Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

Virtual sensors for erroneous data repair in manufacturing *a machine learning pipeline*

Sagar Sen [a], Erik Johannes Husom [a], Arda Goknil [a,*], Dimitra Politaki [b], Simeon Tverdal [a], Phu Nguyen [a], Nicolas Jourdan [c]

[a] *SINTEF Digital, Forskningsvegen 1, Oslo, Norway*
[b] *INLECOM Innovation, Tatoiou 11, Kifisia 145 61, Greece*
[c] *TU Darmstadt, Otto-Berndt-Straße 2 64287 Darmstadt, Germany*

ARTICLE INFO

Keywords:
Machine learning pipeline
Deep learning
Manufacturing
Virtual sensors
Data quality
Erroneous data repair

ABSTRACT

Manufacturing converts raw materials into finished products using machine tools for controlled material removal or deposition. It can be observed using sensors installed within and around machine tools. These sensors measure quantities, such as vibrations, cutting forces, temperature, currents, power consumption, and acoustic emission, to diagnose defects and enable *zero-defect manufacturing* as part of the Industry 4.0 vision. The continuity of high-quality sensor data streams is fundamental to predicting phenomena, such as geometric deformations, surface roughness, excessive coolant use, and imminent tool wear with adequate accuracy and appropriate timing. However, in practice, data acquired by some sensors can be of poor quality and unsuitable for prediction due to sensor faults stemming from environmental factors. In this paper, we answer if we can repair erroneous data in a faulty sensor based on data simultaneously available in redundant sensors that observe the same process. We present a *machine learning pipeline* to synthesize *virtual sensors* that can step in for faulty sensors to maintain reasonable quality and continuity in sensor data streams. We have validated the synthesized virtual sensors in four industrial case studies.

1. Introduction

Industry 4.0 is an ongoing industrial revolution where multivariate sensor data obtained from production lines are analyzed to diagnose problems in manufacturing processes. It aims to realize Zero Defect Manufacturing (ZDM), i.e., achieving zero defect in manufacturing through the continuous observation and control of manufacturing processes. However, automated sensor data analysis is reliable only when data is adequately high quality and available without interruption (Nguyen et al., 2022; Cassoli et al., 2022; Sen et al., 2022; Husom et al., 2022). Sensor data quality is adversely affected (i.e., corrupted or missing data) by several factors, e.g., electromagnetic noise (Ze et al., 2019), thermal errors (Li et al., 2021), and sensor overload (Dahl and Engineer, 2001), or even cyber-attacks (Rajmohan et al., 2022). Erroneous data repair has an important role in advancing ZDM due to the increased accuracy and reliability of the data used to control and monitor manufacturing processes. Manufacturers can ensure that the data used for quality control is accurate and reliable (hence improving the detection of defects and reducing the risk of faulty products shipped to customers). They can increase the accuracy of predictions of when the equipment needs maintenance which prevents downtime and reduces the risk of defects caused by equipment failure. Data repair can improve the accuracy of process control algorithms, which can help optimize manufacturing processes and reduce the risk of defects. Furthermore, repaired data can improve the accuracy of real-time monitoring and decision-making, which can help detect and correct issues before they lead to defects.

Erroneous sensor data is either removed (resulting in missing values/Not-a-number) or tagged with an error code indicating unreliable and unusable data points. The inevitable occurrence of erroneous data interrupts the reliability and execution of data-driven prediction components in Industry4.0 for activities such as tool condition monitoring (Patil et al., 2021), anomaly detection (Pittino et al., 2020), and predictive maintenance (Zonta et al., 2020). Therefore, our motivation is to answer if we can automatically repair erroneous sensor data to maintain the continuity and accuracy of data-driven prediction in manufacturing.

Correlations among various data sources (sensors) can be learned through Machine Learning (ML) techniques to substitute one sensor with another sensor and predict missing values or new data that replace corrupt data. Non-ML data repair approaches (e.g., Lin et al.,

* Corresponding author.
E-mail address: arda.goknil@sintef.no (A. Goknil).

https://doi.org/10.1016/j.compind.2023.103917
Received 14 July 2022; Received in revised form 3 February 2023; Accepted 3 April 2023
Available online 13 April 2023

2019; Khan and Algarni, 2020; Weiss et al., 2013; Kong et al., 2021; Wang et al., 2018; Russell et al., 2019) have different constraints (e.g., the availability of dependent data computations in the application state history and the applicability for only a few data types), which limit their applicability in industrial manufacturing settings. A few approaches (Okafor et al., 2020; Flick et al., 2019) in the literature apply ML to data repair for industrial manufacturing systems. However, they do not exploit correlations among sensors and do not address challenges in deploying ML models in production for online (real-time) data repair.

In this paper, we present an ML pipeline for Erroneous Data Repair for Industry 4.0 (*ErDRe*) to synthesize *virtual sensors* that can repair erroneous sensor data. The synthesis of virtual sensors is based on training ML models that can predict the most representative values for data repair using inputs from redundant sensors observing the same manufacturing process. Our pipeline is open-source and built using an open-source framework called Data Version Control (DVC) (iterative.ai, 2023). DVC facilitates the process of configuring and synthesizing virtual sensors as *configurable stages*.

The training data of the pipeline is a batch of reference multivariate time series acquired from N sensors $(s_1, s_2, \ldots, s_N)$ observing the manufacturing process under normal operation where the degradation in data quality is minimal and within acceptable tolerances (i.e., when a production cycle is successful before environmental and aging factors affect sensor data quality). The pipeline is configured to generate target virtual sensor v_i for faulty sensor s_i using data from sensors $s_1, s_2, \ldots, s_P$ (where $P \leq N$ and $s_i \notin \{s_1, s_2, \ldots, s_P\}$) correlated to s_i. The pipeline stages are (i) data profiling to identify candidate sensors for sensor synthesis, (ii) data cleaning, (iii) generating engineered features (e.g., slope, gradient) from raw data, (iv) cordoning data for training/evaluation, (v) normalizing and sequencing data for training ML models, (vi) training the ML models and (vii) evaluating ML models on unforeseen input data. When the model performance is satisfactory, the model is deployed as a service that can perform both online and offline data repair.

We validated our approach over four case studies. We systematically investigated different configurations (i.e., various window sizes, input features, ML model types and ratio of training and test set sizes) while creating virtual sensors for the case studies. The best ML model for each case study was used to evaluate the performance on unforeseen sensor data. Our main contributions are as follows.

- **Novel Data Repair Approach.** We introduce an ML pipeline that exploits correlations among various data sources to substitute one with another and predict missing values or new data replacing corrupt data.
- **New Tool Support.** We introduce new and open-source tool support that enables online and offline data repair.[1]
- **Evaluation on Industrial Case Studies.** Virtual sensors effectively repair erroneous data in industrial datasets.

The rest of the paper is structured as follows. In Section 2, we present related work on erroneous data repair. Section 3 introduces the problem context. In Section 4, we describe the core technical solutions. Section 5 reports on the results of the empirical validation. We conclude the paper in Section 6.

