Neural Computing and Applications
https://doi.org/10.1007/s00521-021-06865-z

ORIGINAL ARTICLE

Real-time detection of uncalibrated sensors using neural networks

Luis J. Muñoz-Molina[1,2] · Ignacio Cazorla-Piñar[1] · Juan P. Dominguez-Morales[3] · Luis Lafuente[2] · Fernando Perez-Peña[2]

Received: 8 October 2021 / Accepted: 12 December 2021

Abstract
Nowadays, sensors play a major role in several fields, such as science, industry and everyday technology. Therefore, the information received from the sensors must be reliable. If the sensors present any anomalies, serious problems can arise, such as publishing wrong theories in scientific papers, or causing production delays in industry. One of the most common anomalies are uncalibrations. An uncalibration occurs when the sensor is not adjusted or standardized by calibration according to a ground truth value. In this work, an online machine-learning based uncalibration detector for temperature, humidity and pressure sensors is presented. This development integrates an artificial neural network as the main component which learns from the behavior of the sensors under calibrated conditions. Then, after being trained and deployed, it detects uncalibrations once they take place. The obtained results show that the proposed system is able to detect the 100% of the presented uncalibration events, although the time response in the detection depends on the resolution of the model for the specific location, i.e., the minimum statistically significant variation in the sensor behavior that the system is able to detect. This architecture can be adapted to different contexts by applying transfer learning, such as adding new sensors or having different environments by re-training the model with minimum amount of data.

Keywords Neural networks · Sensors · Uncalibrations · Sensor anomalies · Transfer learning

1 Introduction

Nowadays, sensors play a major role in the connected world [1–4]. Science, industry and even day-to-day devices, integrate sensors to collect the information coming from the surrounding environment. Ensuring the reliability and consistency of the information collected is essential in order to guarantee the proper use of the information. The reliability of the acquired information depends not only on the features of the sensor, but also on its calibration status. An uncalibration is said to take place when the sensor is not adjusted or standardized by calibration according to a ground truth value. In general, anomalies can appear in several shapes [5], however the uncalibration event usually appears in the form of long-term drifts with different kind of responses such as linear, exponential, logarithmic or, simply, an irregular drift. Furthermore, there are cases where the magnitude to control is not directly controlled, that is, the set point is unknown. Thus, in the latter case, the detection of potential uncalibrations can be really challenging. As happens with some other kind of anomalies in sensor behaviour [6], the detection of uncalibrations is a key issue in order to ensure fundamental aspects such as reliability, safety or cost-effectiveness in crucial assets in different contexts. These include pharma, energy, aeronautics and any other industrial fields. In this paper, an architecture to track uncalibrations is proposed. Thus, it is

✉ Luis J. Muñoz-Molina
luisjesus.munozmolina@altran.com

✉ Fernando Perez-Peña
fernandoperez.pena@uca.es

Ignacio Cazorla-Piñar
ignaciocazpin@gmail.com

Juan P. Dominguez-Morales
jpdominguez@atc.us.es

Luis Lafuente
luis.molinero@uca.es

[1] Altran Innovation Center for Advance Manufacturing, Cadiz, Spain

[2] School of Engineering, Universidad de Cadiz, Cadiz, Spain

[3] Robotics and Technology of Computers Lab., Universidad de Sevilla, Sevilla, Spain

Published online: 24 January 2022

an ideal candidate to be deployed in a wide variety of contexts, including scientific, industrial and, specifically, in the context of Industry 4.0 [7–10].

In this work, an online uncalibration detector for temperature, humidity and pressure sensors under unknown conditions (set points unknown) has been developed. The major novelty of the paper is the use of an Artificial Neural Network (ANN) as an expert on the sensors behaviour, so that the estimated behaviour by the ANN can be compared to the one measured, and thus, potential uncalibrations can be detected. Different statistical techniques, such as goodness of fit and confidence intervals, are used in order to check that the estimation of the neural network can be considered as equal, or different to the obtained measures.

The developed architecture detects 100% of the uncalibrations events, and it permits flexibility and scalability. Indeed, the obtained results suggest that the system can be trained with a generic set of sensors and then be capable of working in more specific contexts with a much lower amount of data. Thus, for instance, the system could be trained with a standard set of sensors and then be used within a more specific set, devoted to a more concrete task. Furthermore, new sensors can be easily included in the system requiring significantly low amount of data. Eventually, re-training the system requires data coming from approximately one week of measurements (10,080 samples per sensor at a sampling rate of one sample per minute and per sensor).

The proposed detector has been tested in a real context in the facilities of a company of the pharmaceutical sector. Thus, it is important to note that the presented work has been developed under actual production conditions.

The rest of the paper is structured as follows: Sect. 3 introduces the materials used in this work, including the dataset (Sect. 3.2). The implemented methods are described in Sect. 4, together with the Neural Network (NN) model. Then, the results obtained are presented in Sect. 5, which are divided in two different experiments: first, the performance of a proposed NN model is evaluated (Sect. 5.1), and then its scalability and flexibility are evaluated on (Sect. 5.2). Section 6 presents the discussion and also the conclusions of this work.

2 Related works and literature review

The uncalibration detection issue can be tracked using different approaches. The available information is a key factor in order to determine which methods should be used. On the one hand, the approaches based on the use of classical techniques can be considered. This is the case of similarity-based modeling and multivariate analysis [11], or the time series analysis [12]. A survey of some other classical approaches for anomaly detection can be found in [13]. It is important to note that the techniques mentioned in the previous works were used for the detection of specific kinds of anomalies, not uncalibrations.

On the other hand, due to the huge development of Machine Learning (ML) and Artificial Intelligence (AI) during recent years, the use of techniques and approaches based on them has increased [13, 14]. An example of this fact is the case of [15], where anomalies, defined as unusual behaviour at a specific time, were detected by using Hierarchical Temporal Memory. Another interesting example can be seen in [16], where a Convolutional Neural Network (CNN) was developed in order to detect sensor failures. An example of the application of NNs for anomaly detection in power plants can be found in [17]. Finally, an overview of NN applications of fault diagnosis and detection in engineering-related systems can be found in [18]. As in the classical approaches, it is important to note that all these developments were developed to detect specific anomalies, not uncalibrations in the sensor system.

Table 1 shows a comparative study among the aforementioned works and their approaches. In this table, the method used and the capability for detecting both general faults and uncalibrations are shown.

3 Materials

This section describes the sensors used, and the dataset obtained from them.

3.1 Sensors

Three different variables were measured, namely, temperature, humidity and pressure. Temperature and humidity sensors are integrated within a single device, the Novasina nSens-HT-ENS. The humidity measurement range of this device goes from 0% RH to 100% RH, with a measurement accuracy of 0.5% RH. Regarding the temperature measurement range, it goes from -20 °C to $+80$ °C, with a measurement accuracy of 0.1 °C. The sensor used to measure the pressure is the Novasina Pascal-ST-ZB. It is a differential pressure sensor based on static pressure measurement. Regarding its measurement range, it goes from -50 Pa to 50 Pa, with a measurement accuracy of $<0.5\%$ fs (full scale).

