time affects the quality of the final product. Thus, in order to avoid bad quality products and stops in the production process, a specific alarm has been defined to ensure the correct temperatures. This alarm is triggered if the *melting temperature* is lower than 170°C.
- *Incorrect Temperature:* This alarm is triggered if in any of the extruder zones the measured temperature is not correct. The correct temperature is established in the production plant and it is bounded between the established values ± an error margin.
- *Molten Resistor or Broken Thermocouple Cable in Die Zone 2*[1]*:* This alarm is triggered if in the second zone of the die, the heat resistor has been molten or if the thermocouple cable has been broken.

3.3 An Alarm Prediction System Following a Forecaster-Analyzer Approach

As mentioned before, the alarm prediction system follows a two-stage *forecaster-analyzer* approach on which first, the future measurements of the sensors are forecasted; and then, different types of alarms are predicted over the forecasted data, by using three different analyzers (i.e., classifiers trained to detect interesting patterns matching alarm activations). Fig. 4 shows an example of this approach by using the built forecaster for predicting the future values of the *melting temperature* time series and an analyzer that tries to detect if the *Plastic Temperature not Reached in the Die Entry Alarm* will be activated or not in the predicted values. In the *training phase*; first, the forecaster is built and trained by using time-series sub-sequences to predict the following values of the time series, and then, an analyzer is built and trained using those sub-sequences to determine if in a given sub-sequence an alarm will be activated or not. In the *deployment phase*; the future measurements of the sensor (time-series sub-sequences) are predicted by using the built forecaster and introduced into the corresponding analyzer that determines if an alarm will be triggered or not.

The prediction of those alarms, could allow the operators in the plant to reconfigure the settings of the machines in a proactive way in order to avoid bad quality products or hampering the machine. Furthermore, the followed approach is easily extensible, since the predicted data by the forecaster could also be used to anticipate other kind of events by building new analyzers (e.g., abnormal behaviours or faults in the machine or its components).

Models have been built using Tensorflow [1] and Keras [6] libraries and they have been deployed using the Google AI Platform [13]. In particular the training and prediction jobs have been executed on a *n1-highcpu-16*[2] machine (as master

[1] Although this alarm type has been associated to all the resistors or thermocouple cables from the different zones of the extruder, only the one associated to the second zone of the die has been considered, because in the selected period of time is the only one that has been triggered.

[2] Machine types in Google Compute Engine: `https://cloud.google.com/compute/docs/machine-types`.

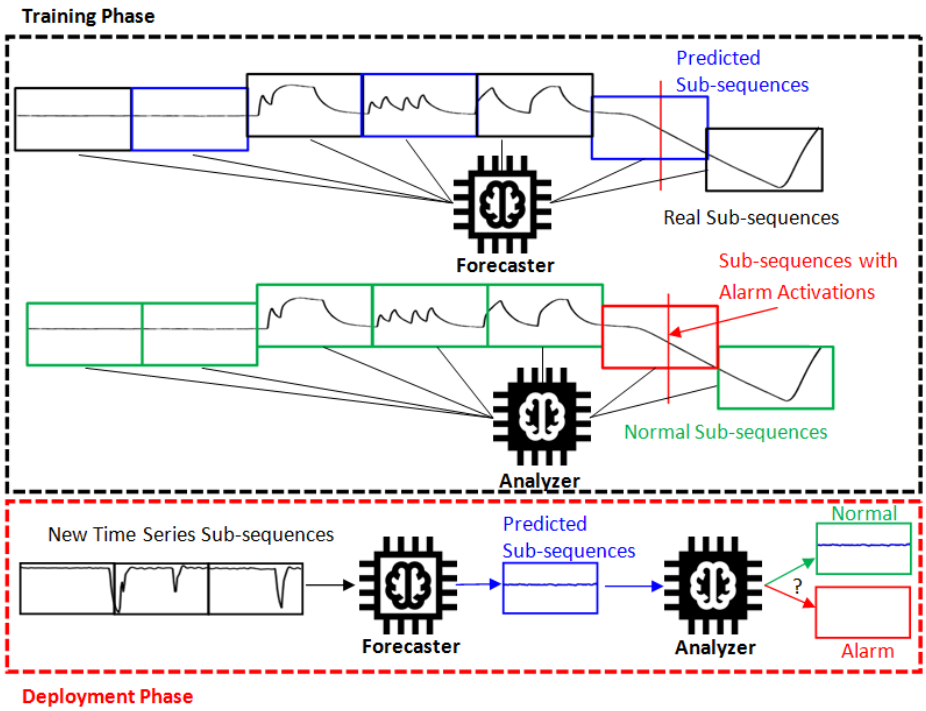

Fig. 4 Alarm Prediction System Following a Forecaster-Analyzer Approach

node) with a *standard_p100*[3] GPU accelerator. Moreover, the Google AI Platform allows to build and train the models over different clusters of GPU workers, which can result particularly interesting for smart manufacturing scenarios dealing with big volumes of data.

4 Industrial Sensors Time-Series Data Forecasting

Time series forecasting is an important research topic in the domain of science and engineering, in which past observations of the data are collected and analyzed to develop a model that can predict future observations [22]. Over the years, various forecasting models have been developed in the literature. In particular, for time series forecasting, Autoregressive Integrated Moving Average (ARIMA) models have been widely used, and more recently, Artificial Neural Networks (ANNs) [58].

In the literature, both approaches have been compared in different application domains with mixed results [58] (in some cases, ANN perform better than classic time series forecasting models, whereas in other cases, classical time series models make more accurate predictions or both show a similar behaviour), mainly due to the complex nature of real-world problems [57]. However, in the particular context of sensor and IIoT devices time-series data forecasting, recent works (such

[3] GPU types in Google AI Platform: `https://cloud.google.com/ml-engine/docs/using-gpus`.

as [18] and [60]) have shown that the inefficiency of classical time series models to capture long-term multivariate dependencies of the data coming from multiple devices of different nature [49], makes ANN-based models more suitable than classical models. In particular, Deep Neural Networks (DNN) of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) [49], [39], [52] have been widely used for time series forecasting tasks.

In order to evaluate the behaviour of different types of models with data coming from a real manufacturing scenario, in this work, three different types of models have been built to predict the future measurements of the sensors mentioned in Table 1, in three different time horizons that are relevant for the considered scenario: a CNN model, a LSTM-RNN model and an ARIMA model. Firstly, in the following sub-sections, the steps followed to prepare the data for the prediction models are presented, and then, the built models together with their performance evaluation.

4.1 Forecasting Data Preparation

As mentioned before, the pre-processed data (see Section 3.1) do not meet the specific necessities of the selected deep learning models and thus, before data can be used in those models, they must be prepared according to the input/output specifications of the models. To prepare the data, a sliding-window approach [35] has been followed to transform the pre-processed dataset into a dataset composed of time-series sub-sequences with the measurements of all the sensors (i.e., a dataset with a structure of ($num\_sub\text{-}sequences \times window\text{-}length \times num\_sensors$), where $window\text{-}length = input\_sequence\_length + output\_sequence\_length$ (see Sections 4.2.2 and 4.2.3)).

Moreover, the deep learning-based time series forecasting models use a time-series sub-sequence as input, to learn how to predict the future sub-sequences of measurements with a given time horizon (i.e., $output\_sequence\_length$). Thus, the first $input\_sequence\_length$ steps (i.e., minutes) will serve as input for the forecasting models, and the following $output\_sequence\_length$ steps as output (the target values to predict). Therefore, the dataset mentioned above has been split into two datasets, an input dataset with a structure of ($num\_sub\text{-}sequences \times input\_sequence\_length \times num\_sensors$), and an output dataset with a structure of ($num\_sub\text{-}sequences \times output\_sequence\_length \times num\_sensors$).

The selection of the time horizons has been determined by two constraints: on the one hand, it is known from the R&D director of the CEM providing the real data that the effects of adjusting some of the settings of the production process may not be noticed until a few minutes (up to 15 minutes) have elapsed. On the other hand, the performance of the models for predicting future values decreases as the number of steps to predict increases (as can be seen in [24]). Therefore, at each prediction, 5 steps (i.e., 5 minutes) are forecasted and then, those predictions are used to predict further time horizons (10 and 15 minutes) following the approach described in [45], that uses the predicted sub-sequence together with the input data to predict the next sub-sequence.

4.2 Time Series Forecasting Models

This section presents the built models in order to test their performance in both univariate and multivariate time series forecasting. For each type of model, a univariate time series forecaster has been built by using the *Melting Temperatures Sensor Data* and a multivariate time series forecaster by using the data of all the sensors shown in Table 1. The built models have been trained and evaluated following the *rolling strategy* described in [43] on which the model predicts the future measurements of the sensors using the last available measurements. This strategy has been applied over the prepared training and evaluation datasets. Next the build models are presented.

4.2.1 ARIMA Model

ARIMA is a linear regression-based forecasting approach that captures temporal structures in time-series data. The acronym ARIMA stands for *Autoregressive*[4] (AR) *Integrated*[5] (I) *Moving Average*[6] (MA) [43] and captures the key components of the model. These three components are specified as parameters when building an ARIMA(p,d,q) model, where p is the lag order (i.e., the number of lag observations used in model training); d is the degree of differencing (i.e., the number of differencing items applied); and q is the order of moving average (i.e., the size of the moving average window). ARIMA models were initially conceived for univariate time series forecasting; however, some generalizations of these models have been developed to allow them involving multiple variables. Such is the case of Vector Autoregresive (VAR) [28] models that capture the linear inter-dependencies among multiple time series introduced as variables. In these models, each variable has a linear function explaining its evolution based on its own lagged values, the lagged values of the other variables in the model, and an error term. When building a VAR(p) model, although usually the only required parameter is the lag-order (p), the model requires all the variables to have the same order of integration; thus, before building the model the data has been differenced with a degree of one (d=*1*).

