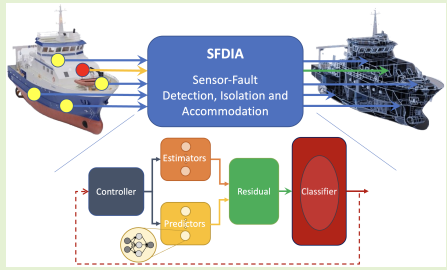
A Machine-Learning Architecture for Sensor Fault Detection, Isolation and Accommodation in Digital Twins

Hossein Darvishi, *Student Member, IEEE*, Domenico Ciuonzo, *Senior Member, IEEE*, and Pierluigi Salvo Rossi, *Senior Member, IEEE*

***Abstract*— Sensor technologies empower Industry 4.0 by enabling integration of in-field and real-time raw data into digital twins. However, sensors might be unreliable due to inherent issues and/or environmental conditions. This paper aims at detecting anomalies instantaneously in measurements from sensors, identifying the faulty ones and accommodating them with appropriate estimated data, thus paving the way to reliable digital twins. More specifically, a real-time general machine-learning-based architecture for sensor validation is proposed, built upon a series of neural-network estimators and a classifier. Estimators correspond to virtual sensors of all unreliable sensors (to reconstruct normal behaviour and replace the isolated faulty sensor within the system), whereas the classifier is used for detection and isolation tasks. A comprehensive statistical analysis on three different real-world data-sets is conducted and the performance of the proposed architecture is validated under hard and soft synthetically-generated faults.**



***Index Terms*— Digital twin, Fault diagnosis, Machine learning, Neural networks, Sensor validation.**

I. Introduction

DIGITAL TWINS (DTs) have recently emerged in several industrial applications and exploit Internet of Things (IoT) technology [1]. More specifically, most environments have been pervaded by the extensive use of spatially-distributed sensors, generating enormous amount of heterogeneous data over time, which requires advanced integrated solutions involving sensing, communication, and processing [2]–[4]. DTs represent one of the main products for building advanced analytics over such data and extract relevant information for prediction and effective control. DTs have been widely employed in various sectors such as industry [5], health care [6] and smart cities [7], [8], where their capabilities to visualize and treat with a perpetual stream of real-time sensor data is enabling new opportunities. Leveraging sensor data enables DTs to model system dynamics effectively for remote monitoring and controlling, for safety and risk analysis and for maintenance purposes. Since DTs rely on accurate sensor data, system performance may be affected severely by sensor failures. Sources of sensor faults are commonly found in: (*i*) *Hardware and inherited limitations* - sensors are electronic components and can collect inaccurate measurements or stop working without any indication due to low production quality, calibration issues, low battery level, end of life span, poor connections [9]; (*ii*) *Harsh environment* - in real-world scenarios, sensors can be deployed in inaccessible and unattended environments with possibility of unlikely situations which would hinder sensors performance [10]; (*iii*) *Malicious attacks* - faulty data can be injected by an attacker into a vulnerable system [11], [12].

A fault in a system refers to a complete (or partial) malfunction and manifests over a permanent (or transient) time span. As shown in Fig. 1, the most common types of sensor faults in a sensor network are defined (a detailed discussion of sensor faults is found in [13], [14]). Depending on the characteristics of sensor data, faults can be classified as following:

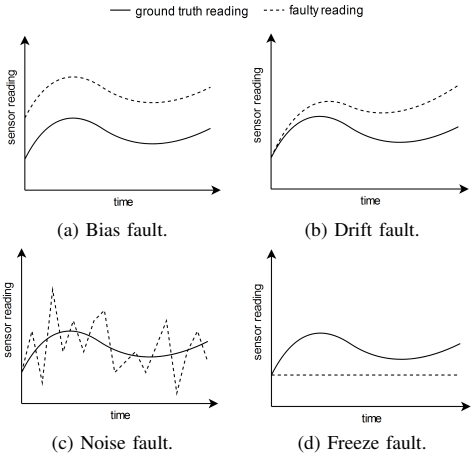
1) *Bias* fault: also known as offset fault, the deviation from nominal values is given by an additive constant bias;
2) *Drift* fault: sensor readings drift with a small slope from nominal values (drift faults are more subtle since they

This work was partially supported by the Research Council of Norway under the project SIGNIFY within the IKTPLUSS framework (project nr. 311902). Part of this work was presented at the IEEE International Conference on Networking, Sensing and Control (ICNSC) 2021.