3.2 Dataset

The dataset used to obtain the results presented in this work consists of temperature, humidity and pressure values coming from the sensors installed across different clean rooms devoted to drug production. Rooms are located on

Table 1 Comparative study between different state-of-the-art works related to fault detection and uncalibrations detection

Reference	Method	Fault Detection	Uncalibration
[11]	Similarity analysis and SVM	Yes	No
[12]	Time series analysis	Yes	No
[15]	Hierarchical temporal memory	Yes	No
[16]	Convolutional neural networks	Yes	No
[17]	MLP RNN and probabilistic	Yes	No
Our proposal	MLP decoder	Yes	Yes

two different floors of a building: the second floor and the basement. More precisely, the data coming from the second floor is described in Table 2. This dataset contains data collected during a whole year in different rooms of the second floor. Each room includes only one sensor per category (one temperature/humidity sensor and one pressure sensor). Thus, sensor redundancy is not available. The sensors used are described in Sect. 3.1. The data were collected at a sample per minute rate during the aforementioned period of time.

A smaller dataset containing data from rooms located in the basement was also collected. This data are shown in Table 2. As opposed to the second floor, the basement dataset is smaller due to the difficulties accessing the data generated within this floor. This dataset contains, approximately, one hundred samples per sensor, coming from a total of eleven sensors per category. Furthermore, samples are not equally distributed. These samples were collected during different periods of the year.

Since the dataset obtained from the second floor is more complete, it was used as the main dataset. Thus, experiments in Sects. 5.1 and 5.2 were conducted taking this dataset for training purposes. However, in Sect. 5.2, the dataset coming from the basement was used for testing the adaptation of the system proposed to a different environment.

Regarding the dataset slicing, two approaches were defined depending on the kind of experiments that were performed. On the one hand, in Sect. 5.1, 60% of the samples of the data coming from the second floor were used as the training set for each type of sensor, 10% were used as the validation set, and the remaining 30% were used as the testing set. On the other hand, in Sect. 5.2, 70% of the samples were used for training the main NN, 10% as the testing set and 20% for the application of the Transfer Learning. In this last case, not only data coming from the second floor was used, but also data coming from the basement was also included, in order to test the transfer learning application from one floor to another.

The use of the different slices is more detailed in Sect. 4.

Table 2 Dataset summary. Note that −1 floor refers to the basement

Floor	Sensor type	Number of sensors	Number of samples
2	Temperature	17	8,935,200
2	Humidity	17	8,935,200
2	Pressure	24	12,614,400
−1	Temperature	11	1,100,000
−1	Humidity	11	1,100,000
−1	Pressure	11	1,100,000

4 Methodology

In this section, the architecture of the system implemented, which is capable of detecting sensor uncalibration, is presented. The whole system has been developed using Python. Different libraries such as numpy, scipy and pandas have been used for the implementation of the filtering processes. For the implementation and training of the model, TensorFlow 2 [19] together with the Keras [20] functional API have been used.

The approach is based on ML techniques, and it follows the block diagram shown in Fig. 1. The architecture is based on an ANN which, during the training phase, learns how the sensors behave. Then, in the inference stage, the ANN is able to predict if sensors are becoming uncalibrated. The way data is processed is summarized in the following steps:

- The data are collected by the sensors. It is important to note that it comes from only one type of sensor, either temperature, humidity or pressure. This means that the network that processes this data is sensor-dependant.
- The data are filtered. Here, different outlier detection and pre-processing techniques are applied.
- The mean of all measured values in a specific time is computed.
- The computed mean value is fed to the ANN.
- The ANN infers the set of measurements that produced the input mean value.
- The inferred set of measures is compared with the real set of measurements. If the discrepancy exceeds the confidence interval, then it is considered a rejection.

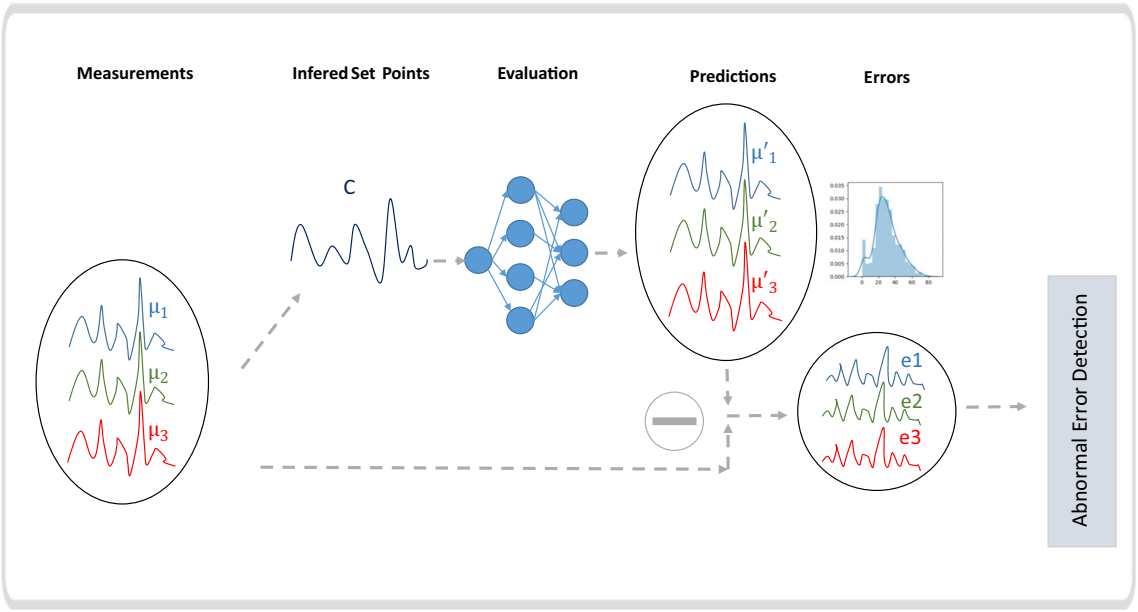

Fig. 1 Block diagram of the approach presented. Five main stages can be observed: the measurement stage, the estimation of the set point, the evaluation of the estimated set point by the NN and the generation of the prediction and, finally, the computation of the error with respect to the real value

- The rejection density is computed for a fixed time window. When rejection density achieves values that exceed a specific threshold, an uncalibration takes place.

4.1 Preprocessing

The data used to both train and test the network have to be pre-processed. This preprocessing, using filtering techniques, is a standard procedure to detect anomalous values. These values could yield undesired effects on the performance of the algorithm if they are not properly detected and treated. Thus, this preprocessing phase can be divided into three stages: a threshold-based filtering, the Mahalanobis distance [21] and the Savitzky-Golay filter [22].