In this work, an ARIMA(4,1,0) model has been built for univariate time series forecasting, and a VAR(4) model has been built for multivariate time series forecasting. The selection of the parameters has been done with a grid search (considering the following parameter ranges: p=[1-10], d=[1-5], q=[0-10]), using the *auto_arima* function and the *RollingForecastCV* model selection function of the *pmdarima* package [44]. For building the models, the approach followed in [43] has been used, on which the model performs multi-step *out-of-sample* forecasting with *re-estimation* (i.e., each time the model is re-fitted to build the best estimation model).

[4] A model that uses the dependent relationship between an observation and some number of lagged observations.

[5] The differencing of raw observations in order to make the time series stationary.

[6] A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

4.2.2 CNN Model

Convolutional Neural Networks (CNN) [23] are a specialized type of neural networks for processing data that has a known, grid-like topology (including time-series data) [39]. These networks, employ a mathematical operation called convolution between the input data and a filter or a kernel, usually alternated with pooling operations to generate a *feature map* that is finally connected to a fully-connected neural network that analyzes the features for classification and prediction tasks [61]. The impressive success arisen by CNNs in the domain of computer vision (powering tasks like image classification, object recognition, etc.) has led researches and practitioners to apply them in other domains such as time series classification [61] and time series forecasting [52].

In this work, different CNN-based models with different parameter configurations have been built in order to select the most appropriate one for the considered scenario. These models are composed by blocks (up to three) of a 1D convolutional layer and a *max-pooling* layer (takes the highest value from each area scanned by the CNN) followed by a flatten layer to reduce the feature maps to a one-dimensional vector and a fully-connected (dense) layer that interprets the features extracted by the convolutional part of the model to predict the future measurements of the sensors. For selecting the best parameter configuration, a grid search has been done by using the GPyOpt library[7] [47] considering the parameter values shown in Table 2. Two constraints have been defined for the grid search: the first one, to ensure that the number of filters of a convolutional layer (in models with more than one layer) is the half of the precedent layer's number of filters; and the second one, to ensure that the kernel size in the subsequent layer is equal or lower than the precedent layer.

The built models with the different parameter configurations have been trained and evaluated five times and the best model, based on the obtained RMSE on the evaluation dataset, has been selected. Taking into account the results of the parameter optimization process, a CNN model has been built for univariate time series forecasting that uses a single convolutional block with 32 filters with a kernel size of 2 and the $ReLU$ activation function, and a pool-size of 2 in the pooling layer. For multivariate time series forecasting, the model that achieved the best performance was a model with a single convolutional block with 64 filters with a kernel size of 8 and the $ReLU$ activation function, and a pool-size of 2 in the pooling layer. The univariate and multivariate time series forecasting models have been trained using the *Adam* Optimizer with a learning rate of 0.002 and 0.001 (respectively), and the *mse* loss function during 400 and 300 epoch (respectively), with a *batch size* of 256 and an input sequence length 100 and 300 steps (respectively).

4.2.3 LSTM Model

Long Short-Term Memory (LSTM) [17] is a special kind of Recurrent Neural Network (RNN) capable of learning order dependence in sequence prediction problems. LSTM neural networks have the chain like structure composed by a set of

[7] A Bayesian Optimization tool for black-box functions that allows tuning automatically machine learning models' parameters.

[8] A hyphen (-) means that the parameter is not applicable for the model or that has not been considered.

Table 2 Deep Learning Models' Parameters[8]

Parameter Description	CNN	LSTM
Blocks of convolutional and max-pooling layers	1, 2, 3	-
Activation function of the convolutional layers	ReLU	-
N° of filters on each convolutional layer	64, 128, 256	-
Kernel size on each convolutional layer	2, 4, 6, 8	-
Pool size on each max-pooling layer	2, 3, 4	-
N° of hidden layers	-	1, 2, 3
N° of units (neurons) on each hidden layer	-	64, 128, 256
Input sequence length	100, 200, 300	100, 200, 300
Output sequence length	5	5
Loss function	mse	mse
Learning rate	0.001, 0.002, 0.005	0.001, 0.002, 0.005
N° of training epoch	100, 200, 300, 400	100, 200, 300, 400
Optimizer	adam, nadam	adam, nadam
Batch size	64, 128, 256	64, 128, 256

cells, typical of RNNs, on which each cell contains a cell state that allows the information to be kept for a long period of time [55]. In LSTM neural networks the information added or removed from the cell state is carefully regulated by structures called gates (composed out of a *sigmoid* neural network layer and a point-wise multiplication operation). A LSTM neural network has three of these gates controlling the cell state: A forget gate and an input gate that control which part of the information should be removed/reserved in the network; and an output gate that uses the processed information to generate the correct output [32]. LSTM neural networks have been explicitly designed to avoid the long-term dependency problem present in other recurrent neural networks. Their ability to remember information for longer periods of time allows them to perform well in diverse time series forecasting tasks for both, one-step-ahead forecasting [18], and multi-step-ahead forecasting [55].

In this work, different LSTM-based models with different parameter configurations have been built in order to select the most appropriate one for the considered scenario, following the same approach described in Section 4.2.2. Table 2 shows the considered parameter values for the optimization process. A constraint has been defined to ensure that the number of neurons of the subsequent layer (in models with more than one hidden layer) is the half of a precedent layer's number of neurons. Taking into account the results of the parameter optimization process, a *Vanilla LSTM* model has been built with a single layer and 128 neurons for both, univariate and multivariate time series forecasting. Both models have been trained by using the *Adam* optimizer with a learning rate of 0.001 and the *mse* loss function. Models have been trained during 300 and 400 epochs (respectively) with a *batch size* of 128 and an input sequence length of 300 steps.

4.3 Forecasting Models Evaluation

In order to select a suitable time series forecasting model, an instance of each of the models mentioned above (after selecting the best parameter configuration) has been built, and its performance has been evaluated. In general, to evaluate

the performance of that type of models, a metric is often defined in terms of the forecasting error, which is the difference between the actual (desired) and the predicted values. Different metrics have been used in the literature to measure the performance of the predictions (a review of them can be found in [41], together with their formula). However, each of them presents different advantages and limitations, and thus, there is not a universally accepted one by the forecasting academicians and practitioners [58]. Therefore, in this work three different error metrics have been selected to evaluate the performance of the forecasters: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).

Table 3 summarizes the performance results of the different models considered for forecasting the whole time series for the three different time horizons and over the two available datasets (train and evaluation). Among all these results, those related with the evaluation dataset (unseen data for the model) have been considered in order to select the most suitable forecasting model. The performance results show that when considering the RMSE metric, the LSTM-based model outperforms the ARIMA and CNN-based models in both univariate and multivariate time series forecasting. When considering MAE and MAPE metrics, on the one hand, for univariate time series forecasting, ARIMA-based models outperform LSTM and CNN-based models. On the other hand, for multivariate time series forecasting (which is the most relevant case for the considered scenario) ARIMA and LSTM-based models show a similar performance and both outperform CNN-based models. However, for near time horizons, ARIMA-based models show a better performance, while as the time window to predict increases, their performance is degraded and LSTM-based models show a better performance.

Table 3 Time Series Forecasting Evaluation Results

			Dataset - Steps - ahead					
			Train			**Evaluation**		
	Metric	**Forecaster**	**5 min**	**10 min**	**15 min**	**5 min**	**10 min**	**15 min**
Univariate	RMSE	ARIMA	0.00028	0.00281	0.01048	0.01426	0.02399	0.03049
		CNN	0.01313	0.02648	0.06689	0.00503	0.01291	0.02346
		LSTM	0.00249	0.01132	0.02875	0.00137	0.00577	0.01560
	MAE	ARIMA	0.00005	0.00047	0.00167	0.00019	0.00064	0.00165
		CNN	0.00402	0.00860	0.01551	0.00196	0.00451	0.00764
		LSTM	0.00068	0.00202	0.00403	0.00037	0.00131	0.00291
	MAPE	ARIMA	0.02696	0.25799	0.78174	0.02350	0.11597	0.73543
		CNN	0.88629	1.94116	3.58219	0.42255	1.33615	1.76896
		LSTM	0.36965	0.80791	1.59330	0.07121	0.72974	1.74319
Multivariate	RMSE	VAR	0.00017	0.00195	0.00807	0.01444	0.02436	0.03180
		CNN	0.02581	0.03723	0.04972	0.01716	0.02119	0.02588
		LSTM	0.00757	0.01219	0.02036	0.00852	0.01215	0.01737
	MAE	VAR	0.00005	0.00051	0.00203	0.00300	0.00458	0.00628
		CNN	0.01297	0.01633	0.01936	0.00882	0.01096	0.01290
		LSTM	0.00444	0.00547	0.00683	0.00409	0.00498	0.00582
	MAPE	VAR	0.03581	0.27815	1.41278	0.54049	1.36444	2.14099
		CNN	4.61300	5.81008	6.53746	1.62916	2.11659	2.43444
		LSTM	1.30136	1.74324	2.59245	0.63228	0.87252	1.18475

In addition to the achieved performance results, regarding the applicability of the built forecasters in real smart manufacturing scenarios; on the one hand, the proposed system should be flexible enough to take into account the non-stationary nature of this environments with dynamically changing industrial processes, that could hamper the performance of the built forecaster (e.g., changes in the machine operation mode, changes in the type of product to produce, etc.); on the other hand, the system should be suitable for making real time predictions in industrial contexts with big volumes of data produced by multiple sensors of different nature.

