

# Fault detection and classification in Industrial IoT in case of missing sensor data

Merim Dzaferagic, Nicola Marchetti, Irene Macaluso

**Abstract**—This paper addresses the issue of reliability in Industrial Internet of Things (IIoT) in case of missing sensors measurements due to network or hardware problems. We propose to support the fault detection and classification modules, which are the two critical components of a monitoring system for IIoT, with a generative model. The latter is responsible of imputing missing sensor measurements so that the monitoring system performance is robust to missing data. In particular, we adopt Generative Adversarial Networks (GANs) to generate missing sensor measurements and we propose to fine-tune the training of the GAN based on the impact that the generated data have on the fault detection and classification modules. We conduct a thorough evaluation of the proposed approach using the extended Tennessee Eastman Process dataset. Results show that the GAN-imputed data mitigate the impact on the fault detection and classification even in the case of persistently missing measurements from sensors that are critical for the correct functioning of the monitoring system.

**Index Terms**—Generative Adversarial Network, Industrial IoT (IIoT), data imputation, fault detection, fault classification.

## I. INTRODUCTION

The Internet of Things (IoT) is a computing paradigm that relies on ubiquitous connection to the Internet, where common objects are turned into connected devices. Up to trillions of smart objects are being deployed, that are capable of sensing their surroundings, transmit and process acquired data, and then in turn feedback relevant information to the environment. A subset of IoT, the Industrial Internet of Things (IIoT) encompasses machine-to-machine (M2M) and industrial communication technologies with applications in the automation sector. IIoT paves the way for better understanding of the manufacturing process, with positive repercussions on the efficiency and sustainability of the production system [1], [2].

In the era of IoT big data, the integration of cloud computing technologies and cyber-physical systems enables the full potential of Industry 4.0 to be harvested in manufacturing processes, with a multitude of sensors being installed around the industrial operating environment and equipment. The networked sensors would continuously send monitoring data, allowing for proactive maintenance to take place, leading to a reduction in the unplanned downtime via data analysis techniques [3]–[7].

The work proposed in this article is based on the FIREMAN project funded by the CHIST-ERA programme, which focuses on modelling and analysing IIoT networks based on specific machine learning algorithms capable of detecting rare events in industrial setups in an ultra-reliable way [8]–[10]. An

important aspect in assuring such ultra-reliability in the IIoT is how to guarantee we have a functioning system in place, even in case some of the measurements are missing due to network or hardware issues. In fact values are often missing from the collected sensor data, and the related issue of missing value imputation becomes then very important. For example, high-frequency data collection can result in large gaps in the data and if the network stops working, all the measurements collected during the network downtime will be missing [3]. Other possible reasons behind missing data are: faulty sensors producing intermittent readings, loss of data during wireless communication owing to packet loss or to interference in the communication medium, or data removed purposely by attackers with malicious intentions during sensing, processing, storing or communication. A related research challenge is to impute the missing values, to enable the data to be analyzed while ensuring that the imputed values are as close as possible to the true values. What complicates things with regard to the imputation of missing data in IoT, is that the data to be collected in such systems is diverse, and the techniques developed must therefore provide a high level of confidence for different types of applications, besides the need to be robust to the increase in the scale of IoT (and IIoT) deployments. Furthermore, techniques must be lightweight to be able to fulfil real-time IoT application requirements [11].

All the approaches reported to date in the literature focus on either data imputation, anomaly detection or fault classification for an industrial process. In this paper we instead propose a framework that unifies all three techniques, allowing us to optimize the monitoring system by taking into account each component and their interconnections. We propose a data-driven decomposition of the process to monitor the various indicators of the health of a machine/component or an entire industrial process. Instead of proposing a new tailored solution to collect, communicate and process data in an industrial environment, we focus on the detection and classification of system failures based on a dataset with missing values, investigating in particular the impact of missing data on the monitoring system in an industrial setting. This is a very important issue to tackle, as small reconstruction errors of missing sensor data could greatly affect the capability of the monitoring system. The data imputation module in our framework relies on a generative adversarial network (GAN) model that learns the correlation between the data from the input layer to replace missing sensor measurements. The GAN was optimized by validating its performance based on the effect of the imputed measurements on the fault identification and detection modules, which ultimately constitute the two

M. Dzaferagic, N. Marchetti, and I. Macaluso are with CONNECT centre, Trinity College Dublin, Ireland.

essential tasks performed by the monitoring system. As we will discuss later, the GAN-imputed data mitigate the loss on these two modules even in the case of persistently missing measurements from sensors that are critical for the correct functioning of the monitoring system.

Section II reports an account of the relevant literature, while also positioning our work highlighting the main differences and advantages of our approach. Section III describes the proposed framework, describing the adopted fault detection, fault classification and GAN-based missing data imputation techniques. Section IV analyses the performance of our techniques in terms of recall and precision, and showing the impact the proposed data imputation has on both metrics. Section V concludes the paper and outlines some possible promising directions for future research.

## II. STATE OF THE ART

IoT is based on the idea of connecting the physical and the digital worlds [12]. Initially, Radio-Frequency IDentification (RFID) was the main technology in this space, allowing microchips to identify themselves to a reader wirelessly [13]. Today, IoT applications have moved from the simple RFIDs, by integrating diverse sensing, computing and communication technologies. An IoT platform provides services and features like: node management, connectivity and network management, data management, analysis and processing, security, access control and interfacing [14]. Domains like transportation, healthcare, industrial automation and education are all affected by the fast development of these platforms. Different domains face different problems related to the deployment of IoT solutions (e.g. low latency communication, high reliability, massive data transfer, energy efficiency). Hence, different IoT platforms are needed to run specific applications.