The threshold-based filtering is based on the thresholds defined for the warning and alarm systems in the rooms. This alarm system is designed to detect extreme values in the incoming measurements. Thus, potential sudden failures in the sensor systems can be detected. It has been observed that these extreme values add noise to the input signal and, therefore, worsen the NN learning. These thresholds were used during this research as a way to discard all of those extreme values that were already detected by the system and which could have a deep impact in the NN learning process. Thus, a measured value is dropped whenever one of the aforementioned thresholds is exceeded. In this way, only calibrated and non-extreme uncalibrated values remain in the signal.

Next, a filter based on the Mahalanobis distance [23, 24] is applied. Mahalanobis distance is an outlier detection method specifically designed for dealing with multidimensional distributions. More precisely, this stage focuses on determining whether an observation including information from the whole set of sensors (i.e., a vector with as many entries as sensors) can be considered as an outlier. Thus, in this stage, non-extreme values whose relative occurrence is low would be classified as outliers.

Finally, the third stage is focused on reducing the noise in the incoming signal. Different kinds of filters such as Butterworth or wavelet filters could have been applied in this stage. However, the incoming signal presents a wide variety of different behaviors along the data collection period. Thus, a more flexible noise filtering process is required. The Savitzky-Golay's filter [22], which computes local polynomial regressions providing new values for each sample, is used. This filtering process preserves essential properties of the signal, such as maximum and minimum values or trends, while noise is efficiently dropped out.

After the data obtained from the sensors is preprocessed, the mean value of the measurements, at a time, of all the sensors involved is used as the input to an ANN.

4.2 Neural network

The proposed ANN model is a Multi Layer Perceptron (MLP) that consists of an input layer with one neuron, five hidden layers with three hundred neurons each and an output layer with as many output neurons as sensors. The

ReLu function was used as activation function in the hidden layers. In the case of the output layer, the Linear function was chosen. The model was trained during 30 epochs using the Adamax optimizer. In order to determine the optimal number of layers and neurons per layer, Scikit-Learn's Grid Search algorithm (GridSearchCV) [25] algorithm was used. During the training stage, validation sets were used in order to control over-fitting. These same hyperparameters were also used for the rest of experiments performed within this work, as in the case of Transfer Learning. As it will be seen later, Mean Squared Error and Mean Absolute Error has been used as loss functions. As it will be seen later in this document, the chosen loss function has a deep impact on the statistical behaviour of the prediction errors. After training the network with the mean value of the pre-processed data from the sensors, the network is able to decompose the input stimulus into the predicted value for each sensor. This is the expected behavior that each sensor should have, based on that input stimuli.

4.3 Error computation

This is the final step of the proposed approach. In this stage, the difference between the real measured value obtained from the sensors and the ones predicted is computed, evaluated and classified as normal or anomalous. This evaluation is based on the work of [15] where Hierarchical Temporal Memory [26] was used to predict the next measurement. That prediction was then compared to the real measurement and residuals were computed.

In our approach, goodness of fit tests were applied on the errors obtained during the training stage. Thus, the underlying distribution of the error could be used to compute specific confidence intervals with the desired signification for each sensor. On the one hand, the absolute mean error of the sensors measurements properly fits a normal distribution. On the other hand, the mean squared error fits an exponential distribution.

Then, confidence intervals were generated by using the normal distribution associated to the error of each sensor. This choice was based on the fact that the absolute mean error was approximately equal to the uncalibration that was taking place. These confidence intervals are associated to each sensor, and they represent the behavior of each sensor from the point of view of the system. They are known as the resolution of the system per sensor.

Finally, a rejection takes place whenever the value exceeds the bounds of the confidence interval. The density of rejections is computed as the rolling ratio between the number of rejections within a window of a fixed time length (with 1,440 minutes as default value, which corresponds to the number of minutes in a day) and the length of the window.

The rejection density is the variable that triggers the uncalibration warning. An uncalibration is said to take place whenever a threshold is reached during a specified amount of time.

The proposed architecture is not able to determine if a sensor is calibrated or not. The expected behaviour is that the NN detects small differences between the current measurement and the one from which it learnt. Thus, in the eventuality of a maintenance task in any sensor, the architecture proposed might detect this new event as a potential uncalibration. This uncalibration event will remain active until this new condition of the sensor is re-learnt as 'calibrated' by the model.

The standard procedure to overcome this issue is to re-train the model under these new conditions. Thus, this process would require a very high temporal cost, since collecting data during almost a whole year would be necessary. For this reason, the use of transfer learning represents a feasible solution. A small amount of data is needed to teach the new condition to the NN by using transfer learning.

4.4 Transfer learning

Transfer learning [27, 28] is a ML technique aimed at specializing ML models with a minimum amount of data. This technique consists of two main steps. The first step is training a model for a more general task conceptually related with the target or specialized task. For instance, the target task could be to identify a specific face in a picture and, in that case, the general task would be training a model for general face identification. The second step would be to re-train a subset of layers of the model with a training dataset made out of elements corresponding to the specialized task (a specific face in the example presented).

In the case of this work, multiple scenarios can benefit from transfer learning. These scenarios can be summarized as follows: changing the ground truth value for all sensors or for a subset of them, including new sensors and adapting to some other environments with, possibly, insufficient information available.

As it was presented in Sect. 3.2, a different slicing of the dataset was used for transfer learning. Transfer learning is applied by dividing the original dataset into three smaller subsets, namely, a training subset for the original model, which will be called model A, a re-training subset for the application of transfer learning (model A will be renamed to model B after this process) and, finally, a testing subset for both models. Regarding the training subset, it contains 70% of the datapoints. This stage corresponds to the training for the general task. Regarding the subset devoted

to be used for re-training the model and, thus, for the application of transfer learning, it contains the 20% of the datapoints. The amount of required datapoints can be reduced to up to 2.5% of the total (10,000 datapoints per sensor) in the case of temperature and humidity. Finally, the testing subset includes 10% of the datapoints. This subset was used to test the performance of the neural network, both before and after the application of the transfer learning. A constant offset was added in the last part of the test set in order to simulate the recalibration due to a maintenance task. It is important to note that this offset is not applied to the whole test set in order to properly see the change of conditions and the associated evaluation by both, model A and model B.

As mentioned, one of the main purposes of the application of transfer learning is the reduced amount of necessary samples for training a model. This amount is directly associated to the number of parameters to retrain, which, at the same time, is closely related to the number of layers to retrain. The nature of the target task, i.e., how abstract it is, and where (in which layers) the concepts learnt are located within the NN, are crucial factors which will determine the amount of required data. During the current research, transfer learning has been applied on the very last hidden layer of the NN, i.e., only the weights connecting the last hidden layer with the output layer has been retrained. The obtained results suggest that training this subset of weights was enough for the existing requirements.

5 Results

This section shows the results obtained using the methods presented in Sect. 4. Two different experiments were conducted: (1) to evaluate the capability for the detection of uncalibrations of the current approach, through performance evaluation experiments; (2) to test the flexibility and scalability of the system proposed, through transfer learning experiments.