In this sense, it is worth mentioning that in order to make accurate predictions, the ARIMA models require to be re-estimated with the latest data before each prediction step. However, although this fact helps the model to make more accurate predictions (since it is always up to date with the newest data), it restricts the feasibility of its application to real world problems in the context of Smart Manufacturing, where the latest data is not always available (e.g., due to stops in the production process), and where constantly re-estimating models for real time predictions could be computationally expensive. Conversely, LSTM and CNN-based models are not re-estimated before each prediction step, a property that could result unfavorable if the environment conditions change. Nevertheless, these models can be updated with new data due to a specific requirement of certain circumstances (e.g., one of the raw materials has been changed) or they could be periodically updated (e.g., daily) to keep the models up to date. Moreover as it is stated in [32], LSTM neural networks have been explicitly designed to avoid the long-term dependency problem by remembering information for long periods of time, an interesting behavior, especially when the model has been trained with large time series (since they could capture and "remember" different operation modes of the machine under different circumstances). Thus, taking into account the results of the performed tests as well as the system applicability in Smart Manufacturing scenarios, LSTM neural networks have been selected to build the forecaster of the proposed system.

4.4 LSTM Forecaster Performance Results

A time series forecaster has been built to predict the future measurements of the sensors, by using the selected LSTM-based model. The built forecaster takes subsequences of the time-series data captured by 11 sensors implanted on an extruder machine as input (see Table 1), and it predicts a 5-step-ahead sub-sequence for each sensor as output (i.e., 11 sub-sequences of 5 sensor measurements corresponding with the following 5 minutes). These predictions will serve as the output for the first time horizon (5 minutes), and also as the input to predict recursively the next two time horizons (10 and 15 minutes) (see Section 4.1).

Table 4 shows the performance results of the selected model when predicting the future measurements of each sensor individually. The performance results are shown with the RMSE, MAE and MAPE metrics. However, in the following, the RMSE metric is used for presenting the performance results of the selected forecaster, for being the one corresponding with the loss function used to build the model (RMSE = $\sqrt{MSE}$). Although there is some variation in the RMSE obtained when predicting the different sensors' data, the built forecaster achieves a great performance with an average RMSE of 0.00852, 0.01215 and 0.01737 (re-

spectively for each time horizon on the evaluation dataset). Furthermore, if due to special requirements of the application scenario more precision is required for a particular sensor, a specific forecaster could be built to predict only the future measurements of that sensor in a more accurately way. Table 5 shows a comparison between the performance results of a specific forecaster for the *Melting Temperature sensor* and the multi-sensor forecaster; and Fig. 5 shows, as an example, the predicted time series for the *Melting Temperature* sensor in a sub-sequence of the evaluation dataset (3600 minutes), when using the multi-sensor forecaster (in green), and when using the specific forecaster (in red). Although the specific forecaster is more precise than the multi-sensor forecaster, the difference does not hamper the ability of the built analyzers to predict the alarms (see Section 6.1). Moreover, the multi-sensor forecaster allows to predict the data of various sensors at the same time, instead of building a specific forecaster for each sensor for those alarms associated to multiple sensors.

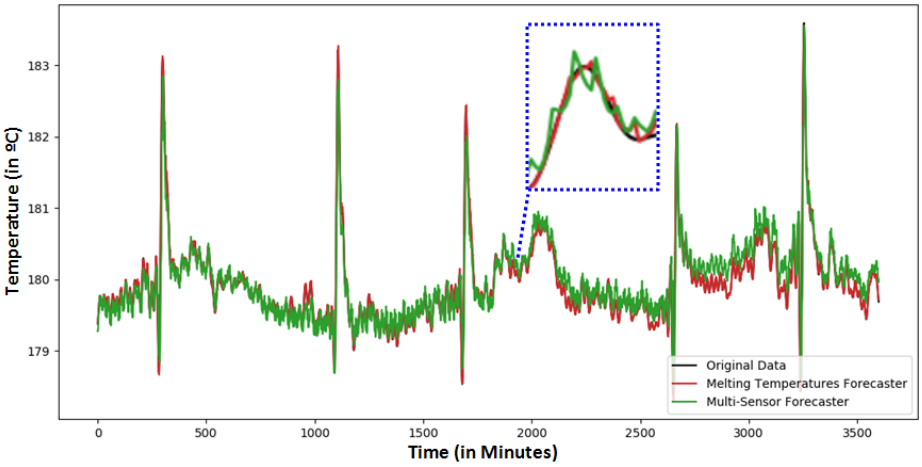

Fig. 5 Melting Temperatures Forecasting: Predicted Measurements vs Original Measurements (Time Horizon 5 Minutes)

4.5 Dealing with Non-stationary Environments

As mentioned in Section 4.3, in non stationary environments with dynamically changing processes, the data distribution can change over the time, yielding the phenomenon of *concept drift* [10]. In these environments, the common approach of training models in an offline manner using historical data could lead to models on which the performance decreases as time goes by and environment conditions change. Therefore, one desirable property of the models is their ability of incorporating new data. This ability to adapt to such *concept drift* can be seen as a natural extension for incremental learning systems, such as the neural networks-based models used in this work, which learn predictive models example by example and thus, they can update the decision model as new examples arrive [10].

Table 4 Forecasting Performance Results of Each Sensor for the Evaluation Dataset

		Steps - ahead		
Sensor	**Metric**	**5 min**	**10 min**	**15 min**
Melting Temperatures	RMSE	0.00564	0.01011	0.01556
	MAE	0.00412	0.00578	0.00617
	MAPE	0.56375	0.99342	2.40473
Extruder Temperature Zone 1	RMSE	0.01766	0.01819	0.02145
	MAE	0.00547	0.00579	0.00724
	MAPE	0.68267	0.72025	0.89809
Extruder Temperature Zone 2	RMSE	0.00955	0.01367	0.01809
	MAE	0.00221	0.00312	0.00456
	MAPE	0.40620	0.56167	0.84430
Extruder Temperature Zone 3	RMSE	0.01456	0.01866	0.02426
	MAE	0.00583	0.00665	0.00870
	MAPE	1.00995	1.12361	1.51917
Extruder Temperature Zone 4	RMSE	0.00831	0.01185	0.01788
	MAE	0.00213	0.00282	0.00415
	MAPE	0.40764	0.53567	0.72905
Extruder Temperature Zone 5 (Union)	RMSE	0.00550	0.00968	0.01525
	MAE	0.00388	0.00516	0.00552
	MAPE	0.49775	0.67376	0.81368
Extruder Temperature Zone 6 (Filter)	RMSE	0.00520	0.00950	0.01573
	MAE	0.00230	0.00262	0.00333
	MAPE	0.32786	0.41065	0.70757
Extruder Temperature Zone 1 (Die)	RMSE	0.00470	0.00844	0.01453
	MAE	0.00265	0.00296	0.00342
	MAPE	0.35323	0.49375	0.66072
Extruder Temperature Zone 2 (Die)	RMSE	0.00569	0.00967	0.01458
	MAE	0.00369	0.00505	0.00549
	MAPE	0.48412	0.67080	0.76792
Extruder Temperature Zone 3 (Die)	RMSE	0.00453	0.00831	0.01447
	MAE	0.00246	0.00308	0.00342
	MAPE	0.91350	1.85027	2.00073
Extruder Temperature Zone 4 (Die)	RMSE	0.01238	0.01559	0.01931
	MAE	0.01028	0.01178	0.01204
	MAPE	1.30838	1.56381	1.68630
Average	RMSE	0.00852	0.01215	0.01737
	MAE	0.00409	0.00498	0.00582
	MAPE	0.63228	0.87252	1.18475

Table 5 Multi-Sensor vs Specific Forecaster Prediction Error for Melting Temperatures (Evaluation Data)

		Steps - ahead		
Forecaster	**Metric**	**5 min**	**10 min**	**15 min**
Multi-sensor	RMSE	0.00564	0.01011	0.01556
	MAE	0.00412	0.00578	0.00617
	MAPE	0.56375	0.99342	2.40473
Melting Temperatures	RMSE	0.00137	0.00577	0.01560
	MAE	0.00037	0.00131	0.00291
	MAPE	0.07121	0.72974	1.74319

Two main approaches are used to adapt a decision model in order to address the concept drift [9]: blind approaches that update the decision model periodically without verifying if changes really occurred; and informed approaches that modify the decision model when changes are detected. In this last approach, several methods have been proposed to detect the concept drift (such as the *Page-Hinkley* method or the *ADaptive sliding WINdow* (ADWIN)), a survey of them can be found in [10]. Most of those methods detect when the concept drift occurs and then the models can be updated independently. Another interesting approach is the one proposed in [37], where the model is slightly updated when new data is available using an anomaly-score computed between the predicted values and the original ones. This anomaly score ensures that a high anomalous score, during large periods of time, due to concept drift, leads to continuous changes in the network parameters, whereas short term anomalies or outliers producing a high anomaly score do not produce significant updates in the network parameters. However, as stated in Section 4.3, updating the model every time new data is available in smart manufacturing scenarios could be computationally inefficient, and although there exist some research works in Online Deep Learning Neural Networks that learns on the fly [36], using a traditional neural network-based model (like the ones used in this work) to change the model itself (every time new data is available) could leverage dangerous consequences (performance degradation) in long term. Thus reaching a trade-off between both approaches could be more interesting.