H. Darvishi is with the Department of Electronic Systems, Norwegian University of Science and Technology, 7491 Trondheim, Norway, and with the Signal Processing Laboratory (LTS4), École polytechnique fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland (e-mail: hossein.darvishi@ntnu.no).

D. Ciuonzo is with the Department of Electrical Engineering and Information Technologies (DIETI), University of Naples "Federico II", 80125 Naples, Italy (e-mail: domenico.ciuonzo@unina.it).

P. Salvo Rossi is with the Department of Electronic Systems, Norwegian University of Science and Technology, 7491 Trondheim, Norway, and with the Department of Gas Technology, SINTEF Energy Research, 7491 Trondheim, Norway (e-mail: salvorossi@ieee.org).



(a) Bias fault. (b) Drift fault.

(c) Noise fault. (d) Freeze fault.

Fig. 1: Types of sensor faults.

appear gradually over time and their effect is not very apparent);

3) *Noise* fault: an increased noise level in sensor readings (when noise power is much larger than usual, it is an indication of sensor malfunctioning).
4) *Freeze* fault: also known as stuck-at fault, the sensor readings stuck at a constant value (i.e. the variance of the readings becomes zero);

The impact of sensor faults would affect stability, reliability and accuracy of the system depending on the specific application. Hence, to fully utilize the expected properties of the DT, it is essential to continuously evaluate and amend sensor data. From this perspective, prompt **Sensor Fault Detection, Isolation and Accommodation (SFDIA)** is one key issue for deploying DTs while assuring reliable performance. SFDIA indeed consists of *three* parts:

- *fault detection*, i.e. determining sensor fault(s) within the system's sensor network;
- *fault isolation*, i.e. identifying specific faulty sensors and block their measurement feeding to DT;
- *fault accommodation*, i.e. feeding DT with some other replaced trustworthy data.

In what follows, related literature is reviewed by focusing on recent progress on sensor fault diagnosis and SFDIA approaches. It is worth highlighting that the following discussion leaves out the (huge) corpus of literature dealing with soft/virtual sensor design (see, for instance, the excellent survey [15]). Indeed, it should be noted the latter field is out of the scope of this paper, as soft/virtual sensors are usually meant to provide predictions only for analyzing, monitoring and/or controlling purposes (corresponding to the first layer of the proposed SFDIA architecture). Also, we emphasize that this work focuses on sensor faults only, i.e. the monitored physical process does not exhibit any anomaly while the measurement data do (e.g. errors in data acquisition and/or communication). Process fault detection and related analysis is beyond the scope of this work.

A. Related Work

In the last years, the main advancements in fault diagnosis technology have relied on the milestone concept of *redundancy* which embraces a wide spectrum of design solutions, e.g. redundancy can be accomplished by either *hardware* or *analytical* schemes. Within the class of *hardware-based approaches* (also referred to as physical-based approaches), multiple identical sensors (i.e. sensing the same physical parameter) along with a voting scheme (or more sophisticated techniques, see [16]) are employed to detect, isolate and accommodate sensor failures [17]–[19]. If the difference (namely, the residual signal) between the measured signal of a sensor and each other sensor in the set is considerably high, the aforementioned sensor is declared faulty and its data is replaced with those from the remaining (identical) sensors. For instance, the aforementioned assumptions apply to the case of homogeneous WSNs, where neighboring nodes are assumed to measure roughly the same parameter [16]. Conventional physical-redundancy approaches however cannot handle cases with simultaneous failures of identical sensors, as they do not capitalize the statistical dependence of measurements originating from other sensor types [17], [18]. Moreover, in many applications, it is impractical to implement these approaches due to space and/or weight and/or cost constraints [18].