In an industrial environment, the detection and prediction of anomalies are important for both economic and security reasons. Difficulties related to these tasks originate from the fact that anomalies are rare events within datasets, making it difficult to apply most of the existing algorithms which result in either false alarms or misdetections [15], [16]. The authors of [8] made an attempt at providing a general framework to model a wide range of cases based on the advances in IIoT networks and Machine Learning (ML) algorithms. Similarly, the authors of [17], [18] describe several deployed solutions of cyber-physical systems in an industrial environment. The most promising solutions include those presented in [19]–[21]. Unlike the work in the above-mentioned papers, instead of proposing a new tailored solution to collect, communicate and process data in an industrial environment, we focus on the detection and classification of system failures based on a dataset with missing values.

Fault detection in an industrial environment has always been a challenging task [22]–[26]. Due to the issues related to interoperability and communication between different devices, collecting a large dataset in such an environment is not an easy task. However, even if we assume that the dataset was collected, different problems arise when using such a dataset for anomaly detection (e.g. unbalanced dataset, noise in the

measurements, missing data). Similar to the work proposed by the authors of [3], [27], we also investigate the impact of missing data on the detection of rare events in an industrial setting. The authors of [27] propose a sensor data reconstruction scheme that exploits the hidden data dynamics to accurately estimate the missing measurements. In [3], the authors focus on missing data imputation for large gaps in univariate time-series data and propose an iterative framework, using multiple segmented gap iteration to provide the most appropriate values. All the approaches mentioned above focus on either data imputation, anomaly detection or fault classification for an industrial process. We, on the other hand, propose a framework that unifies all three techniques, allowing us to optimize each of them in a way that results in the best overall performance. Indeed we train the imputation model to minimize the false alarm rate of the anomaly detection model and the classification error of the fault identification model, so that the monitoring system performance is robust to missing data. In order to assess the gain of our unified approach we benchmarked its performance against the results obtained by considering independent modules that individually focus on the three aspects of the monitoring system, i.e. fault detection, fault classification, and data imputation. In particular, we considered two different options for the imputation module. The first method is the moving average (MA), which has been used as a benchmark for data imputation techniques both in the literature on time series imputation [28] and in works addressing the problem of missing measurements in Wireless Sensor Networks [29]. The second option is a well known iterative imputation method based on random forests called missForest (MF) and proposed in [30]. Since the algorithm relies on averaging over many unpruned classification or regression trees, it intrinsically constitutes a multiple imputation scheme. However, due to the specifics of the approach, the imputation time increases with the number of observations, predictors and number of predictors containing missing values. Since the approach is based on a post-processing algorithm (offline), it has to run each time missing data has to be imputed, which could be problematic in some production environments. The results presented in Section IV show that the proposed unified approach outperforms both benchmark options in mitigating the impact on the fault detection and classification, even in the case of persistently missing measurements from sensors that are critical for the correct functioning of the monitoring system.

## III. FRAMEWORK

In this work we propose a data-driven decomposition of the process to monitor the various indicators of the health of a machine/component or an entire industrial process. Since faults account only for a very small fraction of the data collected in an IIoT scenario, we separately address the problem of fault or anomaly detection and the problem of fault classification. The latter is only triggered in case an anomaly is detected, as shown by the data flow in Figure 1. The fault detection, for which we employ an autoencoder as discussed in Section III-A, can be trained using data collected

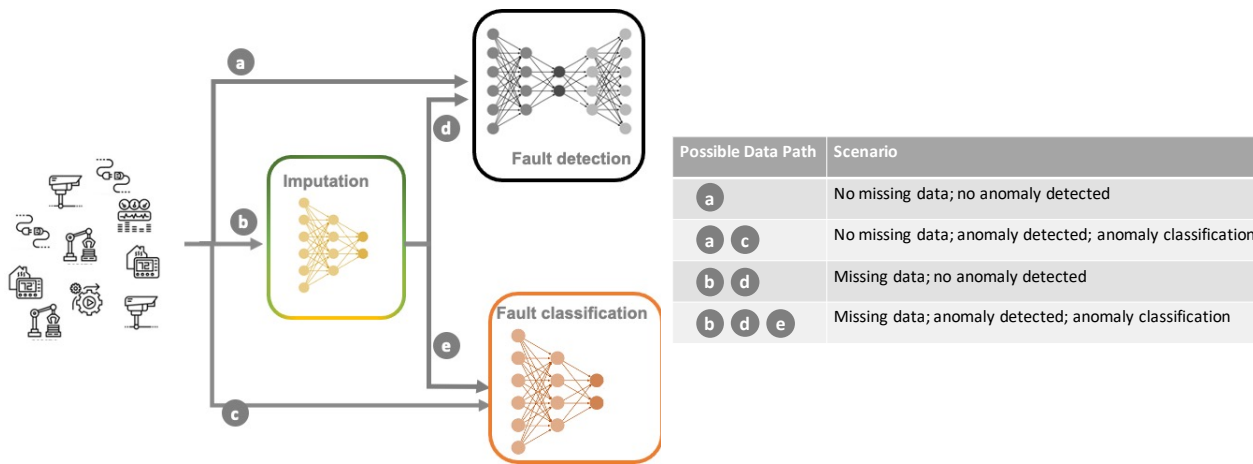


Fig. 1: Data flow within the monitoring system. If all measurements are received the fault detection is activated, followed by the fault classification in the event of an anomaly (paths  $a$  and  $a - c$ ). If some measurements are not received, the fault detection and identification are preceded by the data imputation module (paths  $b - d$  and  $b - d - e$ ).