2. Related work

ML has the potential for data repair in industrial manufacturing settings as correlations among various data sources (sensors) can be learned to substitute one sensor with another sensor and predict missing values or new data that replace corrupt data. Non-ML repair techniques (i.e., Lin et al., 2019; Khan and Algarni, 2020; Weiss et al., 2013; Kong et al., 2021; Wang et al., 2018; Russell et al., 2019) have different constraints which limit their applicability in industrial manufacturing settings. For instance, Lin et al. (2019) require all the dependent data computations in the application state history, which are not always available. Russell et al. (2019) present an approach that repairs the initial sensed data from cameras (the processed output from the edge) with the raw data from an ambient sensor. It uses sensory substitution to increase the data robustness, resilience, and dependability. The approach is limited to the data obtained from cameras and ambient sensors for motion detection. Manufacturing systems have various sensors (e.g., for measuring vibration, acoustic, pressure, temperature, accelerometer, and torque) and produce a massive amount of data in several forms. Virtual metrology (Dreyfus et al., 2022) is a closely related topic that is gaining relevance in zero-defect manufacturing. It emphasizes the use of simulation models and/or AI models to estimate product quality directly from process data (e.g., sensors on the machine). It enables manufacturers to predict defects and trends in product quality well in advance. We can easily transform our approach for creating virtual sensors to a virtual metrology technique if product quality data is synchronized and made available with process data. Nevertheless, we do not investigate virtual metrology in this article as our experimental evaluation focuses on the problem of repairing erroneous data and improving data quality.

ML can support more generic repair solutions for manufacturing systems having multiple sensors that can substitute each other. However, we revealed only two studies (Okafor et al., 2020; Flick et al., 2019) that apply ML to data repair in manufacturing and Industry 4.0. Flick et al. (2019) use ML algorithms (K-means for clustering and regression modeling) only to detect outliers in the clusters, not to predict new values replacing outliers. They employ the overflow, overweight, substitution value, and algebraic sign calculations to calculate the new values. Okafor et al. (2020) present an approach using linear regression and neural networks to correct sensor output. Their technique determines the factors affecting data quality, models their effects on the sensor response, and applies the calibration model to calibrate sensors. It merges data from multiple sensor nodes into the calibration equation to ensure consistent and accurate information for the calibration model. These two studies do not exploit sensor correlations, do not address the deployment challenges of ML models for online data repair, and do not implement different deployment scenarios of ML models on edge and cloud.

To the best of our knowledge, *ErDRe* is the first ML pipeline that exploits correlations among sensors for erroneous data repair in both online and offline scenarios. It can be invoked either on edge or cloud to create ML models based on the availability of training data. The models are containerized as online repair services and deployed on edge for real-time data repair, while they can also run on the cloud for offline data repair.

3. Problem context

Let $S = \{s_1, s_2, \ldots, s_N\}$ be the set of all sensors monitoring a manufacturing process (see Fig. 1). The problem of synthesizing a virtual sensor v_i entails using data from candidate input sensors $C \subseteq S$ to predict the measurements of a faulty sensor $s_i \notin C$. Our motivation relies on the fact that multiple sensors monitoring the same manufacturing process might have *latent* relationships with each other. Therefore, we can use time-varying measurements/signals from one or more sensors to predict time-varying measurements/signals of another (faulty) sensor. For instance, if one of two sensors recording similar vibration signals on different parts of a CNC milling machine is faulty due to electromagnetic interference, a virtual sensor with input data from the other sensor can be used to repair the measurements of the faulty vibration sensor.

An *ML model* $f_\theta(X, Y)$ can be used to learn a latent relationship between time-varying data from C to predict time-varying measurements

[1] https://github.com/sintef-9012/erdre

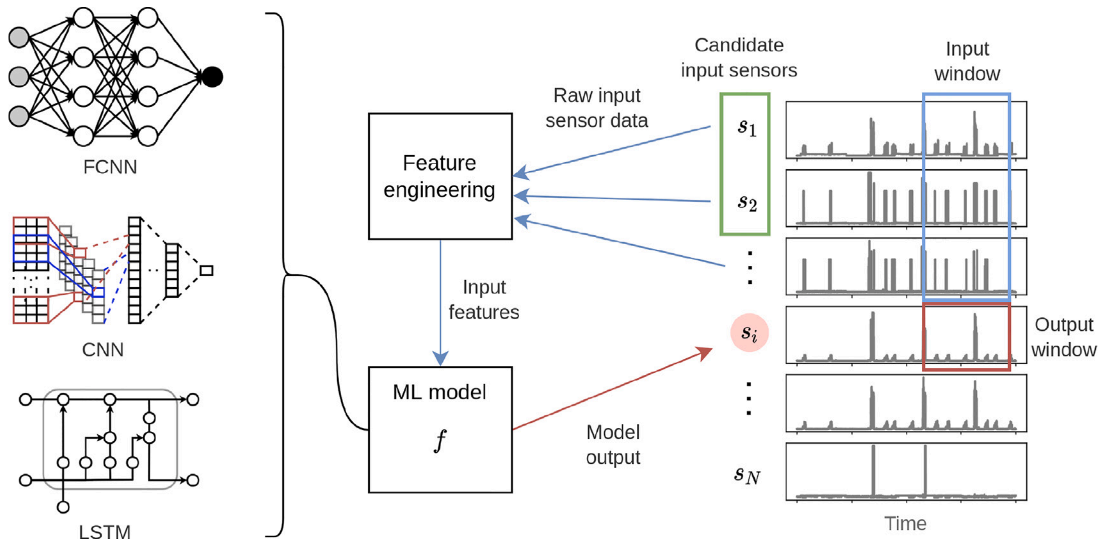

Fig. 1. Synthesizing virtual sensors for erroneous data repair.

of a target sensor s_i. The ML model f requires a high-quality time-varying input sensor dataset X from sensors in C and corresponding time-varying output dataset Y from target sensor s_i. The input and output datasets for learning f should be of high quality. They should be complete, accurate, timely, valid and ideally without faults (often available from early successful production cycles before aging affects equipment). f should be learnt based on a set of parameters θ. These parameters include the ML model type, e.g., a Dense Neural Network (DNN)/Fully Connected Neural Network (FCNN), Convolutional Neural Network (CNN), or a Long Short-Term Memory network (LSTM) (Goodfellow et al., 2016), *window sizes* for input and target sensor data, and percentage split in how much data is used for training and evaluation. Furthermore, virtual sensor v_i should embody f and be deployed as a service (e.g., web-service).

4. Pipeline for erroneous data repair

ErDRe aims to train ML model $f_\theta(X, Y)$ and facilitate configuring the set of parameters θ to explore different virtual sensors. Furthermore, it generates assets to embody f as a service for continuous deployment. We present an overview of our pipeline in Fig. 2 and describe its *stages* below.

Stage 1: Data Profiling. This stage includes computing non-linear *maximum information coefficient* (Reshef et al., 2011) and linear *Pearson's coefficient* (Sedgwick, 2012) to find correlations between data columns of different sensors in S. It generates statistical quantities for each column and alerts if any column contains several zeros or missing values. An interactive HTML dashboard is automatically generated to inspect correlations and select the sensors (from C) correlated to faulty sensor s_i.

Stage 2: Data Cleaning. This stage uses the output of Stage 1 (i.e., alerts for data columns having several zeros and missing values) to automatically remove unwanted data (e.g., columns containing several constant values or null values). Missing values cannot be used to train ML model f. Therefore, affected observations are removed (a missing value can also be replaced by zero if it is consistent with the behavior of the affected variable). Data cleaning parameters, e.g., threshold for variables having null values, can be set in the pipeline configuration. The output of this stage is high-quality datasets X and Y for training and evaluating f.

Stage 3: Feature Engineering. The raw input data in X obtained from Stage 2 may contain noise (poor signal to noise ratio). This stage extracts statistical properties, called *features*, from X that exhibit invariance to noise. Furthermore, the feature-based representations of time-series data (Lubba et al., 2019) perform well in classifying tasks at a fraction of the computational cost of processing raw time-series data. Table 1 presents a list of engineered features where $x_i \in 1, \ldots, w$ is the raw time series data in the rolling window of size w, x is the vector of data points across the rolling window, and t_{peaks} is the time between peaks (distance between heights of nearby points computed using `find_peaks()` Virtanen et al., 2020). While configuring the stage, we may select (optional) engineered features as input to train f.