Accordingly, it is not surprising that methods adopting *analytical redundancy* have gained increasing attention within the research on SFDIA [20]–[22]. Unlike physical redundancy, the latter approaches exploit correlations and functional relationships within the system instead of introducing additional (redundant) hardware. Still, it is worth highlighting that the above two philosophies *are not* mutually exclusive and hybrid approaches can be pursued toward the sophisticated design of fault-tolerant DTs. Analytical redundancy can be usually implemented by either *model-based* or *data-driven* techniques.

Model-based SFDIA have been mostly investigated in the context of power systems [23], e.g. using electrical dynamics equations [20] or Luenberger observers [24]. Some other methods have focused on the detection and accommodation of proportional-type faults in nonlinear systems [25], [26]. Unfortunately, those methods (a) usually result in high complexity, (b) require an explicit, application-dependent, formulation of the analytical redundancy relationship among sensors and (c) are seldom able to handle multiple sensor faults simultaneously. On the contrary, *data-driven* approaches relying on historical data have recently received large interest, starting from simpler methods (e.g. auto-regressive models with exogenous inputs (ARX) [27]) to more complicated (non-linear) learning approaches (e.g. random forest (RF) [28], support vector machines (SVMs) [29], [30] and NNs [31], [32]). Indeed, *data-driven* techniques *do not require exact knowledge* of the mathematical model for sensor fault diagnosis.

Specifically, SVM-based classification was one of the relevant attempts to *detect* sensor faults in WSNs, in both batch [29] and online forms [30], which showed relatively small computational costs, but limited performance. Successive works [33], [34] have also employed the SVM approach to allow *both* detection and identification of faults: a binary

classifier was trained from the residuals of each sensor. Specifically, in the former case [33], the residual signals were generated by comparing the true measurements with a single (global) observer designed by including fault models. Conversely, in the latter case [34], a residual was obtained from each (correlated) sensor pair via an ARX model, thus providing multiple classification outputs for a given sensor then aggregated at a higher level.

A second important class of approaches for SFDIA relies on the well-known Autoencoder (AE) NN [12], [21], [35], [36]. Indeed, the AE is an unsupervised learning technique capable of learning and extracting hidden representations from raw data and it is thus suited for fault detection. Hence, once trained, the AE can provide a reconstructed estimate of the sensors' measurements, thus allowing straightforward computation of residuals (i.e. the difference between inputs and outputs of the AE). Specifically, an AE-based (aided by exogenous inputs) sensor validation scheme for a heating, ventilation and air conditioning system was proposed in NNs [36]. Detection and identification are simply performed by comparing overall and per-sensor residuals to a given threshold. A similar AE-based SFDIA method is presented in [21] for an air quality controlling system, with identification scheme performed via a more involved sensor validity index. In both works [21], [36] accommodation is simply performed by using the AE output associated to the sensor(s) declared as faulty. Differently, a more sophisticated proposal uses an additional denoising AE (a supervised learning technique) to perform the accommodation task [12], namely to clean faulty data. Despite their simplicity, AE-based SFDIA approaches can suffer however from degraded performance under weak-faults, as the latter type of faults does not considerably impact correlations in data.

Multi-layer perceptron (MLP) NNs (including variants) have also been proved to perform satisfactorily for a number of relevant sensor fault diagnosis tasks [22], [37], [38], including heavy-duty diesel engines' and aircrafts, based on a *sensor-centric viewpoint*. Indeed, in all the aforementioned works, *one MLP estimator per each sensor* is designed (solely on the basis of other sensors' measurements) and detection/identification is based on the evaluation of the residual vector. Accommodation is then performed by using the estimator(s) associated to the sensors declared as faulty. Specifically, the proposal in [37] adopts fully-connected cascade NNs (i.e. MLPs allowing direct connections across different hidden layers) for the sensor estimator design, while [22] considers a hybrid structure with a linear NN and resource allocation network (a variant of well-known radial basis function NN) for the same task. More recently, a plain MLP estimator (exploiting the sole spatial correlation among sensors) has been proven to provide reliable detection with low false-alarm rate as well [38].