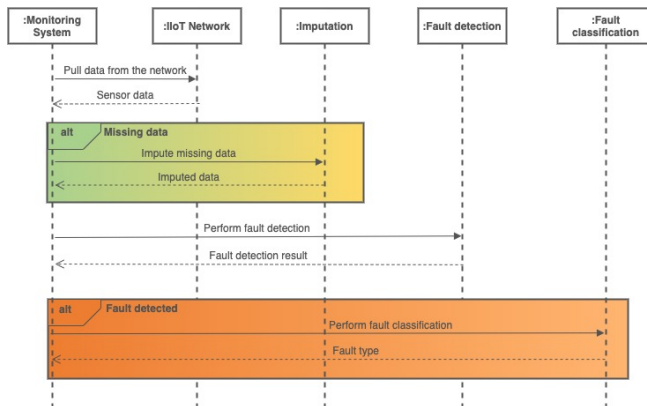


Fig. 2: Sequence diagram of the monitoring system. The two colored rectangles represent alternatives that are executed if the respective conditions are met.

during normal operation. The fault classification, discussed in Section III-B, can be deployed as soon as enough labeled data related to faults become available. Finally, the imputation module, described in Section III-C, requires both faulty and non-faulty data to replace missing sensors' data so that the monitoring process can continue without disruption even if some measurements are not received.

Figure 1 shows the data flow between the different components of the monitoring system, while the sequence diagram in Figure 2 details the interaction between these components. In case all sensors measurements are received, the fault detection procedure is always activated, while the fault classification is performed only if an anomaly is detected (path  $a - c$  in Figure 1). In case of missing measurements, the data imputation module is first triggered, followed by the fault detection and eventually by the fault classification if an anomaly is detected.

### A. Fault detection

We built the fault detection module using an autoencoder that receives as input the  $N$  sensor measurements used to

monitor the industrial process. The outputs of the autoencoder are the reconstructed values from the input. By minimizing the Root Mean Square Error (RMSE) of the reconstructed values, the model learns a representation for the input data and filters the noise. By training the model on fault-free data only, the autoencoder will learn the patterns of normal operating conditions. This way, when faulty data will be inputted the resulting reconstruction error should be larger than the error corresponding to fault-free data. It is worth noting that fault-free measurements represent the normal operation of the system, these measurements constitute the majority of the collected data. Hence the fault detection module can be readily deployed as soon as enough measurements are collected. After training the model to minimize the RMSE, we choose an RMSE threshold, which will indicate the presence/absence of a system fault.

Besides its main purpose, which is fault detection, the autoencoder is also used to fine-tune the training of the model for the missing sensor measurement imputation.

### B. Fault Classification

For the fault classification module, in this paper we adopt a deep neural network (DNN) that receives as input the time lags of the  $N$  sensor measurements used to monitor the industrial process. Another possible solution is to adopt a recurrent neural network. Fault classification is a multinomial (or multi-class) classification problem. While an unequal distribution of the faults might result in an imbalanced classification problem, we have addressed the most severe imbalance by training separately a model for the detection of anomalies that only requires fault-free data. In fact, fault-free sensor measurements, i.e. data collected in normal operating conditions, represent the vast majority of the data collected in IIoT. By separating the fault detection from the fault classification stage, the DNN for fault classification is only used after an unspecified fault has been detected (see Section III-A) to determine which fault has occurred. Therefore the DNN for fault classification is trained only using sensor measurements collected during faulty

operations to classify the different types of faults. For details on the structure of the DNN used in this work the reader is referred to Section IV-B.

The DNN is not only used to determine the type of fault that has occurred, but it is also the basis to fine-tune the training of the model used to impute missing data.

### C. GAN-based missing data imputation

The data imputation module relies on a Generative Adversarial Network (GAN) model that learns the correlation between the data from the input layer to replace missing sensor measurements. Considering that the main purpose of data imputation in our proposed architecture is to replace the missing values so that the fault detection and classification modules can operate correctly, the model requires both faulty and non-faulty data during training. We start from the Generative Adversarial Imputation Network (GAIN) model presented in [31]. In [31], the generator  $G$  observes the  $N$ -dimensional real data vector  $\mathbf{x}$  with missing components. Let us denote by  $M$  the mask that indicates the missing values in the input dataset. The mask  $M$ , when multiplied with the complete input dataset, produces a dataset with missing values. Then  $G$  imputes the missing components conditioned on what is actually observed, and outputs an imputed vector  $\hat{\mathbf{x}}$ . The discriminator  $D$  receives the output of the generator as its input. It takes the complete vector generated by  $G$  and attempts to determine which components were actually observed and which were imputed. In other words the discriminator  $D$  attempts to predict the mask  $M$ . In addition to the output of the generator,  $D$  receives a hint vector, which reveals partial information about the missing values. In particular, the hint vector reveals to  $D$  all the components of the mask  $M$  except for one, which is randomly independently chosen for each sample.

The training of the GAIN model is a two step process. We first optimize the discriminator  $D$  with a fixed generator  $G$  using mini-batches of size  $K_D$ .  $D$  is trained according to equation (1), where  $\mathcal{L}_D$  is defined with equation (2).

$$\min_D - \sum_{j=1}^{k_D} \mathcal{L}_D(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) \quad (1)$$

$$\mathcal{L}_D(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = \sum_{i:b_i=0} [m_i \log(\hat{m}_i) + (1 - m_i) \log(1 - \hat{m}_i)] \quad (2)$$

We denote with  $\mathbf{m}(j)$  the original mask associated with  $j$ -th sample in the mini-batch, while  $\hat{\mathbf{m}}(j)$  is the corresponding predicted mask, i.e. the output of  $D$ , and  $\mathbf{b}(j)$  is an  $N$ -dimensional vector whose elements are all equal to 1 except for one element that is 0. The position of the 0 element in  $\mathbf{b}(j)$  is the position of the only element of the mask  $\mathbf{m}(j)$  that is not provided as input to  $D$ . In other words, by using (2) we train  $D$  only for the element of the mask vector that is unknown to the discriminator for each sample, which is randomly chosen for each sample.