In this work, a combination of both approaches is proposed to test the ability of the model to adapt to non-stationary environments. First, an anomaly-score has been used to detect when the model performance starts to degrade due to concept drift. For that, the RMSE between the predicted values and the real measurements is computed for every prediction made by the model as anomaly-score. If the mean of the obtained anomaly-score in a fixed period of time (i.e., the last hours) is higher than the obtained RMSE by the model when evaluated over a large period of normal operation mode (e.g., the evaluation dataset) plus an error margin, the model could be updated using a width enough period of time of the latest available data (e.g., the last day).

Fig. 6 shows a comparison between the performance of the model to predict sensors' measurements after two different types of concept drift occurs (sudden concept drift in Fig. 6-I, and incremental concept drift in Fig. 6-II) and its performance after updating it with the latest available data (Fig. 6-III and Fig. 6-IV respectively). Moreover, the figure also shows a comparison between the obtained anomaly-scores when predicting the sensors' measurements, before and after updating the model, for each type of concept drift (sudden concept drift in Fig. 6-V and incremental concept drift in Fig. 6-VI). It can be observed that at first, in both cases, the model performs well and its performance starts to degrade when the concept drift has occurred. However, once the concept drift has been detected and the model has been updated (the period compressed between the vertical gray dashed-lines), the performance of the model improves, whereas when the model has not been updated, it remains degraded.

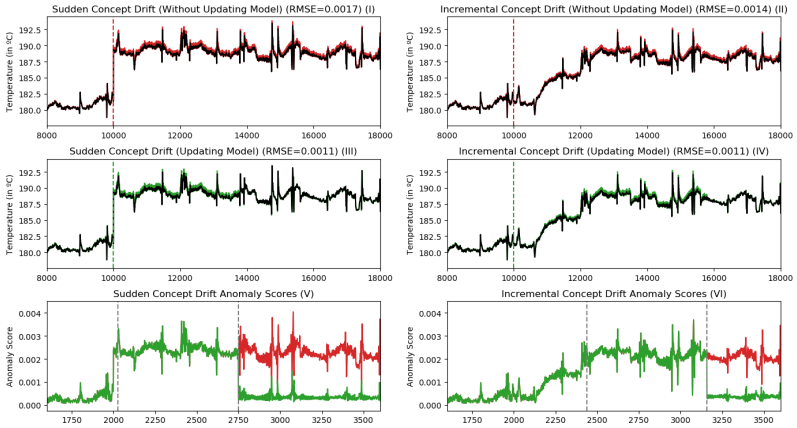

Fig. 6 Time series Forecasting with Different Types of Concept Drift and Obtained Anomaly-score

5 Predictive Analysis of Industrial Sensor Time-Series Data

Smart Manufacturing leverages the tremendous advances in Big Data analytics to improve existing analysis capabilities and provide new ones, such as predictive analytics that analyze current and historical data to make predictions about future unknown events [31]. In this regard, the built forecaster allows to obtain future values of the sensors for a data-driven predictive analysis [30], by using models (i.e., analyzers) capable to detect events in the forecasted data. In this work, in particular, three different analyzers have been built for predicting three different types of alarms (see Section 3.2). Each analyzer is composed by a classifier that determines whether in a given time-series sub-sequence, an alarm will be triggered or not. More precisely, the analyzers are trained to detect negative and positive classes (regarding the activations of the alarms) in a binary classification problem. Thus, each analyzer is specifically used to detect a concrete type of alarm, which seems reasonable for the most critical alarms since specialized classifiers usually outperform general purpose ones.

However, these are not the unique alarms that can be produced in the considered scenario, and building a specific analyzer for each alarm that can be triggered could result tedious. Moreover, considering that all the possible situations which could occur in non-stationary environments (such as Smart Manufacturing scenarios with dynamically changing industrial processes), are included in the training and testing datasets is too optimistic and could leverage to wrong predictions when dealing with unexpected circumstances. Taking into account that situation, the proposed system should be able to detect the alarms for which it has been trained, but also capable to deal properly with new conditions unseen when building it. Next sections show, first, the followed steps to prepare the data for the analyzers; then, the different analyzers built together with their performance evaluation, and finally, how the proposed system deals with new situations not considered *a priori* when building it.

5.1 Analyzers Data Preparation

For the classification data, the sliding window approach has been used to obtain different sub-sequences of the time series with a window-length of 100 (100 steps). In each sub-sequence, if an alarm has been triggered, the sub-sequence has been labelled as *True* and as *False* otherwise. This approach transforms the pre-processed dataset (see Section 3.1) into a dataset composed of time-series sub-sequences with measurements of each sensor and the sub-sequence label (i.e., a dataset with a structure of (*num_ sub-sequences* × *window-length* × (*num_ sensors* + *label*)). The sub-sequences of sensors measurements serve as input for the model (*num_ sub-sequences* × *window-length* × *num_ sensors*)), whereas the label (one-hot encoded), serves as output for the model. Notice that while the number of steps predicted by the forecaster is between 5 and 15, a longer sub-sequence has been selected for the analyzers (100 steps), due to the requirements of the models. In order to make accurate predictions of the alarms, not only are necessary the last measurements of the sensors, but also historic information about the data. Thus, in the *deployment phase*, the predicted values by the forecaster for each time horizon are appended to the last observations of the data (e.g., for the time horizon of 5 minutes, the 5 predicted values by the forecaster are appended to the last 95 observations obtained by the sensor).

5.2 Analyzers

Three different analyzers have been built for predicting three different types of alarms (see Section 3.2). Each analyzer is composed by a classifier that determines if in a given time-series sub-sequence, an alarm will be triggered or not. When building the analyzers, for simple alarms, such as the first two types of alarms presented in this work (*Plastic Temperature not Reached in the Die Entry Alarm* and *Incorrect Temperature Alarm*), a rule-based classifier based on the alarm activation condition (described in Table 1) could be used to predict correctly all the alarms. However, for more complex alarms, like the third alarm presented in this work (*Molten Resistor or Broken Thermocouple Cable in Die Zone 2 Alarm*), for which it is unknown when the activation condition will be fulfilled and therefore, cannot be modeled with rules, an automatic learning-based approach (such as neural networks-based approaches) is necessary. In this work, three Residual Neural Networks-based classifiers have been built, one to predict this last type of alarms, and another two to predict the first two types of alarms (in order to test also the performance of these classifiers when predicting simple alarms).

Different neural networks-based classifiers have been widely used for different time series classification tasks. The selection of the most suitable classifier depends on the type of time-series data and the classification task itself. In order to select an appropriate classifier for an scenario with different types of time series, coming from multiple sensors of heterogeneous nature, and for different purposes (detect different types of alarms), the benchmarks presented in [53] and [20] have been considered. Those benchmarks test up to nine distinct classifiers based on neural networks over 44 and 85 time-series databases (respectively) of different nature. In the benchmark presented in [53] the classifier based on Fully Convolutional Networks (FCN classifier in [53]) achieves the best performance for classifying the

time series databases in the benchmark, followed by the Residual Networks [15] based classifier (ResNet classifier in [53]). Nevertheless, in the benchmark presented in [20] with an extended dataset, that includes a larger number of time series databases, the ResNet classifier achieves the best performance. For that reason the selected classifier for building the analyzers, has been the *ResNet* classifier proposed in [20]. Moreover, in those benchmarks, this classifier has been demonstrated to perform well on diverse time series datasets of different nature, an interesting property for Smart Manufacturing scenarios where multiple heterogeneous sensors produce different types of time series.

The architecture of the ResNet classifier is composed by three residual blocks, followed by a global average pooling layer and a fully-connected layer with the *softmax* activation function. Each residual block is composed of three convolutions whose output is added to the residual block's input and then fed to the next layer. The number of filters for all convolutions in each block is fixed to 64, 128 and 128 respectively, with the *ReLU* activation function that is preceded by a batch normalization operation. In each residual block, the filter's length is set to 8, 5 and 3 respectively for the first, second and third convolution (see [20] for the concrete implementation).

In order to test the suitability of the ResNet classifier when it comes to alarm prediction, for each analyzer 5 different classifiers have been built by using different partitions of the training dataset (*5-fold cross-validation*). Each classifier has been trained with a batch size of 64 samples and during 400 epochs. The selected classifier has been the best one based on the loss function (*binary cross-entropy*) using the *Adam* optimizer with a learning rate of 0.001. To overcome the class imbalance, characteristic of alarm prediction and fault diagnosis scenarios, a balanced class weighting has been used during the training of the models. Then, the built *analyzers* are presented, and Section 5.3 shows their performance on predicting the three different types of alarms.