Table 1
List of engineered features used in our evaluation.

Feature name	Mathematical definition
Mean	$\mu = \frac{1}{w}\left(\sum_{i=1}^{w} x_i\right)$
Sum	$S = \sum_{i=1}^{w} x_i$
Maximum	$M = \max(x)$
Minimum	$m = \min(x)$
Range	$M - m$
Gradient	$\nabla x = \frac{x_{i+1} - x_{i-1}}{2d}$
Slope	$\theta = \arctan\left(\frac{x_i - x_{i-1}}{d}\right)$
Sine of slope	$s_\theta = \sin(\theta)$
Cosine of slope	$c_\theta = \cos(\theta)$
Standard deviation	$\sigma = \sqrt{\frac{1}{w}\sum_{i=1}^{w}(x_i - \mu)^2}$
Variance	$v = \frac{1}{w}\sum_{i=1}^{w}(x_i - \mu)^2$
Peak frequency	$f = \frac{1}{t_{\text{peaks}}}$
Related quantities	**Symbol**
Rolling window size	w
Time step between each data point	d
Time between peaks	t_{peaks}

Stage 4: Splitting Training and Test Datasets. The datasets X and Y can be split into *training* and *test* datasets. The training set is used to develop a good hypothesis/ML model f through hyper-parameter tuning. A part of the training dataset is used for validation in Stage 7. The test dataset is unforeseen (locked away) during model training and used as an unbiased dataset to evaluate virtual sensor performance in repairing erroneous data (in Stage 8).

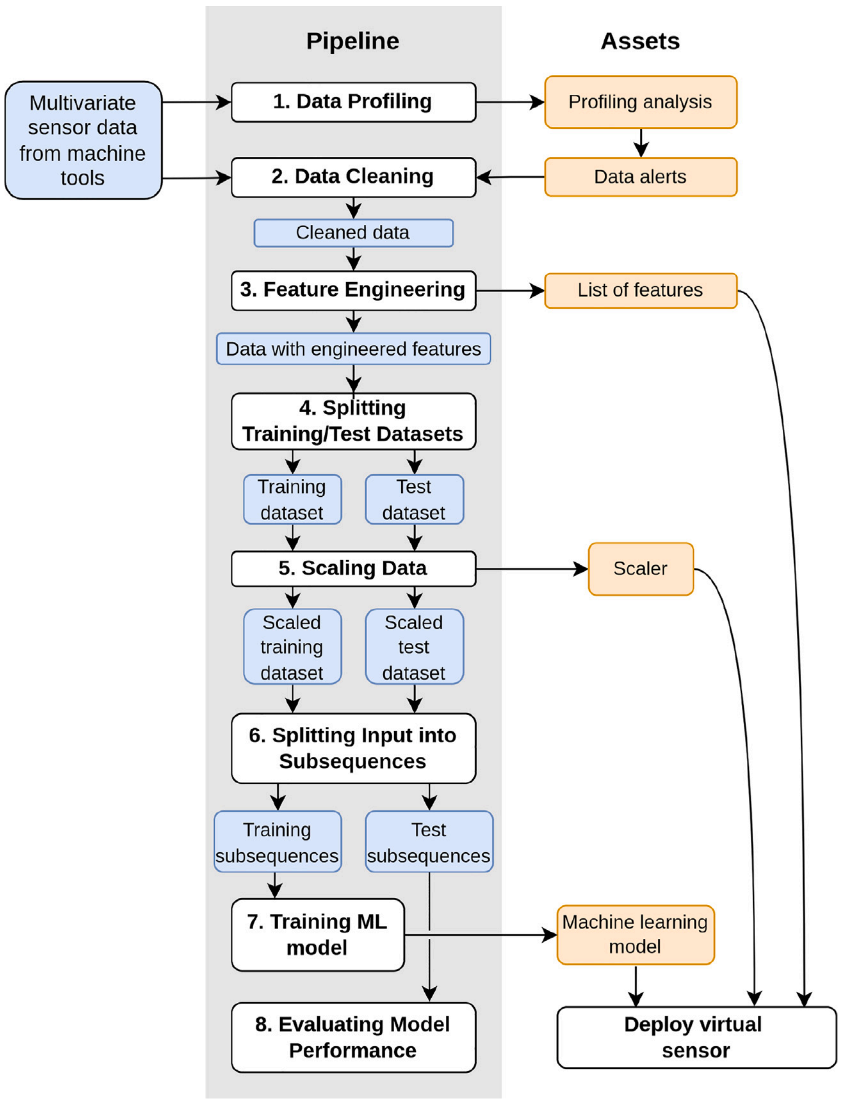

Fig. 2. ML pipeline to synthesize virtual sensors.

Stage 5: Scaling Input and Output Data. The training dataset contains signals/measurements from different sensors with varying value ranges and therefore needs to be scaled (LeCun et al., 2012) for a comparable influence during training. The pipeline saves scaling configuration (as an object) as it is used again while treating raw data in a deployed virtual sensor in production.

Stage 6: Splitting Input into Subsequences. This stage *restructures* the training (including validation dataset) and test datasets into input and output subsequences of the specified *window size* since virtual sensor predictions are based on a window of time-varying observations from input sensors and desired window of output values. Input subsequences of window size $window_i$ can be overlapping (sliding window) or non-overlapping. The target output values of window size $window_o$ can be a single sensor value ($window_o = 1$) or a sequence of values ($window_o > 1$).

Stage 7: Training an ML Model. The input and output subsequences from Stage 6 are used to configure and train model f. ErDRe enables specifying learning parameters and selecting ML model types (architectures). We consider DNNs/FCNNs, CNNs (LeCun et al., 1989), and LSTMs (Hochreiter and Schmidhuber, 1997) to predict time-varying values of a virtual sensor. A small part of the training data (e.g., 20%) is set apart before training to be used as a *validation dataset*. ErDRe automatically stops training if the prediction error of the validation set stops improving, preventing the overfitting of the model to the training data. It saves model f for evaluation.

Stage 8: Evaluating Model Performance. The test dataset is an unforeseen dataset used to evaluate the model performance to minimize bias due to hyper-parameter tuning in Stage 7. We compare the model output and the ground truth to assess how well model f predicts the target variable. To do so, ErDRe generates the plots of predictions on test data. We use Mean Squared Error (MSE), coefficient of determination (R^2 score), and Mean Absolute Percentage Error (MAPE) to evaluate the performance of model f.

Implementation of ErDRe. We implemented ErDRe using DVC (iterative.ai, 2023). The pipeline stages are Python programs configured using a parameter file. DVC can be integrated into Git and allows model and data versioning while maintaining the consistency of models and data across the entire pipeline.

Deploying virtual sensors. A virtual sensor embodies the trained, validated, and evaluated model f as a service (e.g., Flask web service) with an API. The API is invoked using subsequences of data (of size $window_i$) from input sensors and returns a set of target sensor values (of size $window_o$) with time stamps. The raw input sequences cannot always be used by f as they are since f is trained on input data features extracted from raw data and bounded (e.g., values between 0 and 1) by a scaling operation. The feature engineering operations

Fig. 3. Finished Wax Part with an "S" Shape.

require the use of ML libraries such as Sci-kit learn (Pedregosa et al., 2011) and TensorFlow (Abadi et al., 2016). Therefore, parts of the pipeline used in inference, such as code to compute engineered features (Stage 3), scaler (Stage 5), and the ML model (Stage 7) with all its dependencies (e.g., ML libraries), are packaged as a standalone container (e.g., docker). We can deploy the container on an edge device for online (real-time) erroneous data repair or the cloud for offline repair (data repair on long-term data).