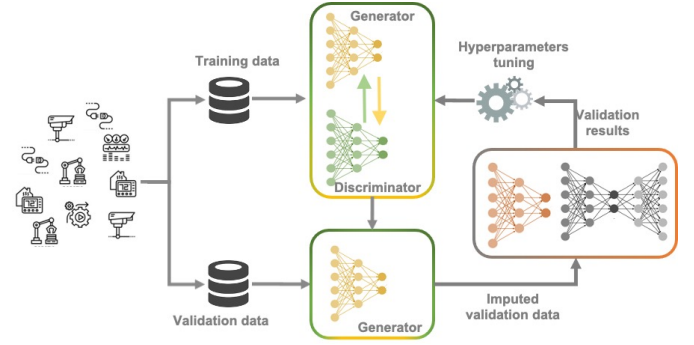


Fig. 3: Hyperparameters tuning for the GAN.

$$\min_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\mathbf{x}(j), \hat{\mathbf{x}}(j)) \quad (3)$$

After we run the training process for the discriminator  $D$ , we optimize the generator  $G$  according to equation (3) with mini-batches of size  $K_G$ . The cost function for  $G$  is the weighted sum with hyperparameter  $\alpha$  of two components: one which applies to the missing sensors measurements -  $\mathcal{L}_G$  (equation (4)); and one which applies to the observed measurements -  $\mathcal{L}_M$  (equation (5)).

$$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = - \sum_{i:b_i=0} (1 - m_i) \log(\hat{m}_i) \quad (4)$$

$$\mathcal{L}_M(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{i=1}^d m_i (\hat{x}_i - x_i)^2 \quad (5)$$

In [31], the authors use the RMSE as the metric to evaluate how well the model performs in terms of imputing the missing values. However, we noticed that even though the RMSE metric can be very low, in an industrial process certain measurements might have a lower tolerance to variations, while other measurements might have very limited impact on the capability to monitor the process. This is for example the case for the Tennessee Eastman (TE) process, which we used for the validation of our framework (see Section IV). Therefore, instead of relying on the RMSE, we use the feedback from the fault detection and classification modules to tune the hyperparameters of the GAIN model (see Figure 3).

The authors of [31] provide theoretical results for a dataset with values Missing Completely At Random (MCAR). Similarly, we use the MCAR approach for training of our model. However, in order to understand the impact of physical sensors failing, validation and testing are done on persistent sensor failures. That is where the feedback coming from the fault detection and classification modules plays an important role, because the imputed values have to provide enough information for these two modules to operate correctly.

## IV. EVALUATION

To train the models and test the performance of the framework presented in the previous section we use the TE process

dataset. The TE is a chemical process that was computationally modelled in 1993 and since then has become widely adopted for benchmarking process monitoring techniques. The TE process simulation produces data in correspondence to normal operation (fault-free data) and in correspondence to 21 different simulated process faults (faulty data). Two sets of data are generated - training and testing datasets. The data consists of a multivariate time series of  $N = 52$  variables sampled at an interval of 3 minutes. The training data and testing data span 25 hours of operation (i.e. 500 samples) and 48 hours of operation (i.e. 960 samples) respectively. While the original TE process dataset consisted of 22 runs, one normal and the remaining 21 for the faulty conditions, in this work we use the extended version provided by Reith et al. [32]. This extended dataset was generated by running 500 independent simulations for each of the runs, differing from the original ones for the random seeds used. Faults are introduced after 1 and 8 simulation hours in the training and testing files respectively. Our analysis presented in the remainder of this section does not include fault 21 since it was not part of the extended dataset. Finally, faults 3, 9, and 15 are not considered because of their unobservability from the data which results in high missed detection rates [33].

The remainder of this section presents a detailed analysis of the performance of the three components of the monitoring mechanism.

#### A. Fault detection

An autoencoder is a type of a neural network with a symmetric structure. The structure can be divided into two mirrored sub-structures (i.e. the encoder and the decoder). For the purpose of fault detection in this paper the autoencoder is constituted of: i) an input layer with  $N$  inputs, ii)  $H_A$  hidden layers with ReLu activation functions, and iii) an output layer with  $N$  outputs. The input layer and the first  $H_A/2$  hidden layers form the encoder, and the remaining  $H_A/2$  hidden layers and the output layer form the decoder. The neural network was trained with the Adam optimiser with a batch size of  $10^3$  samples and a constant learning rate of 0.001 for  $10^5$  epochs in total. The training set consists of 300 of the 500 training files for the fault free scenario. We used 100 of the remaining 200 training files as validation set to optimize the number of hidden layers  $H_A$  and neurons of the autoencoder. The resulting autoencoder has  $H_A = 12$  hidden layers of size [52, 52, 48, 47, 46, 45, 45, 46, 47, 48, 52, 52]. The resulting false alarm and missed detection rate computed on testing files are 0.12 and 0.17 respectively. In particular, we evaluated the false alarm rate using all 500 testing fault-free files and the missed detection rate on all the 500 faulty data files for each of the 17 faults under consideration.

#### B. Faults Classification

The DNN used for fault classification was trained with the Adam optimiser with a batch size of 512 samples and a constant learning rate 0.005 for  $10^3$  epochs in total. The training set consists of 300 of the 500 training files for all the considered faults. Training data corresponding to normal

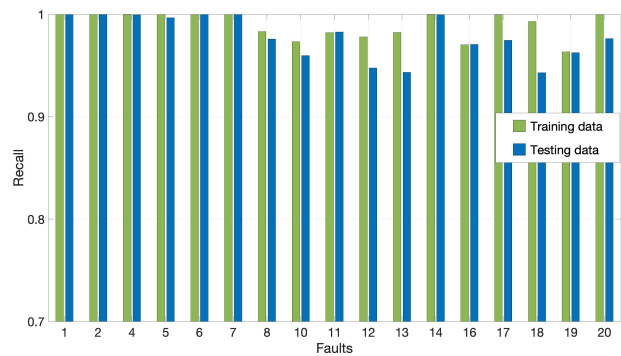


Fig. 4: Recall of the DNN for training and testing data.

operating conditions are not employed to train the DNN since this classifier is only used after an anomaly has been detected. To tune the network hyperparameters (number of neurons, number of hidden layers, and number of lags  $L$ ) we used 100 of the remaining 200 training data files not used during training. The resulting DNN structure for the fault classifier is constituted by: i) an input layer with  $N \times (L + 1)$  inputs, ii) two hidden layers with ReLu activation functions and iii)  $N$  softmax functions as output layer. The number of lags  $L$  is 20, and the number of neurons in the two hidden layers is 250 and 60 respectively. Finally, we tested the DNN using all 500 testing data files for each of the faults under consideration.