A different rationale is pursued in [31], where a *single* Deep belief network (a Bayesian type of NNs) has been trained (in a supervised fashion) to detect a faulty condition whereas sensor identification is naively carried out based on the maximum deviation from data mean-value. Along the same lines, a general approach is presented to detect and identify sensor faults using either a single Recurrent NN (RNN) or an MLP [39] for predicting next-step measurements and comparing with actual ones. A disentanglement regularization term on the NN loss function is introduced to help the algorithm coping with propagation of faults to non-faulty sensors in the identification stage. Unfortunately, the accommodation stage is not taken into account in the above work. Interestingly, also a dynamic Bayesian network has succeeded in sensor fault detection and accommodation exploiting spatial and temporal correlations in the context of intelligent connected vehicles [40]. Still, its training difficulty (in terms of both parameter and structure learning) appears limiting in large-scale sensor systems.

Recently, the sensor-centric viewpoint in [22], [37], [38] has further been exploited to devise a *modular* SFDIA (M-SFDIA) method based on MLP NNs in [32], [41], with focus on supporting DTs. The proposed structure consists of a set of estimators (each associated to a sensor) providing residual signals as well as replacements (estimates) for faulty data. Therein a supervised classifier is trained to make detection & identification decisions upon the residual signals by leveraging their (possibly-nonlinear) relationships. An experimental analysis on three real-world data-sets has demonstrated satisfactory performance of M-SFDIA method. Although promising (from the estimators' design viewpoint), M-SFDIA architecture does not completely exploit the temporal correlations among sensors within the monitored system.

B. Paper Contribution

In view of the previous discussion, some proposals are restricted to a given vertical domain (e.g. aircraft [37], vehicle [34] or HVAC system [36] monitoring), thus *lacking a general formulation*. Secondly, part of the literature evaluates corresponding proposals on *private* (e.g. [39], [40]) or *simulated* (e.g. [28], [36], [37]) measurement data, thus precluding reproducibility and convincing evaluation, respectively. Thirdly, a number of the discussed works evaluate their proposals only on a *single fault type* (e.g. bias [21], [39] or drift [22]). Equally important, some architectures are only *limited to fault detection* [29], [30]. On the other hand, some recent proposals do not foresee all the three tasks in their original formulation, e.g. the identification and accommodation tasks in [12] and [39], respectively. Still, even when all three tasks can be carried out, in some cases *only spatial correlation* [22], [36], [38] is used to accommodate faulty measurements. Finally, some approaches have a *limited modularity* [12], [21], [39]. Accordingly, the *main contributions* of this article are summarized as follows:

- A *real-time* and *modular* data-driven SFDIA architecture is developed, fully exploring (viz. learning) *spatial* and *temporal* dependence in sensory data. The proposed architecture relies on *the novel use of a pair of regressors* for each sensor, performing *estimation* and *prediction* operations, respectively. In the former case, each estimator is leveraging readings from other sensors only to obtain a "virtual measurement". Conversely, each predictor plays a complementary role (to the estimator) by using only previous data from the sensor under consideration to obtain an analogous virtual measurement. Hence, their joint

adoption enables the proposed architecture to ultimately exploit spatio-temporal correlation within the system, thus supporting nearly-instantaneous fault detection and isolation performance.