5.1 Performance evaluation experiments

These experiments were conducted to test the detection of uncalibrated sensors using the architecture presented. The dataset used is the one shown in Sect. 3.2. Uncalibrations were introduced in the testing set as drifts of different nature (e.g linear, exponential or logarithmic) across time and in different temporal frames (for instance, at the beginning, in the middle and at the end of the year).

An uncalibration is considered to be detected once the rejection density value has reached a predefined threshold, which depends on the sensor, and whose mean value is 0.8. Futhermore, this threshold has to be reached during a predefined period of time, which also depends on the sensor, and can be defined as two weeks, since it is enough time for all sensors, and it is a reasonably low period of time for uncalibration detection. An example of the application of this technique can be seen in Fig. 2 where rejection density for both the upper and lower bound of the confidence interval are shown.

The second and third plots show the rejection density values. It can be seen that these values stay low whenever a uncalibration is not taking place. Conversely, when a uncalibration takes place, the rejection density reaches values close to one.

Regarding the performance of the approach, uncalibrations were detected for all different types of sensors. However, two different scenarios can be distinguished, namely, a first optimistic scenario, where the uncalibration is detected within the tolerance ranges defined by the quality requirements of the pharma industry (i.e., $\pm 0.5\,^{\circ}\mathrm{C}$, $\pm 3\%$ and $\pm 0.5\mathrm{P}$ in temperature, humidity and pressure, respectively), and a non-optimistic scenario, where the uncalibration is detected over the tolerance range. For all those sensors that are in the optimistic scenario, an estimation of the remaining time until the tolerance range gets exceeded can be provided.

Thus, for the case of the temperature sensors, the proposed approach was tested on the available data coming from temperature sensors over 17 rooms. All uncalibrations were detected for all sensors. In the best scenario, the architecture detected uncalibrations associated to deviations of 0.25 °C (tolerance 0.5 °C). This accuracy was reached for 16 rooms out of 17 of the second floor. In the worst scenario, the uncalibration was detected for deviations of, approximately, 0.5 °C.

For the humidity sensors case, the proposed method was tested on the available data coming from humidity sensors over 17 clean rooms. All uncalibrations were detected for all sensors. In the best scenario, the algorithm detects uncalibrations associated to deviations of 2% (tolerance 3%). This accuracy was reached for 14 rooms out of 17 of the second floor. In the worst scenario, the uncalibration was detected for deviations close to 3%.

Finally, for the pressure sensors case, the proposed method was tested on the available data coming from over 24 pressure sensors. All uncalibrations were detected for all sensors. In the best scenario, the algorithm detected uncalibrations associated to deviations under 0.5 Pascals (tolerance 0.5 Pascals). This accuracy was reached for 6 out of 20 rooms of the second floor. In the worst scenario, the uncalibration was detected for deviations close to 1.5 Pascals.

Some other ANN architectures were tested during the development of this research. The same experiments were

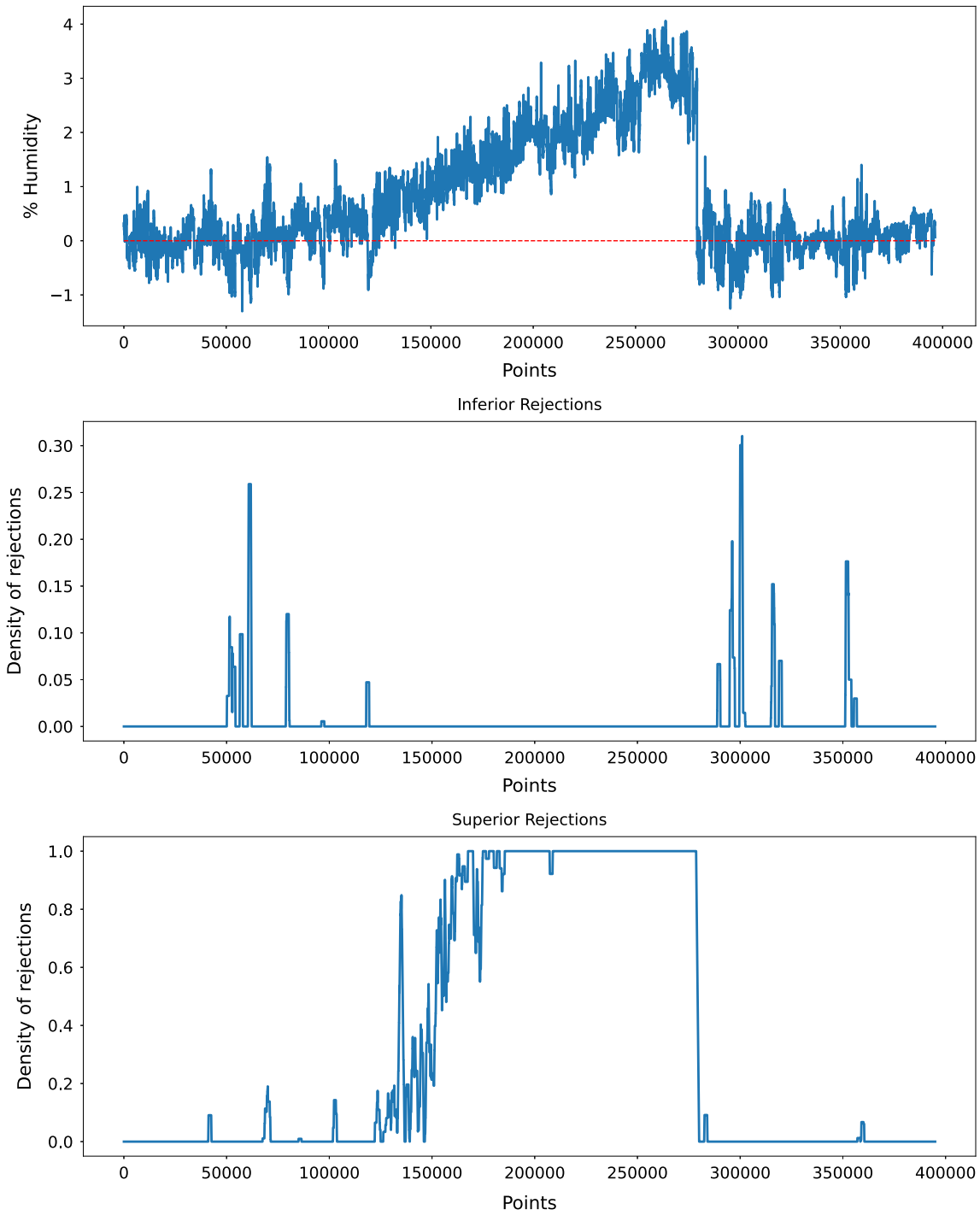

Fig. 2 Uncalibration detection through rejection density. In the upper subplot, the blue trace shows the error corresponding to a linear uncalibration, and the red dashed line stands for the error zero value. In the middle subplot, the value of the rejection density for error values exceeding the lower bounds of the confidence interval is shown. As it can be seen, the maximum value for this rejection is, approximately, 0.3. In the lower subplot, the value of the rejection density for error values exceeding the upper bounds of the confidence interval is shown. As it can be seen, this density reaches the maximum possible value from a critical point and during the whole uncalibration phenomenon

conducted using a Wide & Deep Neural Network for temperature and humidity sensors and Recursive Neural Networks (RNN) for the pressure sensors. The architecture of the Wide & Deep Neural Network was based on the MLP used on the implemented approach, but also includes an input layer per sensor, so that the precise information

coming separately from each sensor can be included. This can be seen in Fig. 3.