5. Evaluation

In this section, we investigate, based on four industrial datasets, the following Research Questions (RQs):

- ***RQ1.*** *How can data profiling be used to generate virtual sensors?*
- ***RQ2.*** *What machine learning architectures are effective in generating virtual sensors?*
- ***RQ3.*** *What features of data from other sensors are useful in creating virtual sensors with good performance?*
- ***RQ4.*** *To what extent does a virtual sensor perform on unforeseen data?*

5.1. Subjects of the evaluation

We perform erroneous data repair using sensor data from four case studies (**C1, C2, C3,** and **C4**) presented in this section.

5.1.1. Publicly available CNC milling tool wear data (C1)

The Computer Numerical Control (CNC) milling tool wear dataset (Anon, 2023) was obtained using the System-level Manufacturing and Automation Research Testbed (SMART) at the University of Michigan. Eighteen experiments were run on 2" × 2" × 1.5" wax blocks in a CNC milling machine; each experiment produced a finished wax part with an "S" shape - S (see Fig. 3).

Time-series dataset was collected from the machine sensors. It has measurements from the four motors in the CNC (X, Y, Z axes and spindle). They are motor position, velocity, and acceleration; the commanded values of these variables (from path planning in the machine) are in the dataset. The motor current, voltage, and power were measured (except the power of the z-axis). The data sampling frequency is 10 Hz, and the total sample count is 25,286.

C1: We predict spindle power using commanded X and Y positions, the relevant estimation of the energy consumption of a motion plan generated by the CNC machine.

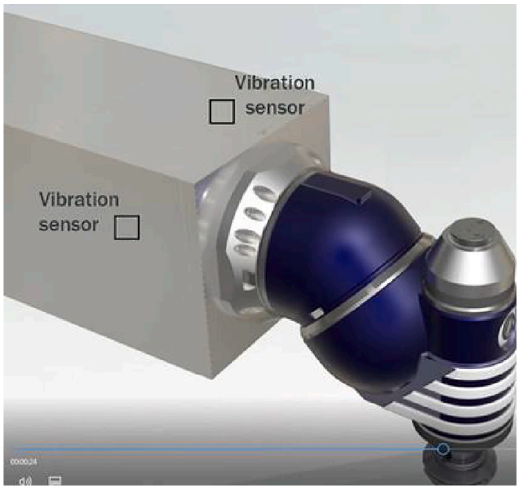

Fig. 4. IDEKO test machine.

5.1.2. Streaming data from test machine at IDEKO (C2)

The dataset contains streaming data from IDEKO (a manufacturing company), i.e., low-frequency (1 Hz) data aggregated from high-frequency data obtained from a test CNC machine and stored on cloud infrastructure. The machine is to test process parameters for new parts before production. The dataset consists of 3,528 records (streamed at 1 Hz for 1-hour on the 3rd of May 2021); each one reports on a single event and has 121 attributes with a time stamp.

The dataset consists of temperature measurements, ten important frequencies and amplitudes from the frequency spectrum of high-frequency accelerometers, and vibration severity from multiple accelerometers. Vibration data were acquired using two accelerometers/vibration sensors located at the ram tip of the machine, measuring both bending directions of the ram (see Fig. 4).

C2: We repair the vibration severity data measured by one accelerometer using the vibration severity data measured by another accelerometer for vibration damping.

5.1.3. Broaching of jet engine turbine discs (C3)

Broaching is a manufacturing process for forming internal or external round, flat, or contoured surfaces. A broaching machine pushes a multi-toothed cutting tool, a broach, into a workpiece to remove material (see Fig. 5). Slots of various dimensions are cut at high production rates. Our dataset was collected from three broaching tools in a tool holder broaching fifty slots for three hours. The broaching operation was to broach *fir tree slots* on jet engine turbine discs.

We had three data sources: (i) two accelerometers with a sample rate of 12.8 kHz, (ii) a data logger with a rate of 250 Hz, and (iii) tool wear measured with an optical microscope. Tool wear was recorded for every five slots. Two types of data were collected: average wear, which gives information about how cutting conditions are for the broaching tool, and maximum wear, which includes micro tool breakage and is needed to replace the tool.

C3: We repair X axis acceleration data of accelerometer 2 using X, Y and Z axis acceleration of accelerometer 1.

5.1.4. CNC milling of combustion chambers in car cylinder heads (C4)

The dataset was collected from a CNC machine of a European car manufacturer. It was recorded over four weeks at a sampling rate of 10 Hz during the milling process of the combustion chamber of a cylinder head (see Fig. 6).

Table 2
Input-target sensor correlations for case studies.

Case study	Target sensor	Input sensor
CNC Milling (C1)	S1 Output Power	Y1 Command Position (ρ: −0.697045, MIC: 0.706563) X1 Command Position (ρ: −0.670205, MIC: 0.639470)
IDEKO (C2)	Severity of Accelerometer 1	Severity of accelerometer 2 (ρ: 0.836260, MIC: 0.347422)
Broaching (C3)	X axis in Accl. 2	X axis in Accelerometer 1 (ρ: −0.015470, MIC: 0.050781) Y axis in Accelerometer 1 (ρ: 0.093902, MIC: 0.050896) Z axis in Accelerometer 1 (ρ: 0.004596, MIC: 0.083667)
Automotive (C4)	Spindle Torque	Axis X Position (ρ: 0.038328, MIC: 0.427362) Axis Y Position (ρ: −0.029126, MIC: 0.529575) Axis Z Position (ρ: 0.020412, MIC: 0.582200)

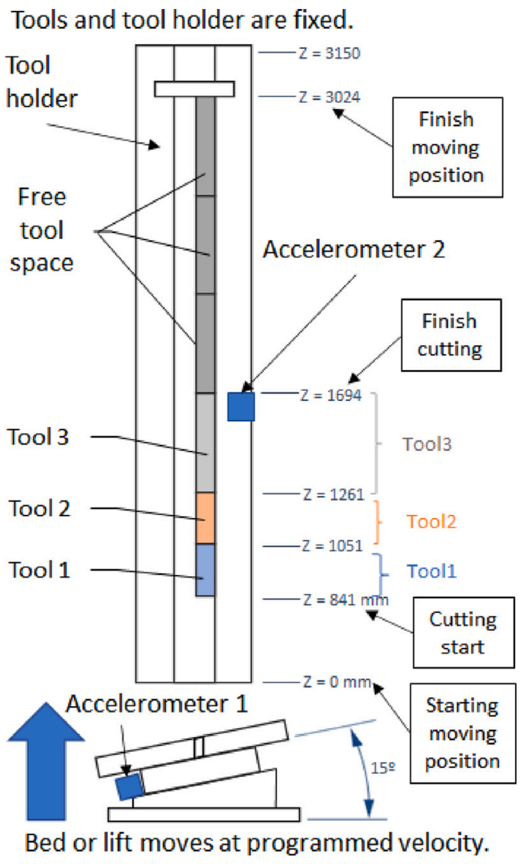

Fig. 5. Broaching machine.

The dataset includes time series for (i) linear axis X, a gantry axis with two linear motors X1 and X2 (e.g., X1 and X2 real position in millimeter), (ii) linear axis Y and Z with only one motor (e.g., real position in millimeter and torque in percent of motor drive nominal torque), (iii) rotational axis A (e.g., position in degree and speed in degree per minute), and (iv) spindle (e.g., speed in rotation per minute and torque).