- The dissimilarity measured by predictors (resp. estimators) and measurements, referred to as *residual signals*, are then used as the perfect candidate for designing a reliable classifier able to perform both *fault detection* (i.e. whether there is a fault in the whole sensor set) and *identification* (i.e. which sensors are faulty).
- The proposed approach employs MLP NNs for both regression (estimation and prediction) and classification modules to capture and process analytical redundancy relations while keeping a *reasonable complexity* at the operational stage. In the latter case, a *multi-task* MLP NN (i.e. each sensor condition is seen as a binary classification task) is designed for detecting and (if any) identifying multiple faulty sensors via a *single* neural network.
- Moreover, classifier decisions, residual signals and virtual measurements are exploited by a *a specifically-designed controller* to make corrections on sensor models inputs and improve overall system performance both for detection and isolation tasks. Specifically, in a feedback loop, the controller is in charge of replacing corrupted input data and, consequently, avoiding propagation of faults throughout the architecture.
- The performance of the proposed SFDIA architecture is assessed on *three real-world (public) data-sets* [42]–[44] which are corrupted with (a) four relevant fault types (bias, drift, noise and freeze) and (b) different levels of faults (with special emphasis on weak faults, as they are more difficult to detect).
- The proposal is compared with two state-of-the-art machine-learning-based architectures [12], [41] from both performance (in terms of detection delay and probabilities of detection, false alarm, and correct identification, and accommodation error) and computational complexity (in terms of number of trainable parameters) standpoints.

The present work extends earlier conference paper [45], which (a) presented only an intermediate version of the proposed novel architecture (no controller block), (b) reported a significantly-smaller experimental analysis (focusing only on the WSN data-set [43]), (c) considered a smaller set of baselines in the comparison and (d) assessed the effectiveness of the SFDIA approach only on bias faults.

The remainder of this paper is structured as follows: in Sec. II, the proposed data-driven SFDIA architecture is presented and the functionalities of each block are illustrated; Sec. III describes the configuration of the NNs and the related training process; the description of the data-sets and the framework for fault generation are provided in Sec. IV; Sec. V presents and discusses the numerical performance of the proposed architecture in contrast with benchmarks from the current literature. Finally, concluding remarks and future directions of research are given in Sec. VI.

Notation - Lower-case bold letters indicate vectors; $\boldsymbol{I}_N$ denotes the null column vector of length N; $(\cdot)^T$ refers to the transpose operator, $\in$ is the set membership, and $\mathcal{O}(\cdot)$ denotes the Landau notation.

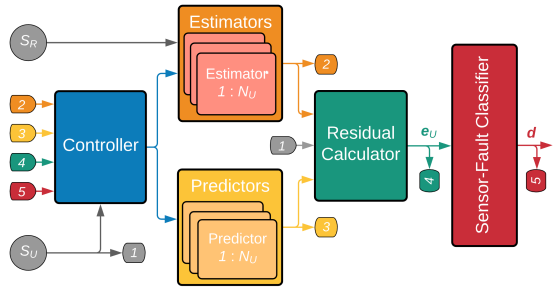


Fig. 2: Block diagram of the proposed SFDIA architecture.

II. SFDIA

The proposed method aims to exploit the full potential of *spatial* and *temporal* correlation among sensors in a system. Specifically, it is assumed that the sensors are divided into *two* sets: (i) the set of unreliable sensors $\mathcal{S}_U$, containing sensors that are vulnerable to faults; and (ii) the set of reliable sensors $\mathcal{S}_R$, which, depending on the working system, include sensors whose flawless functionality can be guaranteed [41]. This (ideal) level of reliability could be associated to: a meta-sensor modeling a group of identical sensors (enjoying hardware redundancy), high-quality sensors, a proper design and safe working environment, a device being at the middle of life span [46], or context measurement information which is assumed to have significantly higher reliability than the considered networked sensor system. In a more general sense, any reliable source of data correlated with the unreliable sensors could be included in the set of reliable sensors. In the following, without loss of generality, it is assumed $\mathcal{S}_U = \{1, \ldots, N_U\}$ and $\mathcal{S}_R = \{N_U + 1, \ldots, N\}$, where N_U and N denote the number of unreliable sensors and total number of sensors, respectively. Also, for compactness, N_R denotes the cardinality of the reliable set $\mathcal{S}_R$ (i.e. $N_R = N - N_U$).