Wide and Deep architecture yielded better results for most of the rooms. However, for two of them, the uncalibrations were not detected. It has been observed that, for these two rooms, how the sensor is supposed to behave under a specific condition is not learnt by the neural network, and the identity function is approximated instead. Thus, when an anomaly of any kind was introduced on the input signal, this same value was retrieved by the NN and, therefore, the uncalibration was not detected. It has been observed that the worst results were obtained from these two rooms independently of the architecture. This fact suggests that the local conditions inside these rooms had a deep impact on the learning capabilities of the different models.

Regarding the RNN, one single Long-Short Term Memory (LSTM) layer was used, containing 164 units and the ReLu activation function. As output layer, a Dense layer with as many units as sensors was used. In this case, the Linear function was used as activation function. The model was trained by using the Adamax optimizer. In this case, uncalibrations were not detected either. This was probably caused due to the long-term nature of uncalibrations. Uncalibrations are long term phenomenons, thus, the deviation associated to an uncalibration will be observed during long periods of time. Hence, the deviation will be included as relevant by LSTM neurons during the information inclusion stage, that is, when the input gate evaluates what information is relevant and what information is not. In this case, the learning of this long-term phenomenons by the NN provoke the worst results.

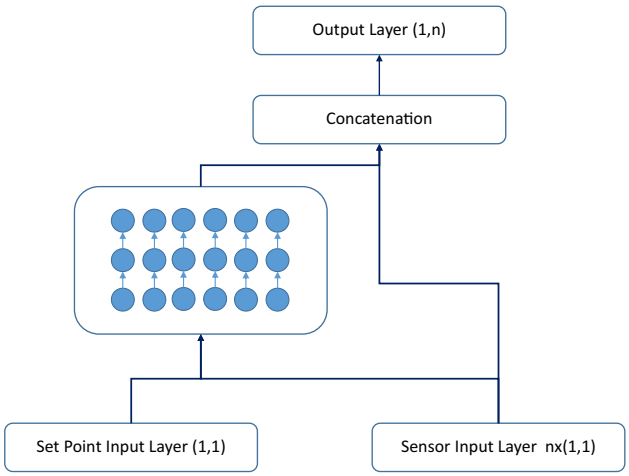

Fig. 3 Wide and Deep model diagram. In this figure, the tested architecture of the wide and deep model is shown. It can be seen $n + 1$ different input layers; an input layer with one single neuron, which is connected to a hidden block of five hidden layers with three hundred neurons each. This input layer takes the estimated global set point as input. Next, n input layers with one single neuron each. Each input layer takes the measurements coming from the different associated sensors as input. Finally, these input layers are concatenated to the output of the hidden block and connected to the output layer, which has a neuron per sensor

5.2 Transfer learning experiments

These experiments were conducted to test the capability of the architecture for learning a new condition with the less temporal cost. Thus, these experiments were split in three categories: change in global or specific conditions, adding new sensors and adapting to some other environments with insufficient information available. To evaluate the transfer learning technique, the error obtained for the main model (model A) and the retrained model (model B) under the specific new condition were compared.

5.2.1 Change of conditions

These experiments were conducted to test the capability of the architecture proposed for learning a new condition as the current calibration status for one or more sensors. In the case of the individual offset, in Fig. 4 two different phases of the error obtained from model A are shown. The first one corresponds to the model evaluation under normal conditions. The second one corresponds to the model evaluation under the generation of a specific offset. It can be seen how the error is displaced, approximately, three units which is the same amount of the generated offset.

However, in model B, it can be seen how the error returns to the same range of values as previously. This behaviour suggests that the expected results are achieved. This experiment was reproduced for different rooms and the different types of sensors, obtaining similar results. It can be observed in Fig. 5.

It should be noted that the minimum number of datapoints to properly use transfer learning is around ten thousand samples. This number of samples corresponds, approximately, to one week of data collection.

Regarding the generation of an offset for the whole set of sensors, the model did not detect any change in its behavior. It suggests that, if the behaviour of the whole set of sensors is equally changed, then, no effect on the joint behaviour can be detected. Thus, no uncalibration is taking place from the point of view of the model. This fact proposes an interesting question regarding how the joint behaviour could be learnt.

5.2.2 Adding new sensors to the model

These experiments were conducted to determine the time that the architecture takes to learn and include the

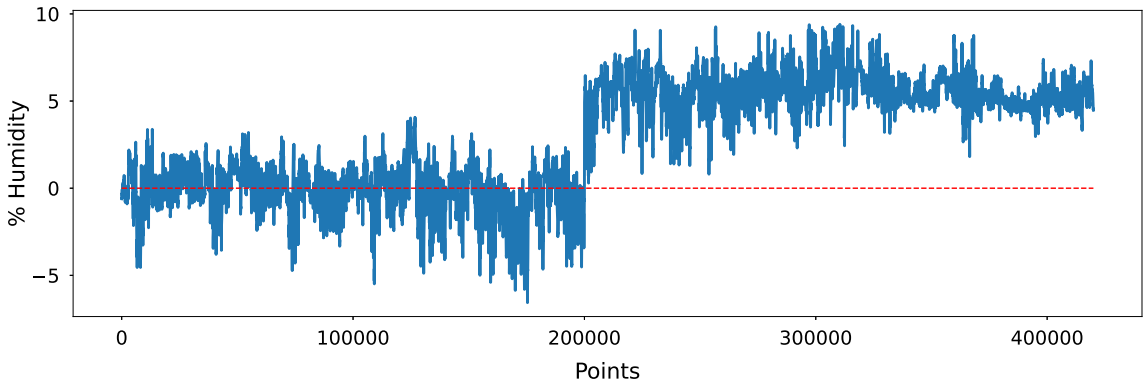

Fig. 4 In this figure, the error associated to a scheduled recalibration can be seen. The blue trace shows the error, and the red dashed line stands for the error zero value. The recalibration takes place at the mid point of the figure. Since transfer learning is not applied, once the recalibration takes place, the error immediately increases