Figure 4 and Figure 5 show the recall and the precision for each fault for both training and testing data. With a minimum recall of 0.963 and 0.943 on the training set and on the test set, respectively, the DNN performs very well and correctly identifies most occurrences of each fault. In particular the minimum recall in the test set occurs for fault 18. Previous results testing a number of techniques, e.g. stacked sparse auto encoders, to classify faults on the TE process dataset show a performance below 0.90 on fault 18 [34]. In general, the results shown in Figure 4 outperform all the classifiers tested in [34] for 7 of the considered faults. For the remaining 10 faults the performance of the DNN and that of the best performance<sup>1</sup> as reported by [34] are comparable. A recent work proposing an enhanced random forest algorithm has also been tested on the TE Process dataset [35]. The analysis performed in our paper and the random forest results presented in [35] only overlap for 12 faults. In 8 of those 12 faults, the DNN recall outperforms the enhanced random forest method in [35]. In the case of fault 13, the difference in performance is more than 60%. For the remaining 4 faults, both approach exhibit the same recall. The DNN also performs very well in terms of precision for each fault, i.e. what proportion of the samples identified by the model as an instance of a fault corresponds to an actual occurrence of that fault.

#### C. GAN-based missing data imputation

The GAN was trained using 400 of the 500 training files for all the considered faults and the normal operating conditions.

<sup>1</sup>It is worth noting that according to the results presented in [34] there is no one single best technique for all faults.

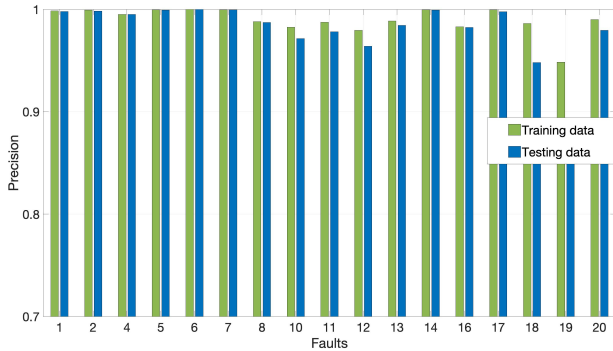


Fig. 5: Precision of the DNN for training and testing data.

TABLE I: Dimensions of the discriminator and generator networks.

	Input layer size	Hidden layers size	Output layer size
Generator	624	[1144, 1144]	52
Discriminator	104	[200, 200]	52

These are the same 400 files that we used to train and validate the fault detection autoencoder and the fault classification DNN. We used the remaining 100 training files as validation data for the GAN. It is important to highlight that these 100 files were not used in the training or validation of the the fault detection autoencoder and the fault classification DNN, since these two models are used in the hyperparameters selection for the GAN. As mentioned earlier, the proposed architecture has to work as one interconnected system, meaning that to understand how the GAN model performs we cannot rely exclusively on validation metrics related to the GAN itself. We have to consider the system as a whole, and look at the impact on the fault detection and classification modules of the data reconstructed by the GAN. In particular, we used the precision and recall of the fault classification module and the missed detection and false alarm rate of the fault detection module to tune the GAN hyperparameters, i.e. to determine the number and size of the layers for the generator  $G$  and discriminator  $D$ , the value of  $\alpha$  in (3) and the missing probability, i.e. how many sensors are missing for each sample during training. We considered 4 different values for  $\alpha$ , 3 different options for the number of hidden neurons for  $G$  and  $D$ , 2 options for the number of hidden layers for  $G$  and  $D$ , and 3 values for the missing probability, resulting in 432 configurations of these hyperparameters. We randomly sampled 50 distinct configurations and trained and validated the GAN in correspondence to each of them. Table II shows the chosen  $\alpha$  parameter and the missing probability  $p_m$ , while Table I reports the selected size for the generator and the discriminator. As shown in Table I, both the generator  $G$  and the discriminator  $D$  have one input layer, two hidden layers and one output layer. The input of the generator  $G$  consists of: the current sensor measurements (with missing values), the measurements for the last  $L_G = 10$  time steps and the mask that indicates which measurements are missing. Considering

TABLE II: GAN hyperparameters

	$\alpha$	$p_m$	Batch size	#epochs
Generator	100	0.1	1000	250,000
Discriminator	/	0.1	1000	250,000

that our system has  $N = 52$  sensors, the size of the input layer is  $N + N \times L_G + N = 624$ . The input of the discriminator  $D$  consists of: the current sensor measurements (with imputed values - output of  $G$ ) and the hinting vector. Hence, the size of the input layer of  $D$  is  $N + N = 104$ .

In the remainder of this section the performance of the GAN-based imputation and the performance of the MA and MF methods that were used as benchmarks, are measured with respect to the impact on the fault-classification DNN and the fault detection autoencoder. The MA method simply replaces missing values in a time series with the average of the last  $W$  observed values. In our simulations  $W = 10$ , the same as the number of lags  $L_G$  used by the GAN.