5.2.1 Melting Temperatures Analyzer

The first analyzer has been built to predict the activation of the *Plastic Temperature not Reached in the Die Entry* alarm using the predicted measurements of the sensor that measures the plastics' *melting temperature*. This one is the simplest alarm to predict, since it can be modeled with a rule based on the activation condition which states that when the melting temperature is lower than 170 °C, the alarm is activated. In this case, it seems that the model manages to learn [16] the activation condition of the alarm which can be reflected on the high performance achieved when predicting the alarms (an AUC-ROC value greater than 0.98%).

5.2.2 Incorrect Temperatures Analyzer

A second analyzer has been built to predict the *Incorrect Temperature* alarm. The prediction of this alarm is more complicated than the first one, because this alarm is triggered if in any of the extruder zones measuring the temperature of the plastic, the measured temperature is higher or lower than the established one, plus or minus an error margin (respectively). Nevertheless, the prediction of this alarm could be also modeled with a more complex rule that checks the activation condition for every temperature zone in the extruder. In this case, the analyzer

also shows a good performance predicting the alarms, so that it seems to have learnt the under-laying condition of the activation of the alarm (an AUC-ROC value greater than 0.98%).

5.2.3 Die Zone 2 Analyzer

A third analyzer has been built to predict the *Molten Resistor or Broken Thermocouple Cable in Die Zone 2* alarm. The prediction of this alarm is even more complicated than the previous one, because although there is an activation condition described by the domain experts (see Table 1), it is unknown when the activation condition will be fulfilled and trigger the alarm, and thus, the detection of this alarm cannot be modeled using rules. Therefore, in this work a residual neural networks-based classifier has been used to build an analyzer that attempts to learn the underlying pattern of the sub-sequences on which the alarm has been triggered, so that if it sees a similar pattern in the predicted measurements, it will anticipate the activation of the alarm and the operators could stop the machine in a safe way or turn on the fans that cold down the resistor. This case reinforces the utility of using a more sophisticated classifier for predicting confidently the activation of this kind of alarms that cannot be modeled with simple rules (an AUC-ROC value greater than 0.93%).

5.3 Analyzers Performance Results

As mentioned in the previous section, for building the analyzers, Residual Neural Networks (ResNet) based classifiers have been used. In order to test the suitability of these classifiers for predicting the alarms, a 5-fold cross-validation process has been followed, on which the initial training dataset has been split into five new partitions of the data into train-test datasets. For each partition of the data, five classifiers have been trained in order to test their performance for predicting the different types of alarms (totaling 25 classifiers for each analyzer). The performance of each classifier, in order to discriminate normal operation sub-sequences from sub-sequences on which an alarm has been activated, has been evaluated by using the area under the Relative Operating Characteristic (ROC) curve (AUC-ROC) metric, a performance measurement for classification problems at various thresholds settings. This metric represents a degree or a measure of separability between the classes by telling how accurate is the model distinguishing between classes (the higher is the AUC-ROC, the better is the model at predicting normal sub-sequences as normal sub-sequences and sub-sequences with alarm activations as sub-sequences with alarm activations). Table 6 summarizes the AUC-ROC value obtained by each analyzer when predicting the alarms on the cross-validation process (see the average of the AUC-ROC value for all the built classifiers on the Cross-Validation column).

Once tested the suitability of the classifiers for predicting the alarms with the cross-validation process, five new classifiers have been built for each analyzer by using all the training dataset. These classifiers have been evaluated using the evaluation dataset, and the average AUC-ROC of the five classifiers has been used to evaluate the performance of each analyzer (see the performance on the Validation column from Table 6). As Table 6 reflects, the more complicated is the

prediction of an alarm, the lower is the performance of the analyzer. The first two analyzers manage to learn to discriminate normal operation sub-sequences from sub-sequences on which an alarm has been activated, and thus, they predict correctly almost all the samples. The third analyzer is the one that shows more difficulties to learn to difference between both types of sub-sequences. Nevertheless, it shows a great performance considering the difficulty of the type of alarm to predict.

Finally, among those classifiers, the one with the highest AUC-ROC (for each analyzer) has been the selected one for predicting the alarms in the system. Fig. 7 plots the ROC curve, showing the True Positive Rate (TPR) against the False Positive Rate (FPR) of the selected classifier for each analyzer, for the evaluation dataset (see also Table 8 in Section 6).

Table 6 Time Series Analyzers Evaluation Results

	Cross-Validation		Validation	
Analyzer	**TRAIN**	**TEST**	**TRAIN**	**EVAL**
Melting Temperatures	0.9978	0.9848	0.9998	0.9992
Incorrect Temperatures	0.9980	0.9817	1	0.9904
Die Zone 2	0.9849	0.9530	0.9464	0.9375

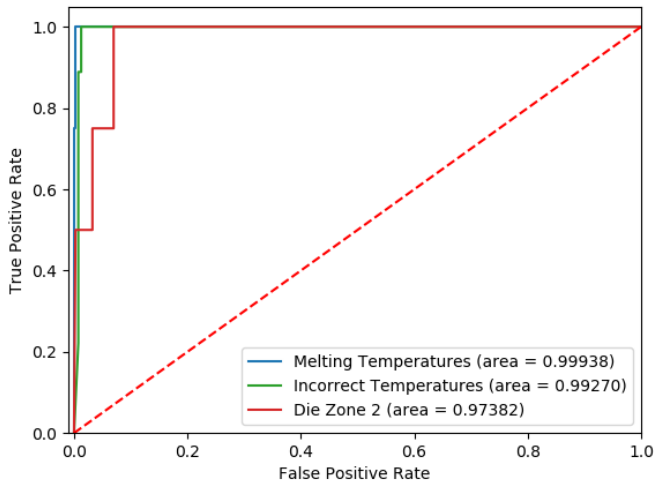

Fig. 7 Area Under ROC Curve for each Analyzer for the Evaluation Dataset

5.4 Dealing with Unknown Situations in Smart Manufacturing Scenarios

In the same way as traditional recognition and classification algorithms, the deep learning-based classifiers used in previous sections usually work under a common closed set (or static environment), where the training and testing data are drawn from the same label and feature spaces. However, this assumption is rather restrictive, given that in real-world recognition/classification tasks it is usually hard to collect training samples representing all the possible situations. Therefore, a more realistic scenario is usually open and non-stationary, due to the fact that unseen situations can emerge unexpectedly (as occurs in smart manufacturing scenarios with dynamically changing processes), which could drastically weaken the robustness of these traditional methods [12]. Taking into account this aspect, in recent years, several approaches have appeared to deal with these situations including among others zero/one-shot learning and open set recognition. A categorization of them can be found in [12], where the authors put particular interest in open set recognition for being able to deal with unknown situations, like those mentioned before.

Open set recognition [38] describes such a scenario where new classes (unseen during the training) could appear in the testing and requires the classifiers not only to classify the known classes accurately, but also, to deal effectively with unknown classes [12]. Therefore, in open set recognition problems, classifiers usually consider a reject option that allows them to refuse to recognize an input sample due to its low confidence, avoiding to classify unknown samples as other classes. For example, in the context of this work, the analyzers manage to detect confidently the alarms for which they have been trained on, however, if the raw materials in the production process change requiring a higher/lower temperature to be melted, the *Melting Temperatures Analyzer* could detect an incorrect temperature in the new samples collected and trigger an alarm continuously. Detecting these situations would allow to update the models so that they can deal with the new situations while also avoiding incorrect classifications.

Open set recognition is a recent research topic to which researchers are devoting many efforts and therefore, some open set recognition algorithms have been developed to extend both traditional machine learning methods and deep neural networks-based models (a review of them can be found in [12]). Regarding the deep neural networks-based models considered in this work, although they were not initially conceived for open set recognition problems, some works have already attempted to extend this models for open set recognition tasks [14], [3]. One of the most popular one is the method proposed in [3] where a new model layer, *OpenMax*, is introduced to estimate the probability of an input sample being from an unknown class.

OpenMax has already been used by other authors in the domain of computer vision [3] and natural language processing [42] for image and text classification/recognition tasks (respectively). However, to the best of the authors' of this paper knowledge it has not been already used in the context of industrial time-series data classification. In this work, a first attempt has been done to adapt a neural network for open set time series classification purposes by changing the *softmax* activation layer of the used *ResNet* classifiers to use the *openmax* layer as

proposed in [3][9]. However, it is worth mentioning that open set recognition is an open research topic that still faces serious challenges on which there is a lack of well known frameworks and algorithms [12]. Therefore, although a proof of concept has been conducted to test the potential of open set recognition methods for detecting alarms, further research is out of the scope of this work.