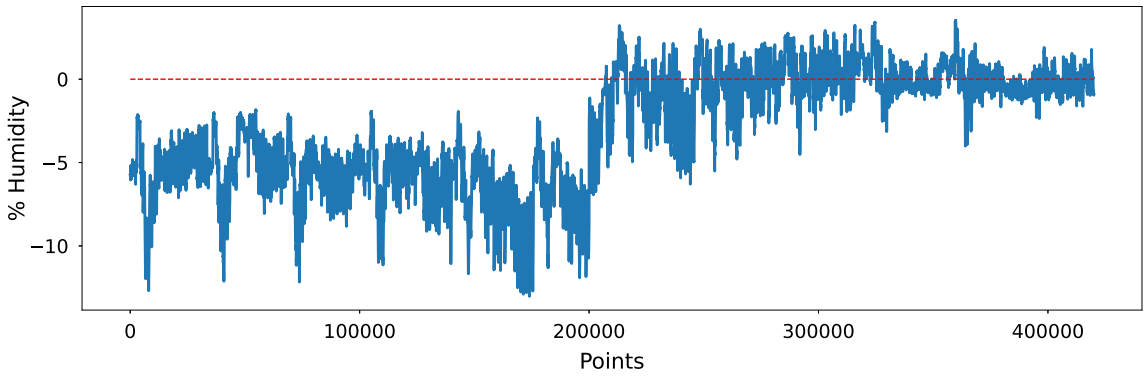

Fig. 5 In this figure, the blue trace shows the error associated to a scheduled recalibration, and the red dashed line stands for the error zero value. In this case, transfer learning is applied, thus, once the recalibration takes place, the error immediately decreases. It is important to note that this model is not applied in the previous moments to the recalibration task

behaviour of new sensors in the loop. These tests were made using the humidity and temperature sensors.

A first dataset with data coming from 13 sensors out of 17 was taken as main set. For this set of sensors, $o_1 = 322,677$ datapoints were available. In this case, model A was trained with this dataset. Then, model B was re-trained with data coming from both the initial set of 13 sensors and a set of 4 new sensors. The obtained results show that uncalibrations on the new set of sensors can be properly detected by model B. Figure 6 shows how uncalibrations are detected in the new sensors that were included for the temperature case.

5.2.3 Adapting to some other environment with insufficient information available

These experiments were conducted to test if the architecture could be trained with data coming from one specific location and then be implemented on a different location after a transfer learning process.

In this case, model A was trained with data coming from the second floor with 17 sensors of both types, temperature and humidity. Then, model B was re-trained with a small amount of data coming from the basement. Approximately, 80,000 datapoints were used. The obtained results show that uncalibrations are detected by model B in this new environment. Furthermore, the problem of the lack of information was solved by applying this method. Figure 7 shows an example of how uncalibrations were detected by model B in this specific context.

The obtained results shows that transfer learning is a valid technique for the adaptation of the system to some other locations.

Regarding the performance of the architecture proposed, uncalibrations can be detected even for values smaller than the defined tolerance ranges for most cases in temperature and humidity sensors. A summary of the obtained resolutions and the corresponding tolerances per sensor type are shown in Table 3.

The application of transfer learning shows the flexibility and scalability of the architecture, enabling the use of the model in a wide variety of contexts such as sensor addition, integration within new environments and partially solving problems associated with the lack of information available.

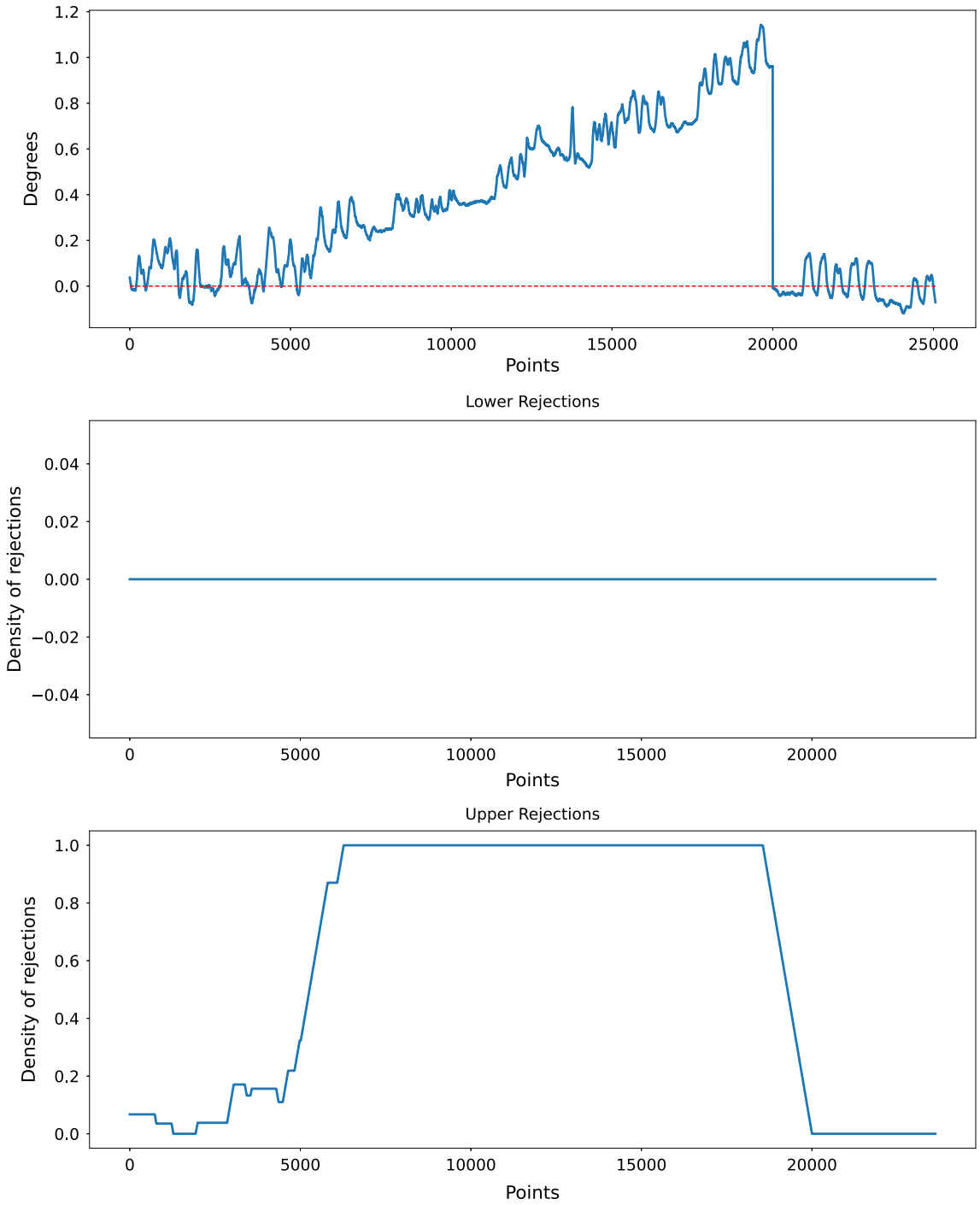

Fig. 6 Uncalibration detection for new included sensors. This figure shows the results of the application of transfer learning in order to include new sensors within the architecture. The detection of an uncalibration for one of those sensors is shown. In the upper subplot,the error corresponding to a linear uncalibration can be seen. The red dashed line stands for the error zero value. In the middle subplot, the value of the rejection density for error values exceeding the lower bounds of the confidence interval is presented. In this case, the maximum value for this rejection is approximately zero. In the lower subplot, the value of the rejection density for error values exceeding the upper bounds of the confidence interval is presented. As it can be seen, this density reaches the maximum possible value from a critical point and during the whole uncalibration phenomenon