We first run  $10^3$  simulations in which approximately 10% of the measurements are randomly missing, i.e. 5 of the  $N = 52$  sensors are missing in each sample. We repeated the same experiment by considering persistent sensors' failures. The results of this preliminary analysis show that in most cases the MA mechanism achieves a very good performance. However, on closer examination of these results we observed a drop in performance of the fault-classification DNN when specific sensors were missing. In light of this, we conducted a thorough analysis by systematically removing sensors one by one and measuring the impact for each fault when imputing the missing values using the MA. This analysis showed that only 12 sensors have a significant impact on the fault classification mechanism. The set  $S_C$  of critical sensors is  $S_C = \{0, 8, 12, 17, 18, 20, 21, 43, 44, 49, 50, 51\}$ . Some of these sensors impact multiple faults (e.g. sensor 17), others are critical just for the detection of one of the faults (e.g. sensor 0). If any of the remaining 40 sensors is missing and its value is imputed using an MA or even using the average over fault-free data, the recall of DNN is always greater than 0.80. On the other hand, if even one of the 12 identified critical sensors is missing, the MA-based imputation does not result in an acceptable loss. In fact, in some cases the recall of the DNN can drop below 0.40. In light of these results, we focus our analysis on these 12 critical sensors to evaluate the performance of the imputation mechanism. The results of this analysis are shown in Figure 6. For each of the 500 testing files of each fault we simulated the unavailability of each of the critical sensors 10 time slots after the fault occurred and for the entire duration of the simulated process. Since the number of lags  $L_G$  used by GAN is equal to 10, for each testing file the first missing sensor measurement imputed by the GAN uses the actual sensors measurements. Henceforth, and until the end of the simulated process, the GAN relies on imputed values for the missing sensors. At each time slot, we performed the imputation using the generator of the trained GAN and the MA. Since the MF cannot be used as an online algorithm, we provide as input to the MF each testing

file as a whole, after having removed each of the critical sensors 10 time slots after the fault occurred. We then fed the imputed data to the fault classifier DNN to determine the resulting performance of the two imputation mechanisms. Let us denote by  $P(f)$  and  $R(f)$  the precision and recall for fault  $f$  of the DNN in correspondence to the original testing data. We also denote by  $R_{MA}(f, s)$  and  $P_{MA}(f, s)$  the recall and precision for fault  $f$  of the DNN when classifying data imputed using MA for missing sensor  $s$ . In a similar way, we define  $R_{GAN}(f, s)$  and  $P_{GAN}(f, s)$  in case of the GAN-based imputation, and  $R_{MF}(f, s)$  and  $P_{MF}(f, s)$  in case of the MF-based imputation. By computing the average recall and precision over all the critical sensors for both imputation methods, we can compare it against  $R(f)$  and  $P(f)$ . In particular, we define the difference in recall and precision of the two imputation methods with respect to the original recall and precision values as:

$$\Delta_R(x, f) = R(f) - \frac{\sum_{s \in S_C} R_x(f, s)}{|S_C|} \quad (6)$$

and

$$\Delta_P(x, f) = P(f) - \frac{\sum_{s \in S_C} P_x(f, s)}{|S_C|}, \quad (7)$$

where  $x$  denotes *MA*, *MF* or *GAN*. Figures 6a and 6b show the resulting distribution of these differences. As we can observe, the GAN imputation results in a shift of the distribution of the difference in recall and difference in precision towards smaller values. In other words, the loss due to the missing sensor measurements is significantly mitigated by the adoption of the GAN-based imputation, with the more complex MF imputation performing better than the MA. Both MF and MA methods exhibit values of recall greater than 0.1, i.e. they have a significant impact on the fault classification performance. In fact, the difference in recall is greater than 0.1 in 59% of the cases when using MA and in 29% of the cases when using MF. As for the difference in precision, it is greater than 0.1 in 29%, 18%, and 6% of the cases when using MA, MF, and GAN respectively.

Figures 6c and 6d show the resulting distribution of the difference in recall and precision in case 2 critical sensors measurements are not available. In this case, for each of the 500 testing files of each fault we simulated the unavailability of each of the possible 2-combinations of the critical sensors (i.e.  $\binom{12}{2} = 66$ ). As before, we consider the worst case scenario of persistent sensors failures, i.e. the sensor measurements are unavailable for the duration of the simulated process. We then perform the imputation using the generator of the trained GAN, the MA, and the MF and tested the DNN on the resulting imputed data. The results confirm that the GAN imputation significantly outperforms the MA and MF also in this case.

As for the anomaly detection evaluation, we performed an analysis similar to that conducted for the fault-classification DNN. Table IV shows the comparison between the false alarm and the missed detection rate for the original testing files, for imputed values using MA, MF and GAN. It is worth highlighting that to compute the missed detection rate the testing files corresponding to faulty operations have to be

TABLE III: Number of samples used for false alarm and miss detection rate computation.

	No missing data	MA/MF/GAN
False alarm rate	$500 \times 960$	$12 \times 500 \times 960$
Miss detection rate	$17 \times 500 \times 800$	$12 \times 17 \times 500 \times 800$

TABLE IV: False alarm and miss detection rate on the test set.

	No missing data	MA	MF	GAN
False alarm rate	0.12	0.17	0.14	0.14
Miss detection rate	0.17	0.17	0.18	0.17

used, while the false alarm rate is computed on the testing files corresponding to normal operations. As before, for each of the 500 testing files of each of the 17 faults, used in this case to compute the missed detection rate, we simulated the unavailability of each of the 12 critical sensors 10 time slots after the fault occurred and for the entire duration of the simulated process<sup>2</sup>. In the case of false alarm rate, for each of the 500 testing files corresponding normal operations, we simulated the unavailability of each of the 12 critical sensors 10 time slots after the beginning of the simulation. Table III summarizes the number of samples used for each case. As we can see from Table IV, the MA and GAN methods perform as well as the original data with respect to the miss detection rate, while the MF method shows degradation in its performance. However, the false alarm rate is affected by the use of imputed values, with the GAN and MF outperforming the MA method.