A. Architectural Overview

The block diagram of the proposed SFDIA architecture is shown in Fig. 2. It consists of five building blocks (controller, estimators, predictors, residual calculator, classifier) arranged in *four layers*, whose function is explained as follows. The *first layer* contains two parallel blocks, the estimators block and the predictors block, each providing a virtual measurement for all the unreliable sensors in the system either regressed via other sensors' observations (i.e. the estimator) or based only on previous measurements of the same sensor under consideration (i.e. the predictor). The *second layer* is responsible for the computation of a discrepancy measure between the true and each calculated virtual measurement, usually in the form of a function of the residual signals. The *third layer* is fed with the aforementioned discrepancy measures and is able to perform a multidimensional classification to (a) detect a faulty condition and (b) identify the corresponding faulty sensors. Finally, the

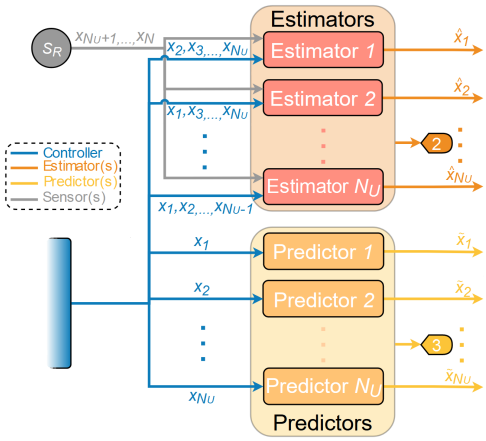


Fig. 3: Diagram detailing the estimators and predictors blocks.

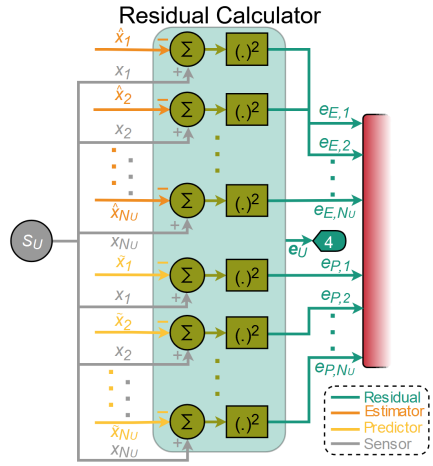


Fig. 4: Diagram detailing the residual calculator block.

fourth layer controls the inputs of the blocks in the first layer in order to preserve estimators and predictors accuracy, by avoiding error propagation.

The present architecture improves over the one proposed in [41] where the main novelty is the introduction of the controller and the predictors. Despite the addition of these two modules, it is worth remarking that the proposed architecture retains the advantages of *modularity* and *real-time implementation*. Indeed, regarding the former property, the proposed approach allows the implementation of diversified ML techniques for different modules and a more flexible deployment, also taking computational/hardware limitations into account. Differently, regarding the latter property, each of the proposed modules can be flawlessly implemented in real-time since they are all based on a sliding window implementation. Finally, given the adoption of MLP-based solutions for the estimators/predictors (Sec. II-B) and the classifier (Sec. II-D), the proposed implementation also retains simplicity. The following subsections detail each of the four layers constituting the proposed approach.

B. First Layer: Estimation & Prediction

The first layer aims to model the unreliable sensors within the system and is based on *two subsystems*: (a) a bank of estimators and (b) a bank of predictors.

As detailed in Fig. 3, the bank of **estimators** is composed of N_U estimators (each associated to an unreliable sensor), each providing the estimation $\hat{x}_s[n]$ of the measurement (at current time step n) from its corresponding unreliable sensor $s \in \mathcal{S}_U$. Each estimator receives as input the vector $\boldsymbol{x}_{(s)}$ collecting all existing sensors readings (from current time step n back to L_e previous time steps using a sliding window mechanism) except the one from the sensor to be estimated $\{\mathcal{S}_U \cup \mathcal{S}_R - s\}$, i.e.

$$\hat{x}_s[n] = f_s^{(H_v,N_v)}(\boldsymbol{x}_{(s)}[n],\ldots,\boldsymbol{x}_{(s)}[n-L_e])\ , \qquad (1)$$

where $f_s^{(H_v,N_v)}(\cdot)$ denotes the function model of the MLP-based estimator for the sth sensor, being H_v and N_v the number of hidden layers and the number of neurons, respectively. Previous time samples are fed into the estimators in order to exploit the *temporal correlation* among the input signals.