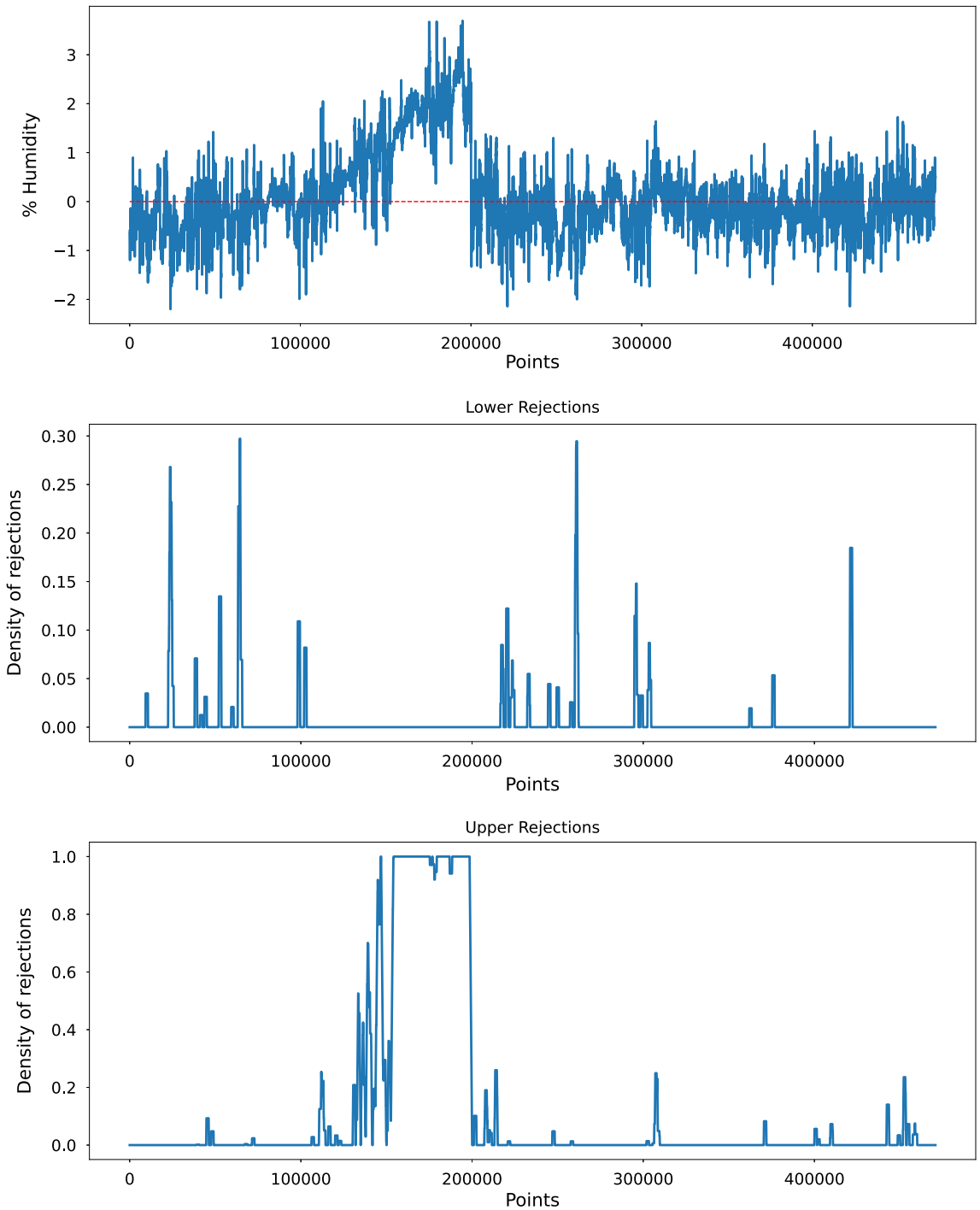

Fig. 7 Uncalibration detection in the basement. In this figure, the detection of an uncalibration taking place in the basement is shown. In this case, transfer learning has been applied in order to re-train a model with a small quantity of data coming from the basement. The model was previously trained with data coming from the second floor. In the upper subplot, the error corresponding to a linear uncalibration is shown in blue. The red dashed line stands for the error zero value. In the middle subplot, the value of the rejection density for error values exceeding the lower bounds of the confidence interval is shown. In this case, the maximum value for this rejection is 0.3. In the lower subplot, the value of the rejection density for error values exceeding the upper bounds of the confidence interval is shown. As it can be seen, this density reaches the maximum value starting from a critical point and remains during the whole uncalibration phenomenon

Table 3 Resolution of the different systems

Sensor type	Min	Mean	Max	Tolerance
Temperature	0.10 °C	0.26 °C	0.64 °C	0.5 °C
Humidity	0.49 %	1.17 %	3.11 %	3 %
Pressure	0.23 Pa	1.77 Pa	6.68 Pa	0.5 Pa

6 Discussion

It is proven that transfer learning retrieved good results for very low amounts of data. The minimum amount of required data is about 10,000 samples, which is the data obtained from one week of observations at a sampling rate of one sample per minute. The fact that only the last weight matrix was needed to re-train means that the information associated to all these modifications were learned on a very abstract level in the NN [29]. This suggest, on the one hand, that offsets have a low impact on the joint dynamics of sensors. Thus, the joint behavior of the whole set of sensors is ruled by much deeper relationships than the addition of specific values. On the other hand, the results obtained for pressure sensors, despite they are not under a common constant condition, may be explained by this fact. The sudden changes in pressure values could be understood as an offset, what implies that the model has a deep knowledge of the sensor behavior, although these offsets reduce the accuracy of the detection of uncalibrations.

The architecture presented has been trained and tested in a real context, with data coming directly from a production pharma factory. Besides, due to the flexible capabilities associated to sensor additions and re-training, the system can be seen as a resilient system as defined in [30]. Furthermore, the architecture presented in this work is currently deployed in Azure [31], where the system performs the uncalibration detection directly from the information of the sensors from a digital twin of the building. This digital twin contains the current status of the physical sensors.

7 Conclusions and future works

In this work, a system with the capability for detecting uncalibrations in real time has been presented. This system was able to detect all the presented uncalibration events (100% accuracy). In most cases, these uncalibration events could be detected before the specified tolerances were exceeded. Furthermore, this solution can be easily retrained to be adapted to a variety of different scenarios, such as new environmental conditions, the integration of new devices in the sensor network and the deployment in new places never seen before by the system. This adaptability is achieved by means of Transfer Learning.