C4: We repair the spindle torque data using axis positions as spindle torque is used for tool wear prediction.

5.2. Results of the evaluation

This section discusses the results of our case studies, addressing, in turn, each of the RQs.

RQ1: How can data profiling be used to generate virtual sensors?

To address RQ1, we investigated the linear and non-linear correlation between sensor variables ErDRe computes in the data profiling stage to identify candidate input sensors for virtual sensors. Table 2 presents, for **C1**, a strong correlation between spindle power and the command positions on the x-axis (ρ=−0.670205, MIC=0.639470) and y-axis (ρ=-0.697045, MIC=0.706563). The negative Pearson's coefficient ρ is because the movement in X and Y values indicates that the spindle moves to a specific location without milling and consumes power at a static location. The high Maximum Information Coefficient (MIC) values capture the non-linear relationship to create the virtual sensor for spindle power.

There is a strong linear correlation among the severities of accelerometers 1 and 2 in **C2** (ρ: 0.83626, MIC: 0.347422) due to their close proximity. We can use the severity amplitudes to predict the severity of another accelerometer (see Fig. 8).

Table 2 presents, for **C3**, a weak correlation between the acceleration in the X-axis (in accelerometer 2) and X, Y, and Z-axis variables in accelerometer 1, since the two accelerometers measure the acceleration of two different but connected parts of the broaching machine. The lift (observed by accelerometer 1 in Fig. 5) moves at programmed velocity to move up and down the broaching tool (observed by accelerometer 2).

Table 2 presents, for **C4**, a significant non-linear correlation (high MIC values but low ρ values) between the spindle torque and workpiece axis positions. The correlation is likely because the spindle contacts with the workpiece at some coordinates and produces more torque.

RQ1 Conclusion. Candidate input sensors for virtual sensors can be determined using linear Pearson's coefficient and non-linear maximum information coefficient scores.

RQ2: What machine learning architectures are effective in generating virtual sensors?

To address RQ2, we investigated the performance of different ML architectures/models for virtual sensors in our case studies. All possible combinations of input features in **C1** performed well (R^2 = [0.69, 0.8] & MSE = 0.001) for any model (see Table 3). However, based on the coefficient of determination (R^2 = 0.8) and mean square error (MSE = 0.001008), LSTM is the best architecture (the best ones for all cases are highlighted in green) with the raw feature for the Y1 command position and the engineered features for the X1 command position. The best ML

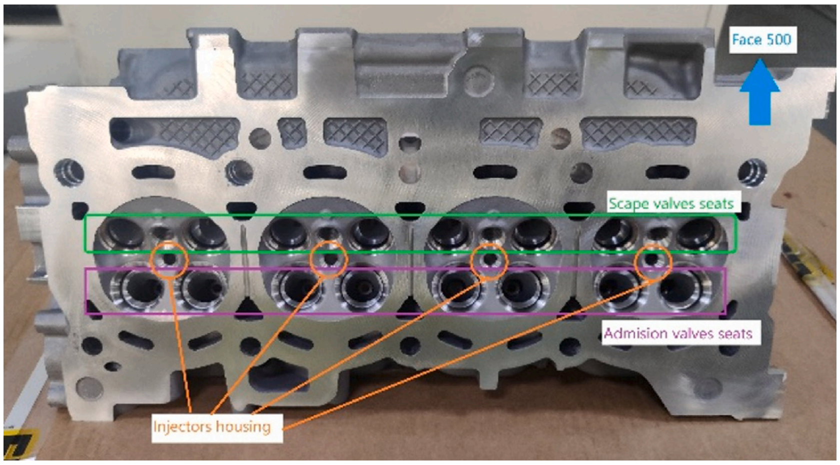

Fig. 6. Combustion chamber.

Table 3
Model performance for the CNC milling dataset (C1).

Feat. set	Features				Performance					
	X1 Command Position		Y1 Command Position		DNN		CNN		LSTM	
	Raw	Eng. features	Raw	Eng. features	MSE	R^2	MSE	R^2	MSE	R^2
1	x		x		0.001102	0.783763	0.001077	0.788668	0.001110	0.782228
2	x			x	0.001409	0.720667	0.001133	0.775306	0.001119	0.778150
3		x	x		0.001292	0.743946	0.001081	0.785697	0.001008	0.80024
4		x		x	0.001517	0.699318	0.001208	0.760574	0.001251	0.752055

Table 4
Model performance for the IDEKO dataset (C2).

Feat. set	Features		Performance					
	Severity of accelerometer 2		DNN		CNN		LSTM	
	Raw	Eng. features	MSE	R^2	MSE	R^2	MSE	R^2
1	x		0.028318	−0.069767	0.019863	0.241584	0.011730	0.717607
2		x	0.018846	0.297966	0.021356	0.212721	0.017192	0.355563

Table 5
Model performance for the broaching dataset (C3).

Feat. set	Features						Performance					
	X axis of accl. 1		Y axis of accl. 1		Z axis of accl. 1		DNN		CNN		LSTM	
	Raw	Eng. features	Raw	Eng. features	Raw	Eng. features	MSE	R^2	MSE	R^2	MSE	R^2
1	x		x		x		0.0103	0.0799	0.0125	−0.1127	0.0116	−0.0357
2	x		x			x	0.0105	0.0698	0.0116	−0.0255	0.0243	−1.1454
3	x			x	x		0.0104	0.0759	0.0137	−0.2174	0.0254	−1.2458
4	x			x		x	0.0105	0.0721	0.0183	−0.6226	0.0274	−1.4219
5		x	x		x		0.0105	0.0705	0.0114	−0.0119	0.0312	−1.7553
6		x	x			x	0.0118	−0.0456	0.0278	−1.4540	0.0225	−0.9927
7		x		x	x		0.0113	−0.0024	0.0133	−0.1812	0.0306	−1.7078
8		x		x		x	0.0109	0.0340	0.0235	−1.0799	0.0221	−0.9562

architecture in **C2** is LSTM using the raw severity of accelerometer 2 to predict the severity of accelerometer 1 (see Table 4). R^2 for the LSTM model is 66% and 71% (better than R^2 for CNN and DNN).

The combinations of input features in **C3** did not perform as well as the combinations of input features in other datasets (see Table 5) due to the weak correlation between the **C3** variables. Based on R^2 (= 0.0799) and MSE (=0.0103), DNN is the best architecture with the raw features of X, Y, and Z-axes of accelerometer 1. The architecture producing the best model for **C4** is LSTM (see Table 6) when using only engineered features (R^2 = 0.858). The CNN model performance is poorer, but the model creation is more time-efficient.

The best ML architecture in our experiments based on R^2 and MSE scores is LSTM. Some literature reviews (Alom et al., 2019; Emam et al., 2020) justify the performance of LSTM; LSTM can look at long sequences of inputs without increasing the network size. It is the slowest architecture to train. CNN is a good alternative for time efficiency at the expense of accuracy.

RQ2 Conclusion. LSTM models perform well in terms of accuracy in most cases, but they need more training time. CNN models are a good compromise between performance and training time.

RQ3: What features of data from other sensors are useful in creating virtual sensors with good performance?

To address RQ3, we investigated the performance of virtual sensors created with and without engineered features. Using raw sensor data vs. engineered features brings insight into the robustness of a virtual sensor

Table 6
Model performance for the automotive dataset (C4).