The bank of **predictors** operates a complementary approach. Each of the N_U predictors provides a prediction $\tilde{x}_s[n]$ of the measurement (at current time step n) from its corresponding unreliable sensor $s \in \mathcal{S}_U$. Each predictor receives as input the readings $x_s[\cdot]$ of the sensor to be predicted (from previous time step $n-1$ back to L_p previous time steps using a sliding-window mechanism), i.e.

$$\tilde{x}_s[n] = g_s^{(H_v,N_v)}(x_s[n-1],\ldots,x_s[n-L_p])\ , \qquad (2)$$

where $g_s^{(H_v,N_v)}(\cdot)$ denotes the function model of the MLP-based predictor for the sth sensor, again being H_v and N_v the number of hidden layers and the number of neurons, respectively.

C. Second Layer: Residual Evaluation

The second layer computes the square of residual signals i.e. the difference of sensors reading with their respective estimation or prediction values (see Fig. 4), namely

$$e_{E,s}[n] = (x_s[n] - \hat{x}_s[n])^2, \qquad (3)$$
$$e_{P,s}[n] = (x_s[n] - \tilde{x}_s[n])^2, \qquad (4)$$

for each unreliable sensor $s \in \mathcal{S}_U$. Residual signals are used as input to the classifier in the third layer as they contain effective information for fault classification. It is worth noticing that the proposed SFDIA architecture enjoys modularity and generality: thus other discrepancy measures (other than that used in Eqs. (3) and (4)) may be adopted *without any substantial change* in the subsequent layers.

D. Third Layer: Classification

An MLP **classifier**, meant to work in real-time, is used for fault *detection* and the *identification* of the faulty sensors, and its detailed structure is shown in Fig. 5. Denoting

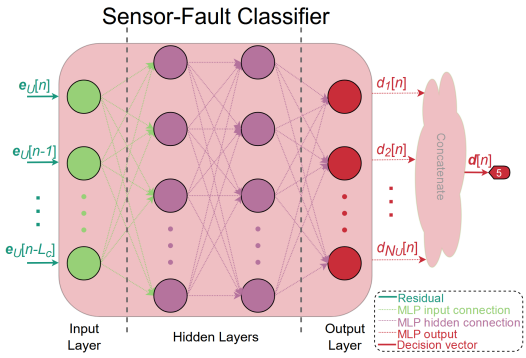


Fig. 5: Structure of the MLP-based classifier block.

$\boldsymbol{e}_U[n] = (e_{E,1}[n], \ldots, e_{E,N_U}[n], e_{P,1}[n], \ldots, e_{P,N_U}[n])^T$ the residual vector containing the residual signals of all N_U sensors at time step n, the input of the classifier is the collection of residual vectors from L_c previous time steps up to current time step n, namely $\boldsymbol{e}_U[n], \ldots, \boldsymbol{e}_U[n-L_c]$. Conversely, a decision vector $\boldsymbol{d}[n] = (d_1[n], d_2[n], \ldots, d_{N_U}[n])^T$ represents the output of the classifier and identifies which among the unreliable sensors are suspected to be in failure, i.e.

$$\boldsymbol{d}[n] = \boldsymbol{h}^{(H_c,N_c)}(\boldsymbol{e}_U[n], \ldots, \boldsymbol{e}_U[n-L_c]), \quad (5)$$

where $\boldsymbol{h}^{(H_c,N_c)}(\cdot)$ denotes the function model of the MLP-based classifier, being H_c and N_c are the number of hidden layers and the number of neurons of the classifier, respectively. More specifically, the sth entry of the decision vector, i.e. $d_s[n] \in [0,1],\ s = 1, \ldots, N_U$, represents a pseudo-probability for the sth unreliable sensor to be faulty. Apparently, $d_s[n] = 1$ (resp. $d_s[n] = 0$) represents the situation in which the system declares with maximum confidence the sth sensor to be faulty (resp. fault-free). As a consequence, a vector $\boldsymbol{d}[n] = \boldsymbol{0}_{N_U}$ indicates healthy operation of *all* the sensors within the system at time n.