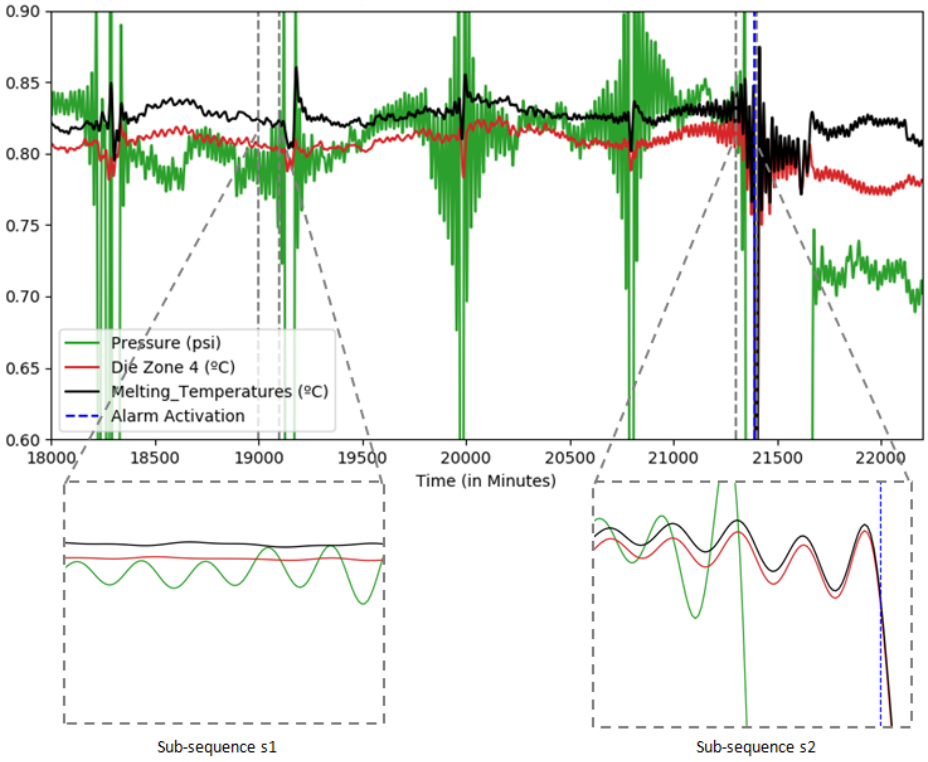

Fig. 8 Example of Open set Recognition with Different Time Series

As a proof of concept, the *softmax* activation layer of the *Melting Temperatures Analyzer* has been changed to use the *openmax* activation layer. The performance of the classifier with both activation layers has been tested with sub-sequences of three different time series to predict whether an alarm will be activated or not in those sub-sequences. Concretely, the *Melting Temperatures* time series (those with which the classifier has been trained to detect the *Melting Temperature not Reached in the Die Entry Alarm* activations); the *Die Zone 4 Temperature* time series (which is a highly correlated time series to the *Melting Temperatures* time series since the sensors measuring the temperatures are placed close to each other in the extruder machine); and the *Melting Pressure* time series (which has been

[9] Code and data for the research paper "Towards Open Set Deep Networks" [3] `https://github.com/abhijitbendale/OSDN` and an example of its implementation with Keras `https://github.com/aadeshnpn/OSDN`.

captured by a different nature sensor) have been considered for the test. Fig. 8 shows a segment of the considered time series and two example sub-sequences: a normal operation mode sub-sequence (*s1*) and a sub-sequence on which an alarm has been activated (*s2*).

The predictions made by the classifier with each activation layer for those sub-sequences are shown in Table 7. It can be noticed that both layers classify properly the sub-sequences of the *Melting Temperatures* time series, predicting that an alarm will be triggered in the sub-sequence *s2* and the contrary case for the sub-sequence *s1*. The same predictions are obtained for the *Die Zone 4 Temperatures* time series which seems reasonable since the sensor capturing these series is close to the sensor measuring the melting temperatures and therefore, both time series are really similar and highly correlated. Finally, when dealing with a different nature time series, the *softmax* layer predicts that a *Plastic Temperature not Reached in the Die Entry Alarm* will be triggered in both sub-sequences which is not correct since it is unknown whether in those sub-sequences an alarm would be activated or not. In this case, the *openmax* layer is able to reject to classify the sample sub-sequence and thus, it predicts it as *unknown*.

Table 7 Open set Recognition Example Results

		Activation Layer					
			Softmax			Openmax	
Time Series	Label	Class	Probability	Predicted	Class	Probability	Predicted
Melting Temperatures S1	No-Alarm	No-Alarm	9.9994e-01	No-Alarm	No-Alarm	9.4576e-01	No-Alarm
		Alarm	5.0694e-05		Alarm	9.3959e-04	
		Unknown	-		Unknown	5.3294e-02	
Die Zone 4 Temperatures S1	-	No-Alarm	9.9985e-01	No-Alarm	No-Alarm	0.9285	No-Alarm
		Alarm	1.4477e-04		Alarm	0.00189	
		Unknown	-		Unknown	0.06953	
Melting Pressure S1	-	No-Alarm	1.0282e-20	Alarm	No-Alarm	3.3574e-13	Unknown
		Alarm	≈ 1.000e+00		Alarm	2.7945e-01	
		Unknown	-		Unknown	7.2054e-01	
Melting Temperatures S2	Alarm	No-Alarm	0.0065	Alarm	No-Alarm	0.0230	Alarm
		Alarm	0.9934		Alarm	0.5123	
		Unknown	-		Unknown	0.4646	
Die Zone 4 Temperatures S2	-	No-Alarm	0.0032	Alarm	No-Alarm	0.0144	Alarm
		Alarm	0.9967		Alarm	0.5249	
		Unknown	-		Unknown	0.4606	
Melting Pressure S2	-	No-Alarm	2.3415e-09	Alarm	No-Alarm	2.1567e-06	Unknown
		Alarm	≈ 1.000e+00		Alarm	4.9905e-01	
		Unknown	-		Unknown	5.0094e-01	

6 Alarm Prediction System Performance Evaluation

This section shows first, the performance of the proposed alarm prediction system as a whole, to predict the three different types of alarms by using the built analyzers over the forecasted measurements by the built forecaster; and then, an example of the prediction of an alarm on a real use case.

6.1 System performance

With the objective of testing a real use case in a real scenario, the evaluation dataset (unseen data for the built models) has been used to evaluate the system performance. First, the measurements of each sensor in the evaluation dataset

have been predicted by using the built forecaster. Iteratively, each sub-sequence of 5 measurements in the evaluation dataset (10 and 15 respectively for the time horizons of 10 and 15 minutes) has been replaced with the predicted values by the forecaster for the previous 300 observations. Then, the predicted values have been transformed by using the same approach described in Section 5.1 for the *classification data* (the sub-sequences of predicted values have been labelled with the corresponding true labels from the evaluation dataset). Finally, those values have been used to evaluate the analyzers.

The performance of the analyzers, to discriminate normal operation sub-sequences from sub-sequences on which an alarm has been activated, has been evaluated by using the area under Relative Operating Characteristic (ROC) curve (AUC-ROC) metric. Table 8 summarizes the AUC-ROC value obtained by each analyzer when predicting the alarms on the forecasted values for each time horizon for the evaluation dataset compared to the AUC-ROC value obtained when predicting alarms over the original evaluation dataset.

Table 8 Time Series Analyzers Evaluation Results over the Forecasted Data (AUC ROC)

	Raw Data	**Forecasted (steps-ahead)**		
Analyzer	**EVAL**	**5**	**10**	**15**
Melting Temperatures	0.99937	0.99937	0.99937	0.99812
Incorrect Temperatures	0.99270	0.99270	0.99270	0.99270
Die Zone 2	0.97381	0.97381	0.97381	0.97381

6.2 Alarm Prediction Example

Fig. 9 shows the materialization of the proposed system in a real use case. In the figure, it can be seen on the one hand, a 3D model of an extruder machine on which an alarm activation has been predicted with a time horizon of 15 minutes. Taking into account that the alarm has been predicted 15 minutes ahead (the less critical time horizon), the extruder zones corresponding to the sensors associated to that alarm are colored in yellow, indicating that something could be wrong (see the warning icon and the extruder part highlighted in yellow). Furthermore, in the chart displayed under the extruder it can be seen in black the last measurements of the extruder (real measurements), whereas in red, orange and yellow the predicted measurements of the extruder with a time horizon of 5, 10 and 15 minutes respectively (the vertical red line indicates the activation of the real alarm).

7 Conclusions and Future Work

This paper presents an alarm prediction system that applies deep learning techniques to predict the activation of diverse types of alarms that could serve for different purposes such as the predictive maintenance of the equipment or production and product quality optimization. The system follows a two-stage *forecaster-analyzer* approach on which first a LSTM-RNN based forecaster predicts the future

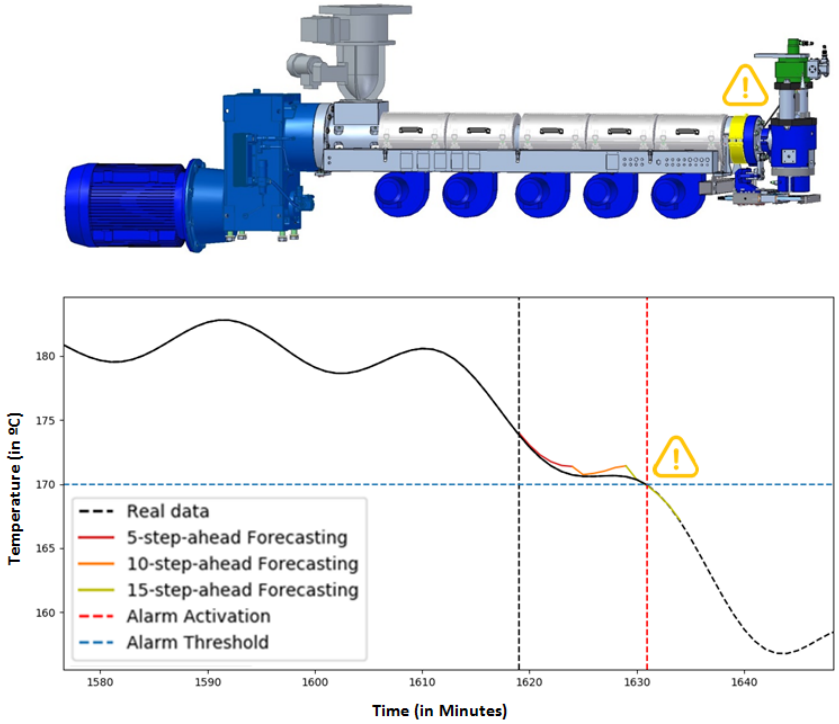

Fig. 9 Example of an Alarm Prediction using the Proposed System

measurements of the sensors and then, distinct analyzers based on Residual Neural Networks determine whether the predicted measurements will trigger an alarm or not. It has been tested with real time-series data coming from a real production plant, on which it has shown a great performance to predict diverse types of alarms in three different time horizons (5, 10 and 15 minutes).