## V. CONCLUSION

In this work we proposed a data-driven decomposition of the process to monitor the various indicators of the health of a machine/component or an entire industrial process. We included in the monitoring system a module for data imputation that guarantees we have a functioning system in place even in case some of the critical sensors' measurements are missing, due for example to hardware or network issues. The data imputation module is based on GAN and was optimized by taking into account the feedback from the fault detection and classification modules, rather than using a metric, e.g. the RMSE, specific to the GAN model alone. This allowed us to fine-tune the data imputation so as to minimize the impact of missing sensor data on the capability of the system to detect and identify faults. We conducted a thorough evaluation of the proposed approach using the extended Tennessee Eastman Process dataset. Results show that the GAN-imputed data mitigate the impact on the fault detection and identification even in the case of persistently missing measurements from sensors that are critical for the correct functioning of the monitoring system.

## ACKNOWLEDGMENT

This work is supported by CHIST-ERA (call 2017) via the FIREMAN consortium, which is funded by the following

<sup>2</sup>Since 20 measurements per hour are collected and faults are introduced in each testing run after 8 simulation hours, the number of faulty samples in each testing file is  $960 - 8 \times 20 = 800$ .

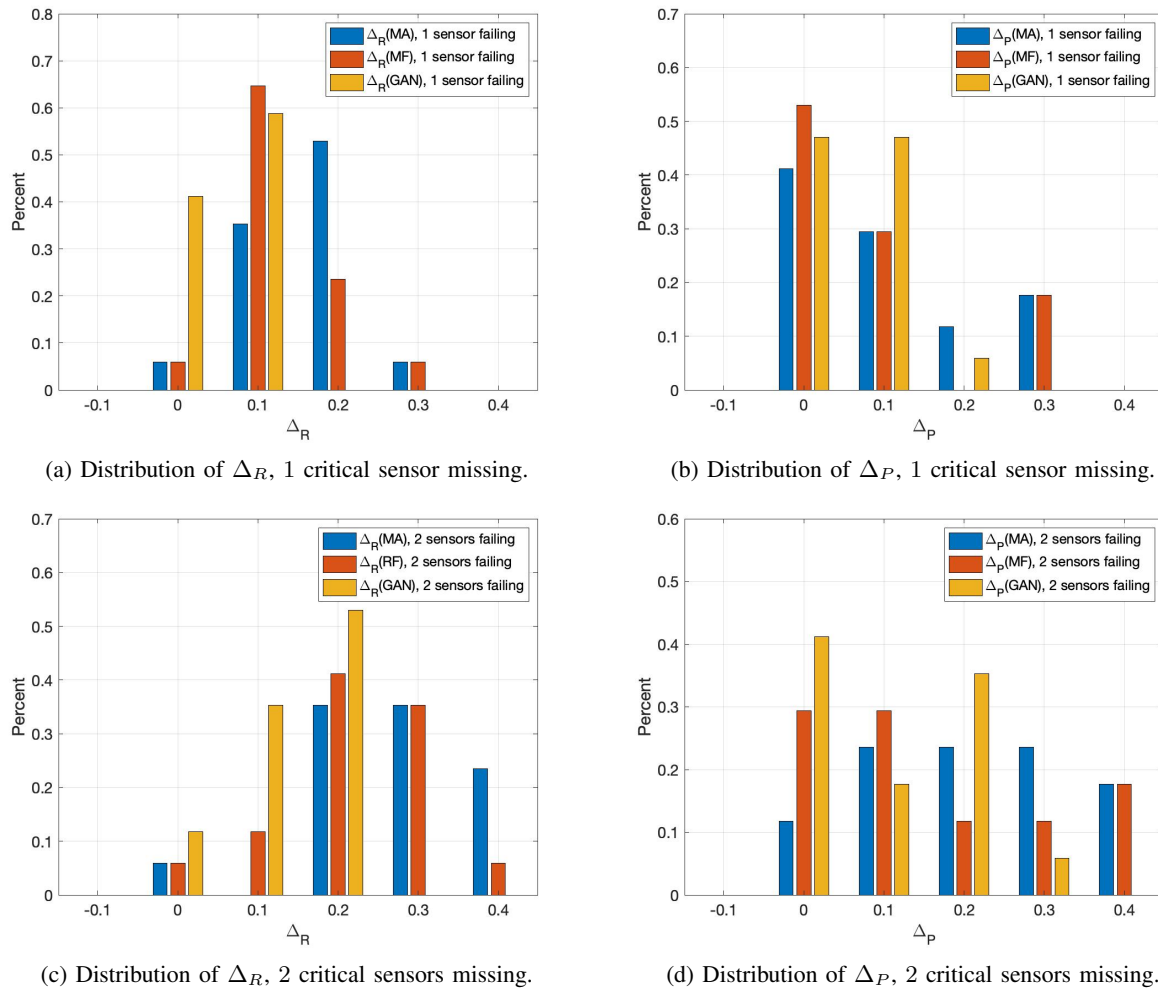


Fig. 6: Performance comparison on the test data.

national foundations: Academy of Finland (n. 326270, n. 326301), Irish Research Council, Spanish and Catalan Government under grants TEC2017-87456-P and 2017-SGR-891, respectively. This material is also supported by the Air force Office of Scientific Research under award number FA9550-18-1-0214, and co-funded by Science Foundation Ireland (SFI) under the European Regional Development Fund with Grant Numbers 13/RC/2077 and 13/RC/2077\_P2.