Feat. set	Features						Performance					
	Axis X Position		Axis Y Position		Axis Z Position		DNN		CNN		LSTM	
	Raw	Eng. features	Raw	Eng. features	Raw	Eng. features	MSE	R^2	MSE	R^2	MSE	R^2
1	x		x		x		52.349	0.402	33.975	0.612	30.199	0.655
2	x		x			x	28.634	0.672	19.332	0.778	17.141	0.804
3	x			x	x		34.493	0.605	21.931	0.749	20.656	0.763
4	x			x		x	27.713	0.682	15.130	0.827	12.529	0.856
5		x	x		x		36.078	0.587	25.669	0.706	22.149	0.746
6		x	x			x	28.079	0.678	15.523	0.822	13.596	0.844
7		x		x	x		32.960	0.622	20.622	0.764	19.422	0.777
8		x		x		x	23.690	0.7285	14.728	0.831	12.366	0.858

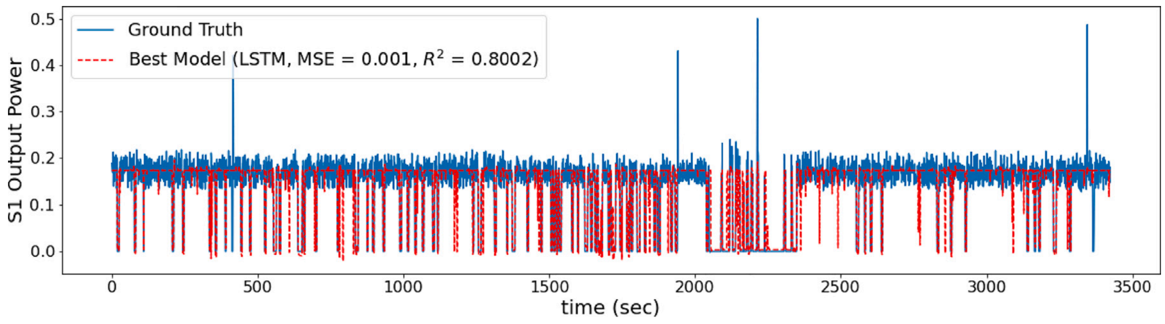

Fig. 7. C1: Predicting the spindle output power using the X and Y command positions.

using low-dimensional statistical features vs. its need to use high-dimensional raw data containing specific patterns and non-statistical properties.

Table 3 shows that the LSTM model using raw data for the Y1 command position and engineered features for the X1 command position performs the best (models only using raw data also perform well) in **C1**. Models using raw severity data from accelerometer 1 to predict the severity of accelerometer 2 in **C2** perform better than models using engineered features (see Table 4). The DNN model using raw data performs the best in **C3** (see Table 5). Models using engineered features in **C4** perform significantly better than models using raw data.

ML architectures automatically constructed an internal feature representation from raw data in **C1**, **C2**, and **C3**, having smaller datasets than the **C4** dataset. In **C4**, models using the engineered feature *Slope* perform better than models using raw data, most likely due to the availability of four weeks of data.

RQ3 Conclusion. We did not observe an obvious benefit of engineered features in model performance since ML architectures automatically construct internal feature representations for relatively small datasets.

RQ4: To what extent does a virtual sensor perform on unforeseen data?

To address RQ4, we investigated the performance of the best model (highlighted green in Tables 3–6) in each dataset on unforeseen data. ErDRe cordons part of the unforeseen input/output data for unbiased evaluation of virtual sensors (see Stage 4 in Fig. 2). Fig. 7 presents the prediction of spindle power in **C1** alongside the ground truth. The high R^2 score (= 0.8) indicates the high prediction accuracy of ErDRe. The prediction is closer to the average power during milling, and it reactively drops to zero when moving between positions. Furthermore, we can use the virtual sensor for spindle power before production to estimate the energy consumption and carbon footprint of manufacturing.

Fig. 8 presents the severity prediction for accelerometer 1 using severity from accelerometer 2 in **C2**. Severity is computed using the amplitude spectrum of the accelerometer using the formula $Severity = \sqrt{\sum_{i=1}^{N} x^2}$, where N is the number of the lines of the amplitude spectrum, and x is the spectrum value for each spectral line. We observe that accelerometer 2 can replace accelerometer 1 (R^2 = 0.71). The virtual sensor detected about 66% of the transients accurately. There are two transients/spikes observed in the predicted data for approximate severity of 1.5 m/s. The magnitude of these repairs is not harmful to most applications in monitoring tool wear. Consulting a domain expert, we note that only vibrations above 5–6 mm/s vibration can lead to problems on a workpiece. These high-severity vibrations are more consistent and not just transients (hence a clearly distinguishable pattern over time).

Fig. 9 presents the acceleration prediction in Y-axis for accelerometer 2 using X, Y, and Z-axes accelerations measured by accelerometer 1 in **C3**. The low R^2 score (= 0.08) indicates the low prediction accuracy caused by the distance between the accelerometers. Generally, virtual accelerometer 2 follows a similar pattern to physical accelerometer 2, although it fails to match the amplitude, particularly in the negative direction. This behavior is exasperated after 30000 s (high-frequency vibrations are recorded by physical accelerometer 2 but are not predicted by virtual accelerometer 2). We explain these factors with virtual accelerometer 2's performance on test data ($R^2 = 0, 0799$), where the predictions of virtual accelerometer 2 are less erroneous for low-frequency vibrations (while predictions for high-frequency accelerations are poor). Therefore, we conclude that virtual accelerometer 2 cannot replace accelerometer 2 under all circumstances (especially when the input from physical accelerometer 1 does not contain predictive information to estimate high-frequency vibrations recorded by accelerometer 2). We may improve virtual accelerometer 2 by moving physical accelerometer 1 to a position on the broaching machine that makes it more sensitive to negative accelerations in accelerometer 2. However, this investigation for an optimal location for accelerometer 1 was not undertaken as part of this research, and we present it only as a recommendation.

Despite this low score however, as seen in Fig. 9, the virtual sensor can detect the periods of activity when the broaching occurs close to accelerometer 2. We can use the virtual sensor as an activity sensor when accelerometer 2 fails close to the broaching machine in cases the temperature is high and the coolant flow is limited/obstructed.

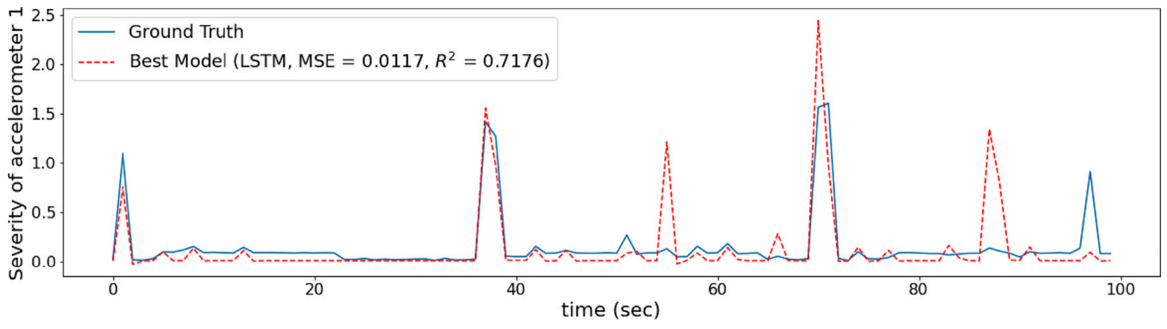

Fig. 8. C2: Predicting the severity of accelerometer 1 using the severity of accelerometer 2.