To the best of our knowledge, this is the first time that potential uncalibrations of a set of sensors are online detected whenever the set point is unknown. Furthermore, the proposed architecture can be extended by means of transfer learning to a wide range of different fields.

Regarding the future work, different objectives are considered: (1) extending the results obtained after the application of transfer learning. Training the architecture with a generic set of sensors of one type (such as temperature sensors) and then, testing the performance on sensor systems of a different kind, for instance, engine, smart-city and power plants sensors systems, among others; (2) obtaining information about the learning of the solution through the application of Explainable Artificial Intelligence (XAI) [32]. This could provide useful information about the potential scalability and flexibility of the solution and the different models to which it could be applied.

Acknowledgements The authors would like to thank the Spanish Agencia Estatal de Investigación (AEI) for supporting this work

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work was partially supported by the Altran Innovation Center for Advance Manufacturing, the EU H2020 project Virtual IoT Maintenance System (VIMS Grant ID: 878757), the Spanish grant (with support from the European Regional Development Fund) MIND-ROB (PID2019-105556GB-C33) and by the EU H2020 project CHIST-ERA SMALL (PCI2019-111841-2).

Declarations

Conflict of interest The authors have no conflict of interest to declare that are relevant to the content of this article.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Lee SK, Bae M, Kim H (2017) Future of iot networks: a survey. Appl Sci 1072:7–10
2. Lammel G (2015) The future of mems sensors in our connected world. In: 2015 28th IEEE international conference on micro electro mechanical systems (MEMS), pp. 61–64 . https://doi.org/10.1109/MEMSYS.2015.7050886
3. Alharbi N, Soh B (2019) Roles and challenges of network sensors in smart cities. IOP Conf Ser Earth Environ Sci 322:012002. https://doi.org/10.1088/1755-1315/322/1/012002
4. Singh M, Sachan S, Singh A, Singh KK (2020) Internet of Things in pharma industry: possibilities and challenges. Academic Press, London, pp 195–216
5. Sharma A, Golubchik L, Govindan R (2010) Sensor faults detection methods and prevalence in real-world datasets. ACM Trans Sens Netw 6(3):1–39
6. Tobar FA, Yacher L, Paredes R, Orchard ME (2011) Anomaly detection in power generation plants using similarity-based modeling and multivariate analysis. In: proceedings of the 2011 American control conference, IEEE, pp. 1940–1945
7. Ayvaz S, Alpay K (2021) Predictive maintenance system for production lines in manufacturing: a machine learning approach using iot data in real-time. Exp Syst Appl 173:114598
8. Dogan A, Birant D (2020) Machine learning and data mining in manufacturing. Exp Syst Appl 166:114060
9. De Aguiar ASP, de Oliveira MAR, Pedrosa EF, dos Santos FBN (2021) A camera to lidar calibration approach through the optimization of atomic transformations. Exp Syst Appl 176:114894
10. Belmonte-Fernandez O, Montoliu R, Torres-Sospedra J, Sansano-Sansano E, Chia-Aguilar D (2018) A radiosity-based method to avoid calibration for indoor positioning systems. Exp Syst Appl 105:89–101
11. Lamrini B, Gjini A, Daudin S, Pratmarty P, Armando F, Travé-Massuyès L (2018) Anomaly detection using similarity-based one-class svm for network traffic characterization. In: DX@ Safeprocess (2018)
12. Feremans L, Vercruyssen V, Cule B, Meert W, Goethals B (2019) Pattern-based anomaly detection in mixed-type time series. In: joint European conference on machine learning and knowledge discovery in databases, pp. 240–256, Springer
13. Muruti G, Rahim FA, bin Ibrahim Z (2018) A survey on anomalies detection techniques and measurement methods. In: 2018 IEEE conference on application, information and network security (AINS), pp. 81–86 . https://doi.org/10.1109/AINS.2018.8631436
14. Chalapathy R, Chawla S (2019) Deep learning for anomaly detection: a survey. CoRR **abs/1901.03407**arXiv:1901.03407
15. Ahmad S, Lavin A, Purdy S, Agha Z (2017) Unsupervised real-time anomaly detection for streaming data. Neurocomputing 262:134–147
16. Liu J, Zhang M, Wang H, Zhao W, Liu Y (2019) Sensor fault detection and diagnosis method for ahu using 1-d cnn and clustering analysis. Comput Intell Neurosci. https://doi.org/10.1155/2019/5367217
17. Banjanovic-Mehmedovic L, Hajdarevic A, Kantardzic M, Mehmedovic F, Dzananovic I (2017) Neural network-based data-driven modelling of anomaly detection in thermal power plant. Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije 58(1), 69–79
18. Amiruddin AAAM, Zabiri H, Taqvi SAA, Tufa LD (2020) Neural network applications in fault diagnosis and detection: an overview of implementations in engineering-related systems. Neural Comput Appl 32(2):447–472
19. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. Software available from tensorflow.org . https://www.tensorflow.org/
20. Ketkar N (2017) Introduction to keras. In: Deep learning with python, pp. 97–111 . Springer
21. De Maesschalck R, Jouan-Rimbaud D, Massart DL (2000) The mahalanobis distance. Chemomet Intell Lab Syst 50(1):1–18
22. Acharya D, Rani A, Agarwal S, Singh V (2016) Application of adaptive savitzky-golay filter for eeg signal processing. Perspect Sci 8:677–679
23. Ghorbani H (2019) Mahalanobis distance and its application for detecting multivariate outliers. Facta Univ Ser Math Inform 34:583–95
24. Leys C, Klein O, Dominicy Y, Ley C (2018) Detecting multivariate outliers: use a robust variant of the mahalanobis distance. J Exp Soc Psychol 74:150–156
25. Pontes FJ, Amorim G, Balestrassi PP, Paiva A, Ferreira JR (2016) Design of experiments and focused grid search for neural network parameter optimization. Neurocomputing 186:22–34
26. Hawkins J, Blakeslee S (2004) On intelligence. Macmillan, United Kingdom
27. Bozinovski S (2020) Reminder of the first paper on transfer learning in neural networks, 1976. Informatica. https://doi.org/10.31449/inf.v44i3.2828
28. Pan SJ, Yang Q (2009) A survey on transfer learning. IEEE Trans knowl Data Eng 22(10):1345–1359
29. Géron A (2019) Hands-on machine learning with scikit-learn, keras, and tensorflow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media
30. Zhang W-J, Lin Y (2010) On the principle of design of resilient systems-application to enterprise information systems. Enterprise Inf Syst 4(2):99–110
31. Chappell D et al (2010) Introducing the windows azure platform. David Chappell & Associates White Paper
32. Gunning D (2017) Explainable artificial intelligence (XAI). Defense advanced research projects agency (DARPA), nd Web **2**(2)