Regarding the forecaster, different models, including classical time series forecasting models and ANN-based models, have been tested and compared for the selection of an adequate forecasting model. Based on the obtained results, a time series forecaster has been built to predict the future measurements of 11 different sensors implanted in an extruder machine by using LSTM neural networks. The built forecaster achieves a great performance when predicting the future measurements of the multiple sensors with an average RMSE of 0.00852, 0.01215 and 0.01737 (respectively for each time horizon). Moreover, the suitability of the built model for smart manufacturing scenarios with dynamically changing processes, and its ability to adapt to non-stationary environments on which concept drift could occur has also been tested, under the perspective of system applicability in smart manufacturing scenarios.

With respect to the analyzers, three different analyzers have been built using residual neural networks to detect if in a time-series sub-sequence an alarm will be triggered or not. The residual neural networks used, learn, example by example, which kind of time series patterns are associated with the alarms activations, so that when a new pattern matching those ones is seen, the alarm activation is

predicted. Each analyzer has been specialized in the detection of a particular type of alarm. The built analyzers have shown a great performance in predicting the three different types of alarms with an AUC-ROC value of 0.99937, 0.99270 and 0.97381 respectively. Thus, the built analyzers have demonstrated that residual neural networks are able to detect efficiently, not only alarms that can be modeled with simple rules based on the activation condition, but also more complex alarms on which it is unknown when the activation condition will be fulfilled, and thus, cannot be modeled by rules. That behaviour reinforces the interest of using pattern matching-based analyzers. Moreover, as shown in this work, the ability of these models to deal also with unknown situations not seen before could be really interesting in Smart Manufacturing scenarios. Further research in this line would involve not only detecting unknown situations, but also detecting new types of alarms for those cases (i.e., novelty discovery [34]).

Concerning the followed two-stage approach, it requires building and training two different models instead of using a single model to predict the alarms in a straightforward way. However, using different models for forecasting the future measurements of the sensors and for detecting alarms in the predicted measurements gives the system more modularity and makes it more flexible to changes and extensible to different predictive analysis tasks, since the predicted measurements of the sensors could be used for other analysis processes. Therefore, further research would involve building more analyzers for different purposes, such as anomaly detection, and also, in order to predict more types of alarms, such as alarms that are triggered with a really low frequency (by using different techniques, such as Generative Adversial Networks [40], to generate synthetic time series on which an alarm has been activated for training the models).

Finally, from what it comes to its real applicability, the system supports some features that make it particularly suitable for Smart Manufacturing scenarios: on the one hand, the forecaster is able to predict the future measurements of different types of time-series data captured by various sensors in non-stationary environments. On the other hand, the analyzers are able to detect alarms that can be modeled with simple rules based on the activation condition, and also more complex alarms on which it is unknown when the activation condition will be fulfilled. In these regard, recent research works are advocating for online stream analytics [36] and novelty discovery in data streams [9], which could be particularly interesting for such scenarios with dynamically changing processes and thus, further research would involve the adoption of some of these technologies into the system.

Acknowledgements This work is supported by the Spanish Ministry of Economy and Competitiveness (MEC) under Grant No.: FEDER/TIN2016-78011-C4-2-R. The work of Kevin Villalobos is funded by the Basque Government under Grant No.: PRE_2018_2_0263. Johan Suykens acknowledges support by ERC Advanced Grant E-DUALITY (787960), KU Leuven C14/18/068, FWO project GOA4917N and Flemish Government Onderzoeksprogramma Artificiële Intelligentie Vlaanderen programme.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In:

Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation, OSDI'16, p. 265–283. USENIX Association, USA (2016)
2. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: D.B. Lomet (ed.) Foundations of Data Organization and Algorithms, pp. 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg (1993)
3. Bendale, A., Boult, T.E.: Towards open set deep networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1563–1572 (2016)
4. Boyes, H., Hallaq, B., Cunningham, J., Watson, T.: The industrial internet of things (iiot): An analysis framework. Computers in Industry **101**, 1 – 12 (2018). DOI https://doi.org/10.1016/j.compind.2018.04.015. URL http://www.sciencedirect.com/science/article/pii/S0166361517307285
5. Cai, S., Palazoglu, A., Zhang, L., Hu, J.: Process alarm prediction using deep learning and word embedding methods. ISA Transactions **85**, 274 – 283 (2019). DOI https://doi.org/10.1016/j.isatra.2018.10.032. URL http://www.sciencedirect.com/science/article/pii/S0019057818304105
6. Chollet, F., et al.: Keras. https://keras.io/ (2015). Accessed: 2019-06-24
7. Choudhary, A.K., Harding, J.A., Tiwari, M.K.: Data mining in manufacturing: a review based on the kind of knowledge. Journal of Intelligent Manufacturing **20**(5), 501 (2009). DOI https://doi.org/10.1007/s10845-008-0145-x. URL https://link.springer.com/article/10.1007/s10845-008-0145-x
8. Davis, J., Edgar, T., Porter, J., Bernaden, J., Sarli, M.: Smart manufacturing, manufacturing intelligence and demand-dynamic performance. Computers & Chemical Engineering **47**, 145 – 156 (2012). DOI https://doi.org/10.1016/j.compchemeng.2012.06.037. URL http://www.sciencedirect.com/science/article/pii/S0098135412002219. FOCAPO 2012
9. Faria, E.R., Gonçalves, I.J.C.R., de Carvalho, A.C.P.L.F., Gama, J.: Novelty detection in data streams. Artificial Intelligence Review **45**(2), 235–269 (2016). DOI 10.1007/s10462-015-9444-8. URL https://doi.org/10.1007/s10462-015-9444-8
10. Gama, J.a., Žliobaitundefined, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (2014). DOI 10.1145/2523813. URL https://doi.org/10.1145/2523813
11. García, V., Sánchez, J.S., Rodríguez-Picón, L.A., Méndez-González, L.C., Ochoa-Domínguez, H.d.J.: Using regression models for predicting the product quality in a tubing extrusion process. Journal of Intelligent Manufacturing (2018). DOI 10.1007/s10845-018-1418-7. URL https://doi.org/10.1007/s10845-018-1418-7
12. Geng, C., Huang, S.j., Chen, S.: Recent advances in open set recognition: A survey. arXiv preprint arXiv:1811.08581 (2018)
13. Google Inc.: Google artificial intelligence platform. https://cloud.google.com/ai-platform/ (2019). Accessed: 2020-01-21
14. Hassen, M., Chan, P.K.: Learning a neural-network-based representation for open set recognition. arXiv preprint arXiv:1802.04365 (2018)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016). DOI 10.1109/CVPR.2016.90
16. Hinton, G.E.: How neural networks learn from experience. Scientific American **267**(3), 144–151 (1992). URL http://www.jstor.org/stable/24939221
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997). DOI 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735
18. Horelu, A., Leordeanu, C., Apostol, E., Huru, D., Mocanu, M., Cristea, V.: Forecasting techniques for time series from sensor data. In: 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp. 261–264 (2015). DOI 10.1109/SYNASC.2015.49
19. Iqbal, R., Maniak, T., Doctor, F., Karyotis, C.: Fault detection and isolation in industrial processes using deep learning approaches. IEEE Transactions on Industrial Informatics **15**(5), 3077–3084 (2019). DOI 10.1109/TII.2019.2902274
20. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019). DOI 10.1007/s10618-019-00619-1. URL https://doi.org/10.1007/s10618-019-00619-1

21. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems **3**(3), 263–286 (2001). DOI 10.1007/PL00011669. URL https://doi.org/10.1007/PL00011669
22. Khandelwal, I., Adhikari, R., Verma, G.: Time series forecasting using hybrid arima and ann models based on dwt decomposition. Procedia Computer Science **48**, 173 – 179 (2015). DOI https://doi.org/10.1016/j.procs.2015.04.167. URL http://www.sciencedirect.com/science/article/pii/S1877050915006766. International Conference on Computer, Communication and Convergence (ICCC 2015)
23. Koushik, J.: Understanding convolutional neural networks. arXiv preprint arXiv:1605.09081 (2016)
24. Langone, R., Alzate, C., Bey-Temsamani, A., Suykens, J.A.K.: Alarm prediction in industrial machines using autoregressive ls-svm models. In: 2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 359–364 (2014). DOI 10.1109/CIDM.2014.7008690
25. Li, H., Qian, B., Parikh, D., Hampapur, A.: Alarm prediction in large-scale sensor networks — a case study in railroad. In: 2013 IEEE International Conference on Big Data, pp. 7–14 (2013). DOI 10.1109/BigData.2013.6691771
26. Li, L., Ota, K., Dong, M.: Deep learning for smart industry: Efficient manufacture inspection system with fog computing. IEEE Transactions on Industrial Informatics **14**(10), 4665–4673 (2018). DOI 10.1109/TII.2018.2842821
27. Lin, J., Keogh, E., Lonardi, S., Chiu, B.: A symbolic representation of time series, with implications for streaming algorithms. In: Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD '03, p. 2–11. Association for Computing Machinery, New York, NY, USA (2003). DOI 10.1145/882082.882086. URL https://doi.org/10.1145/882082.882086
28. Lütkepohl, H.: Vector Autoregressive Models, pp. 1645–1647. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). DOI 10.1007/978-3-642-04898-2_609. URL https://doi.org/10.1007/978-3-642-04898-2_609
29. Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G.: Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder (2016)
30. Menezes, B.C., Kelly, J.D., Leal, A.G., Roux, G.C.L.: Predictive, prescriptive and detective analytics for smart manufacturing in the information age. IFAC-PapersOnLine **52**(1), 568 – 573 (2019). DOI https://doi.org/10.1016/j.ifacol.2019.06.123. URL http://www.sciencedirect.com/science/article/pii/S2405896319302095. 12th IFAC Symposium on Dynamics and Control of Process Systems, including Biosystems DYCOPS 2019
31. Moyne, J., Iskandar, J.: Big data analytics for smart manufacturing: Case studies in semiconductor manufacturing. Processes **5**(3), 39 (2017)
32. Olah, C.: Understanding lstm networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (2015). Accessed: 2019-06-24
33. Palpanas, T., Beckmann, V.: Report on the first and second interdisciplinary time series analysis workshop (itisa). SIGMOD Rec. **48**(3), 36–40 (2019). DOI 10.1145/3377391.3377400. URL https://doi.org/10.1145/3377391.3377400
34. Pimentel, M.A., Clifton, D.A., Clifton, L., Tarassenko, L.: A review of novelty detection. Signal Processing **99**, 215 – 249 (2014). DOI https://doi.org/10.1016/j.sigpro.2013.12.026. URL http://www.sciencedirect.com/science/article/pii/S016516841300515X
35. Sadouk, L.: Cnn approaches for time series classification. In: Convolutional Neural Network. IntechOpen (2018)
36. Sahoo, D., Pham, Q., Lu, J., Hoi, S.C.: Online deep learning: Learning deep neural networks on the fly. arXiv preprint arXiv:1711.03705 (2017)
37. Saurav, S., Malhotra, P., TV, V., Gugulothu, N., Vig, L., Agarwal, P., Shroff, G.: Online anomaly detection with concept drift adaptation using recurrent neural networks. In: Proceedings of the ACM India Joint International Conference on Data Science and Management of Data, CoDS-COMAD '18, p. 78–87. Association for Computing Machinery, New York, NY, USA (2018). DOI 10.1145/3152494.3152501. URL https://doi.org/10.1145/3152494.3152501
38. Scheirer, W.J., de Rezende Rocha, A., Sapkota, A., Boult, T.E.: Toward open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(7), 1757–1772 (2013). DOI 10.1109/TPAMI.2012.256
39. Selvin, S., Vinayakumar, R., Gopalakrishnan, E.A., Menon, V.K., Soman, K.P.: Stock price prediction using lstm, rnn and cnn-sliding window model. In: 2017 International

Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1643–1647 (2017). DOI 10.1109/ICACCI.2017.8126078
40. Shao, S., Wang, P., Yan, R.: Generative adversarial networks for data augmentation in machine fault diagnosis. Computers in Industry **106**, 85 – 93 (2019). DOI https://doi.org/10.1016/j.compind.2019.01.001. URL http://www.sciencedirect.com/science/article/pii/S0166361518305657
41. Shcherbakov, M.V., Brebels, A., Shcherbakova, N.L., Tyukov, A.P., Janovsky, T.A., Kamaev, V.A.: A survey of forecast error measures. World Applied Sciences Journal **24**(24), 171–176 (2013)
42. Shu, L., Xu, H., Liu, B.: Doc: Deep open classification of text documents. arXiv preprint arXiv:1709.08716 (2017)
43. Siami-Namini, S., Namin, A.S.: Forecasting economics and financial time series: Arima vs. lstm. arXiv preprint arXiv:1803.06386 (2018)
44. Smith, T.G., et al.: pmdarima: Arima estimators for python (2017). URL http://www.alkaline-ml.com/pmdarima. [Online; accessed 21-01-2019]
45. Taieb, S.B., Bontempi, G., Atiya, A.F., Sorjamaa, A.: A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. Expert Systems with Applications **39**(8), 7067 – 7083 (2012). DOI https://doi.org/10.1016/j.eswa.2012.01.039. URL http://www.sciencedirect.com/science/article/pii/S0957417412000528
46. Tao, F., Qi, Q., Liu, A., Kusiak, A.: Data-driven smart manufacturing. Journal of Manufacturing Systems **48**, 157 – 169 (2018). DOI https://doi.org/10.1016/j.jmsy.2018.01.006. URL http://www.sciencedirect.com/science/article/pii/S0278612518300062. Special Issue on Smart Manufacturing
47. The GPyOpt authors: Gpyopt: A bayesian optimization framework in python. http://github.com/SheffieldML/GPyOpt (2016)
48. Wan, J., Tang, S., Li, D., Wang, S., Liu, C., Abbas, H., Vasilakos, A.V.: A manufacturing big data solution for active preventive maintenance. IEEE Transactions on Industrial Informatics **13**(4), 2039–2047 (2017). DOI 10.1109/TII.2017.2670505
49. Wan, R., Mei, S., Wang, J., Liu, M., Yang, F.: Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. Electronics **8**(8), 876 (2019)
50. Wang, J., Ma, Y., Zhang, L., Gao, R.X., Wu, D.: Deep learning for smart manufacturing: Methods and applications. Journal of Manufacturing Systems **48**, 144 – 156 (2018). DOI https://doi.org/10.1016/j.jmsy.2018.01.003. URL http://www.sciencedirect.com/science/article/pii/S0278612518300037. Special Issue on Smart Manufacturing
51. Wang, J., Yang, F., Chen, T., Shah, S.L.: An overview of industrial alarm systems: Main causes for alarm overloading, research status, and open problems. IEEE Transactions on Automation Science and Engineering **13**(2), 1045–1061 (2016). DOI 10.1109/TASE.2015.2464234
52. Wang, K., Li, K., Zhou, L., Hu, Y., Cheng, Z., Liu, J., Chen, C.: Multiple convolutional neural networks for multivariate time series prediction. Neurocomputing **360**, 107 – 119 (2019). DOI https://doi.org/10.1016/j.neucom.2019.05.023. URL http://www.sciencedirect.com/science/article/pii/S092523121930685X
53. Wang, Z., Yan, W., Oates, T.: Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp. 1578–1585 (2017). DOI 10.1109/IJCNN.2017.7966039
54. Wu, Q., Ding, K., Huang, B.: Approach for fault prognosis using recurrent neural network. Journal of Intelligent Manufacturing pp. 1–13 (2018). DOI https://doi.org/10.1007/s10845-018-1428-5. URL https://link.springer.com/article/10.1007/s10845-018-1428-5
55. Yunpeng, L., Di, H., Junpeng, B., Yong, Q.: Multi-step ahead time series forecasting for different data patterns based on lstm recurrent neural network. In: 2017 14th Web Information Systems and Applications Conference (WISA), pp. 305–310 (2017). DOI 10.1109/WISA.2017.25
56. Zhang, B., Zhang, S., Li, W.: Bearing performance degradation assessment using long short-term memory recurrent network. Computers in Industry **106**, 14 – 29 (2019). DOI https://doi.org/10.1016/j.compind.2018.12.016. URL http://www.sciencedirect.com/science/article/pii/S0166361518304640
57. Zhang, G.: Time series forecasting using a hybrid arima and neural network model. Neurocomputing **50**, 159 – 175 (2003). DOI https://doi.org/10.1016/S0925-2312(01)00702-0. URL http://www.sciencedirect.com/science/article/pii/S0925231201007020

58. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting **14**(1), 35 – 62 (1998). DOI https://doi.org/10.1016/S0169-2070(97)00044-7. URL http://www.sciencedirect.com/science/article/pii/S0169207097000447
59. Zhang, S., Zhang, S., Chen, X., Wu, S.: Analysis and research of cloud computing system instance. In: 2010 Second International Conference on Future Networks, pp. 88–92 (2010). DOI 10.1109/ICFN.2010.60
60. Zhang, W., Guo, W., Liu, X., Liu, Y., Zhou, J., Li, B., Lu, Q., Yang, S.: Lstm-based analysis of industrial iot equipment. IEEE Access **6**, 23551–23560 (2018). DOI 10.1109/ACCESS.2018.2825538
61. Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics **28**(1), 162–169 (2017). DOI 10.21629/JSEE.2017.01.18
62. Zhu, J., Wang, C., Li, C., Gao, X., Zhao, J.: Dynamic alarm prediction for critical alarms using a probabilistic model. Chinese Journal of Chemical Engineering **24**(7), 881 – 885 (2016). DOI https://doi.org/10.1016/j.cjche.2016.04.017. URL http://www.sciencedirect.com/science/article/pii/S1004954116303044