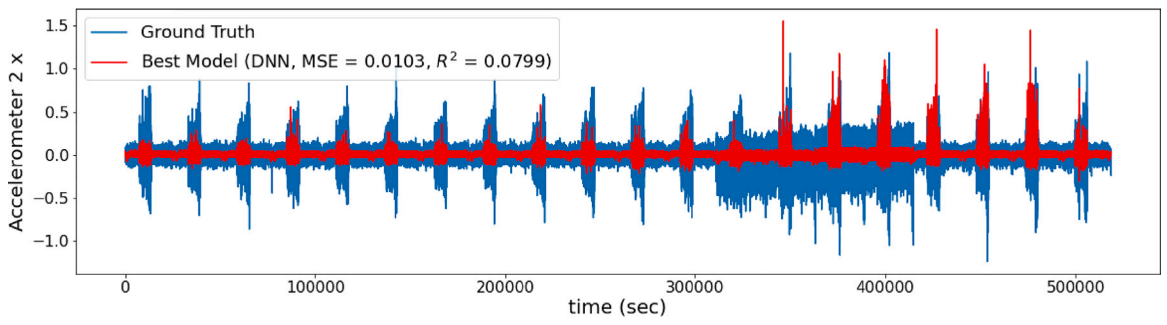

Fig. 9. C3: Predicting the Y axis acceleration of accelerometer 2 using the X, Y, and Z axis accelerations of accelerometer 1.

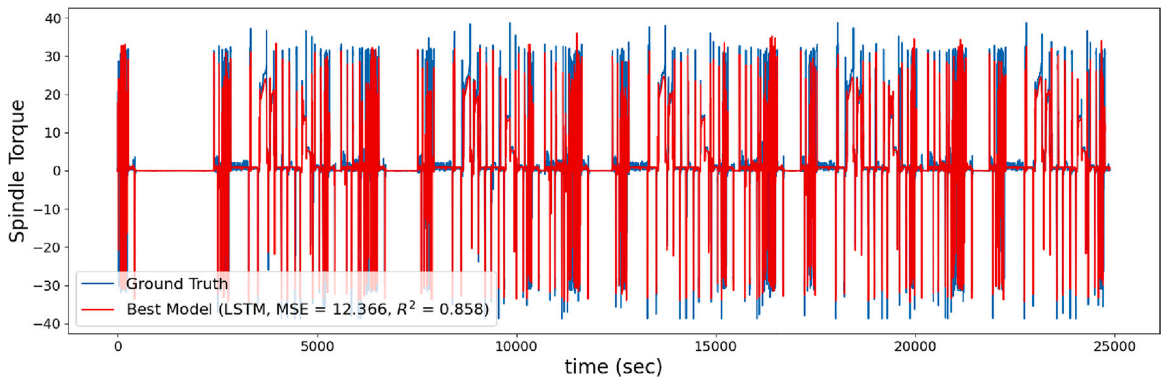

Fig. 10. C4: Predicting the spindle torque using the X, Y, and Z workpiece position.

Fig. 10 presents the spindle torque prediction in **C4**. The high R^2 score (= 0.858) indicates that the virtual sensor predicted most patterns without anomalies although the predictions have some irregularities/fluctuations (and deviations from the ground truth) when predicting long periods of spindle torque. The fluctuations can be smoothed using a filter.

RQ4 Conclusion. Virtual sensors effectively repair erroneous data if redundant/correlated sensors are available. They may exhibit rare irregularities/fluctuations in their predictions, which we can smooth using a filter.

5.3. Threats to validity

Internal validity. To limit threats to internal validity, we use virtual sensors to exploit the correlation between sensor variables and not a cause–effect relationship between input and output sensors. Virtual sensors are not designed to predict future values (using previous time frames). For instance, the spindle can be in a specific position in a time frame and may move in the next time frame elsewhere based on the CNC application. However, in **C1** and **C4**, we see that part position may have a causal relationship to spindle power and torque as motion plans for CNC milling are determined and simulated before real-world manufacturing takes place.

External validity. To mitigate the threat to generalizability, we designed our pipeline as reusable, modular, and extensible to new ML architectures and methods to pre-process time-varying data. However, ErDRe requires synchronized reference data from different sensors to train virtual sensors. Furthermore, virtual sensors based on deep learning models cannot extrapolate well to data outside of the distribution of the reference data.

Industrial sensor data experience different distributional shifts: (a) *concept drift* that occurs when the underlying distribution of the sensor data changes over time (resulting in a shift in the data distribution) and (b) *sensor drift* that occurs when the sensor experiences a shift in its measurement capabilities. Concept drift happens due to changes in the manufacturing process, equipment wear and tear, and environmental factors. Sensor drift may occur due to physical damage, calibration errors, or changes in temperature or humidity. All these distributional shifts may affect the virtual sensor performance. Therefore, we should estimate uncertainty in virtual sensor predictions and learn from new data. In future work, we intend to take a Bayesian approach (e.g., *dropout* Gal and Ghahramani, 2016) to obtain an uncertainty estimate as a standard deviation. A standard deviation above a certain threshold should trigger re-training from a new batch of sensor data to adapt to distributional shifts. However, it is also desirable to replay (Hayes et al., 2021) some of the old data to mitigate the problem of *catastrophic forgetting* (French, 1999) in neural networks where gradients from new data overwrite weights from past learning. We expect the virtual sensor performance to be poor initially but gradually improve with more data continually adapting to distributional shifts.

6. Conclusion

We presented an open-source ML pipeline to create virtual sensors for erroneous data repair in manufacturing. The pipeline supports selecting features, window size, and ML architectures to train, validate, and test virtual sensors. Furthermore, we can deploy virtual sensors as services on edge and cloud infrastructure omnipresent in Industry 4.0 manufacturing to perform online and offline erroneous data repair. We assess the usefulness and performance of virtual sensors using four real-world case studies in manufacturing using a publicly available dataset amenable to scientific reproduction and real-world datasets from the aerospace and automotive domains.

Future Work. The quality of the data acquired for decision support is getting more important since more manufacturing data are shared across industrial ecosystems (Isaja et al., 2023; Tran et al., 2023). Sharing quality data could be one of the key aspects in realizing zero-waste strategies at the value chain as one of the future research directions beyond zero-defect manufacturing (Powell et al., 2022). In addition, erroneous data repair could be one of the inspection solutions and advanced monitoring to support continuous manufacturing processes, which are less explored in terms of zero-defect manufacturing (Powell et al., 2022). On the other hand, manufacturing environments are highly dynamic, where processes are adjusted during production, and ML models may become obsolete. A phenomenon known as *concept drift* may be detected using uncertainty estimation methods, e.g., Bayesian neural networks (Charnock et al., 2022) and conformal prediction (Stankeviciute et al., 2021). Confidence in virtual sensor predictions gives information about concept drift. With large confidence intervals, we are uncertain about the virtual sensor predictions. There is a need to update/retrain the virtual sensor due to a shift in input data distribution. We will investigate uncertainty estimation (Jourdan et al., 2020) in conjunction with continual/lifelong learning (Parisi et al., 2019) to train virtual sensors with new data while avoiding the catastrophic forgetting of what is already learned and addressing the stability-plasticity dilemma.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: All authors reports financial support was provided by European Commission.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work was supported by the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No. 958357 (InterQ).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., 2016. Tensorflow: A system for largescale machine learning. In: OSDI’16. pp. 265–283.

Alom, M.Z., Taha, T.M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M.S., Hasan, M., Van Essen, B.C., Awwal, A.A.S., Asari, V.K., 2019. A state-of-the-art survey on deep learning theory and architectures. Electronics 8 (3).