## REFERENCES

- [1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE transactions on industrial informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.
- [2] C. Kuhlins, B. Rathonyi, A. Zaidi, and M. Hogan, "Cellular networks for massive IoT," <https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-networks-for-massive-iot--enabling-low-power-wide-area-applications>, 2020.
- [3] Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa, "Missing value imputation for industrial iot sensor data with large gaps," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6855–6867, 2020.
- [4] F. Civerchia, S. Bocchino, C. Salvadori, E. Rossi, L. Maggiani, and M. Petracca, "Industrial internet of things monitoring solution for advanced predictive maintenance applications," *Journal of Industrial Information Integration*, vol. 7, pp. 4–12, 2017.
- [5] J. Wan, S. Tang, D. Li, S. Wang, C. Liu, H. Abbas, and A. Vasilakos, "A manufacturing big data solution for active preventive maintenance," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2039–2047, 2017.
- [6] B. Cheng, J. Zhang, G. Hancke, S. Karnouskos, and A. Colombo, "Industrial cyberphysical systems: Realizing cloud-based big data infrastructures," *IEEE Industrial Electronics Magazine*, vol. 12, no. 1, pp. 25–35, 2018.
- [7] W. Yu, T. Dillon, F. Mostafa, W. Rahayu, and Y. Liu, "A global manufacturing big data ecosystem for fault detection in predictive maintenance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 183–192, 2020.
- [8] P. Nardelli, C. Papadias, C. Kalalas, H. Alves, I. T. Christou, I. Macaluso, N. Marchetti, R. Palacios, and J. Alonso-Zarate, "Framework for the identification of rare events via machine learning and iot networks," in *2019 16th International Symposium on Wireless Communication Systems (ISWCS)*, 2019, pp. 656–660.
- [9] C. Kalalas and J. Alonso-Zarate, "Sensor data reconstruction in industrial environments with cellular connectivity," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–6.
- [10] P. Mulinka, C. Kalalas, M. Dzaferagic, I. Macaluso, D. Gutierrez, Rojas, P. Nardelli, and N. Marchetti, "Information processing and data visualization in networked industrial systems," in *Accepted: IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2021.
- [11] A. González-Vidal, P. Rathore, A. S. Rao, J. Mendoza-Bernal, M. Palaniswami, and A. F. Skarmeta-Gómez, "Missing data imputation with bayesian maximum entropy for internet of things applications," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

- [12] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018. [Online]. Available: <https://doi.org/10.1016/j.jisa.2017.11.002>
- [13] X. Jia, Q. Feng, T. Fan, and Q. Lei, "RFID technology and its applications in Internet of Things (IoT)," *2012 2nd International Conference on Consumer Electronics, Communications and Networks, CECNet 2012 - Proceedings*, pp. 1282–1285, 2012.
- [14] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Computer Communications*, vol. 89–90, pp. 5–16, 2016. [Online]. Available: <http://dx.doi.org/10.1016/j.comcom.2016.03.015>
- [15] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
- [16] B. Krawczyk, "Learning from imbalanced data: open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.
- [17] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart agents in industrial cyber-physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1086–1101, 2016.
- [18] S. J. Oks, A. Fritzsche, and K. M. Möslin, "An application map for industrial cyber-physical systems," in *Industrial internet of things*. Springer, 2017, pp. 21–46.
- [19] S. Yin, J. J. Rodriguez-Andina, and Y. Jiang, "Real-time monitoring and control of industrial cyberphysical systems: With integrated plant-wide monitoring and control framework," *IEEE Industrial Electronics Magazine*, vol. 13, no. 4, pp. 38–47, 2019.
- [20] W. Dai, H. Nishi, V. Vyatkin, V. Huang, Y. Shi, and X. Guan, "Industrial edge computing: Enabling embedded intelligence," *IEEE Industrial Electronics Magazine*, vol. 13, no. 4, pp. 48–56, 2019.
- [21] H. Hellstrom, M. Luvisotto, R. Jansson, and Z. Pang, "Software-defined wireless communication for industrial control: A realistic approach," *IEEE Industrial Electronics Magazine*, vol. 13, no. 4, pp. 31–37, 2019.
- [22] L. H. Chiang, R. D. Braatz, and E. L. Russel, *Fault detection and diagnosis in industrial systems*. Springer, 2001.
- [23] S. Yin and O. Kaynak, "Big data for modern industry: challenges and trends [point of view]," *Proceedings of the IEEE*, vol. 103, no. 2, pp. 143–146, 2015.
- [24] D. Zurita, M. Delgado, J. A. Carino, and J. A. Ortega, "Multimodal forecasting methodology applied to industrial process monitoring," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 494–503, 2017.
- [25] T. Chen, X. Liu, B. Xia, W. Wang, and Y. Lai, "Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder," *IEEE Access*, vol. 8, pp. 47 072–47 081, 2020.
- [26] P. Hu and J. Zhang, "5g-enabled fault detection and diagnostics: How do we achieve efficiency?" *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3267–3281, 2020.
- [27] C. Kalalas and J. Alonso-Zarate, "Sensor data reconstruction in industrial environments with cellular connectivity," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, 2020, pp. 1–6.
- [28] S. Moritz and T. Bartz-Beielstein, "imputeTS: Time Series Missing Value Imputation in R," *The R Journal*, vol. 9, no. 1, pp. 207–218, 2017. [Online]. Available: <https://doi.org/10.32614/RJ-2017-009>
- [29] Y. Li and L. E. Parker, "A spatial-temporal imputation technique for classification with missing data in a wireless sensor network," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3272–3279.
- [30] D. J. Stekhoven and P. Bühlmann, "Missforest—non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 2012.
- [31] J. Yoon, J. Jordon, and M. Schaar, "Gain: Missing data imputation using generative adversarial nets," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5689–5698.
- [32] C. Rieth, B. Amsel, R. Tran, and M. Cook, "Additional tennessee eastman process simulation data for anomaly detection evaluation," *Harvard Dataverse*, vol. 1, 2017.
- [33] Y. Zhang, "Enhanced statistical analysis of nonlinear processes using kpca, kica and svm," *Chemical Engineering Science*, vol. 64, no. 5, pp. 801–811, 2009.
- [34] F. Lv, C. Wen, Z. Bao, and M. Liu, "Fault diagnosis based on deep learning," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 6851–6856.
- [35] Z. Chai and C. Zhao, "Enhanced random forest with concurrent analysis of static and dynamic nodes for industrial fault classification," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 54–66, 2019.