Anon, 2023. The (CNC) milling tool wear dataset. https://www.kaggle.com/datasets/shasun/tool-wear-detection-in-cnc-mill.

Cassoli, B.B., Jourdan, N., Nguyen, P.H., Sen, S., Garcia-Ceja, E., Metternich, J., 2022. Frameworks for data-driven quality management in cyber–physical systems for manufacturing: A systematic review. In: 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering, Vol. 112. pp. 567–572.

Charnock, T., Perreault-Levasseur, L., Lanusse, F., 2022. Bayesian neural networks. In: Artificial Intelligence for High Energy Physics. World Scientific, pp. 663–713.

Dahl, K., Engineer, D., 2001. Accelerometer overload.

Dreyfus, P.-A., Psarommatis, F., May, G., Kiritsis, D., 2022. Virtual metrology as an approach for product quality estimation in industry 4.0: A systematic review and integrative conceptual framework. Int. J. Prod. Res. 60 (2), 742–765.

Emam, A., Shalaby, M., Aboelazm, M.A., Bakr, H.E.A., Mansour, H.A., 2020. A comparative study between CNN, LSTM, and CLDNN models in the context of radio modulation classification. In: 2020 12th International Conference on Electrical Engineering. ICEENG, pp. 190–195.

Flick, D., Gellrich, S., Filz, M.-A., Ji, L., Thiede, S., Herrmann, C., 2019. Conceptual framework for manufacturing data preprocessing of diverse input sources. In: 2019 IEEE 17th International Conference on Industrial Informatics, Vol. 1. INDIN, IEEE, pp. 1041–1046.

French, R.M., 1999. Catastrophic forgetting in connectionist networks. Trends in Cognitive Sciences 3 (4), 128–135.

Gal, Y., Ghahramani, Z., 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning. PMLR, pp. 1050–1059.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Hayes, T.L., Krishnan, G.P., Bazhenov, M., Siegelmann, H.T., Sejnowski, T.J., Kanan, C., 2021. Replay in deep learning: Current approaches and missing biological elements. Neural Comput. 33 (11), 2908–2950.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Husom, E.J., Tverdal, S., Goknil, A., Sen, S., 2022. Udava: An unsupervised learning pipeline for sensor data validation in manufacturing. In: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI. pp. 159–169.

Isaja, M., Nguyen, P., Goknil, A., Sen, S., Husom, E.J., Tverdal, S., Pedersen, K.J., Anand, A., Jiang, Y., Myrseth, P., Stang, J., Niavis, H., Pfeifhofer, S., Lamplmair, P., 2023. A blockchain-based framework enabling trusted quality data sharing towards zero-defect manufacturing. Comput. Ind. 146, 103853.

iterative.ai, 2023. Open-source version control system for machine learning projects. https://dvc.org/.

Jourdan, N., Rehder, E., Franke, U., 2020. Identification of uncertainty in artificial neural networks. In: Uni-DAS EV, Vol. 2. p. 12.

Khan, M.A., Algarni, F., 2020. A healthcare monitoring system for the diagnosis of heart disease in the iomt cloud environment using msso-anfis. IEEE Access 8, 122259–122269.

Kong, T., Hu, T., Zhou, T., Ye, Y., 2021. Data construction method for the applications of workshop digital twin system. J. Manuf. Syst. 58, 323–328.

LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R., 2012. Efficient backprop. In: Neural Networks: tricks of the Trade. Springer, pp. 9–48.

LeCun, Y., et al., 1989. Generalization and network design strategies. Connect. Perspect. 19, 143–155.

Li, T.-j., Zhao, C.-y., Zhang, Y.-m., 2021. Real-time thermal error prediction model for CNC lathes using a new one-dimension lumped capacity method. Int. J. Adv. Manuf. Technol. 1–12.

Lin, W.-T., Bakir, F., Krintz, C., Wolski, R., Mock, M., 2019. Data repair for distributed, event-based iot applications. In: The 13th ACM International Conference on Distributed and Event-Based Systems. DEBS’19, pp. 139–150.

Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S., 2019. Catch22: Canonical time-series characteristics. Data Min. Knowl. Discov. 33 (6), 1821–1852.

Nguyen, P.H., Sen, S., Jourdan, N., Cassoli, B., Myrseth, P., Armendia, M., Myklebust, O., 2022. Software engineering and ai for data quality in cyber- physical systems - sea4dq’21 workshop report. SIGSOFT Softw. Eng. Not. 47 (1), 26–29.

Okafor, N.U., Alghorani, Y., Delaney, D.T., 2020. Improving data quality of lowcost IoT sensors in environmental monitoring networks using data fusion and machine learning approach. ICT Express 6 (3), 220–228.

Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Netw. 113, 54–71.

Patil, S., Pardeshi, S., Patange, A., Jegadeeshwaran, R., 2021. Deep learning algorithms for tool condition monitoring in milling: A review. J. Phys. Conf. Ser. 1969 (1), 012039.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Pittino, F., Puggl, M., Moldaschl, T., Hirschl, C., 2020. Automatic anomaly detection on in-production manufacturing machines using statistical learning methods. Sensors 20 (8), 2344.

Powell, D., Magnanini, M.C., Colledani, M., Myklebust, O., 2022. Advancing zero defect manufacturing: A state-of-the-art perspective and future research directions. Comput. Ind. 136, 103596.

Rajmohan, T., Nguyen, P.H., Ferry, N., 2022. A decade of research on patterns and architectures for IoT security. Cybersecurity 5 (1), 2.

Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J., Lander, E.S., Mitzenmacher, M., Sabeti, P.C., 2011. Detecting novel associations in large data sets. Science 334 (6062), 1518–1524.

Russell, L., Kwamena, F., Goubran, R., 2019. Towards reliable IoT: Fog-based AI sensor validation. In: IEEE Cloud Summit. pp. 37–44.

Sedgwick, P., 2012. Pearson's correlation coefficient. Bmj 345.

Sen, S., Husom, E.J., Goknil, A., Tverdal, S., Nguyen, P., Mancisidor, I., 2022. Taming data quality in AI-enabled industrial internet of things. IEEE Software 39 (6), 35–42.

Stankeviciute, K., Alaa, A.M., van der Schaar, M., 2021. Conformal time-series forecasting. Adv. Neural Inf. Process. Syst. 34.

Tran, T., Nguyen, P.H., Erdogan, G., 2023. A systematic review of secure IoT data sharing. In: The International Conference on Information Systems Security and Privacy, Science and Technology Publications.

Virtanen, P., Gommers, R., Oliphant, T.E., et al., 2020. SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 17, 261–272.

Wang, C., Zhu, Y., Shi, W., Chang, V., Vijayakumar, P., Liu, B., Mao, Y., Wang, J., Fan, Y., 2018. A dependable time series analytic framework for cyberphysical systems of IoT-based smart grid. ACM Trans. CyberPhys. Syst. 3 (1).

Weiss, S.M., Dhurandhar, A., Baseman, R.J., 2013. Improving quality control by early prediction of manufacturing outcomes. In: KDD'13. pp. 1258–1266.

Ze, Y., Liu, L., Cheng, T., Kun, Z., Jianhua, Z., 2019. Measurement based characterization of electromagnetic noise for industrial internet of things at typical frequency bands. In: 2019 IEEE Wireless Communications and Networking Conference. WCNC, IEEE, pp. 1–6.

Zonta, T., da Costa, C.A., da Rosa Righi, R., de Lima, M.J., da Trindade, E.S., Li, G.P., 2020. Predictive maintenance in the industry 4.0: A systematic literature review. Comput. Ind. Eng. 106889.