Therefore, faulty sensors are identified via a threshold-based logic for each of the components of the decision vector. The considered threshold will be denoted γ in what follows. Principally, herein faulty sensors are detected and identified/isolated when the entries of the decision vector $\boldsymbol{d}[n]$ exceed the threshold γ. Specifically, $\max_{s=1}^{N_U} d_s[n] \gtrless \gamma$ is used for detection. Accordingly, for the identification task, the set of identified faulty sensors (denoted with $\mathcal{I}_U$) is obtained as $\mathcal{I}_U \triangleq \{s \in \mathcal{S}_U : d_s[n] > \gamma\}$.

It is worth mentioning that, from overall SFDIA system perspective, the measurements from the sensors declared faulty are *replaced* (viz. *accommodated*) with their corresponding *estimates* in order to preserve system utility.

E. Fourth Layer: Control

The role of the control block is to preserve the performance of the proposed SFDIA method when faults occur. Referring to Fig. 2, this block operates at the beginning of each time step, and controls inputs-outputs of both estimators and predictors regarding the latest residual signals and the decision vector $\boldsymbol{d}[n-1]$.

The symbol $\phi_{E,s}$ (resp. $\phi_{P,s}$) denotes the average residual signal for the sth estimator (resp. predictor) computed with a moving average over a window of size L_r starting from time step $n-1$ while *excluding* the identified faulty time steps. The signal $\phi_{E,s}$ (resp. $\phi_{P,s}$) of the unreliable sensor s is used by the controller as a metric to define the estimation (resp. prediction) accuracy of the corresponding estimator (resp. predictor).

In the first step, after applying the proposed SFDIA scheme at time step $(n-1)$, the elements of the decision vector $\boldsymbol{d}[n-1]$ larger than a predefined threshold υ identify faulty sensors for the controller. Then, the following process will be conducted at the beginning of each time step n. To keep the discussion simple, we will generically refer to sth sensor as the one identified as faulty.

As for the **predictor controlling** scheme, if the estimator's average residual signal $\phi_{E,s}$ is smaller than a certain value τ (i.e. the system tolerable level of deviation), the estimator output $\hat{x}_s[n-1]$ *replaces* the respective sensor input $x_s[n-1]$ to the corresponding predictor. In other words, the predictor in Eq. (2) will be then fed as:

$$\tilde{x}_s[n] = g_s^{(H_v,N_v)}(\underbrace{\hat{x}_s[n-1]}_{\text{replacement}}, \ldots, x_s[n-L_p]), \quad (6)$$

This logic is intended to use only those estimates whose quality is better than the faulty-data within the sth predictor.

As for the **estimator controlling** scheme, if the predictor's average residual signal $\phi_{P,s}$ smaller than both (i) the system tolerable level of deviation τ and (ii) $\phi_{E,s}$, the predictor output $\tilde{x}_s[n]$ is obtained and *replaces* the respective sensor input $x_s[n]$ (updates all estimators' input vectors except $\boldsymbol{x}_{(s)}[n]$) to the estimators. In other words, we have $\forall s_\star \in \mathcal{S}, s_\star \neq s$:

$$\hat{x}_{s_\star}[n] = f_{s_\star}^{(H_v,N_v)}(\underbrace{\tilde{\boldsymbol{x}}_{(s_\star)}[n]}_{\text{replacement}}, \ldots, \boldsymbol{x}_{(s_\star)}[n-L_e]), \quad (7)$$

where the vector $\tilde{\boldsymbol{x}}_{(s_\star)}[n]$ collects all existing sensors readings except for $s_\star$ and with sth reading being replaced by $\tilde{x}_s[n]$. Otherwise, if $\phi_{E,s}$ is smaller than the system tolerable level of deviation[1], the estimator output $\hat{x}_s[n]$ is obtained and replaces the respective sensor input $x_s[n]$ (updates all input vectors except $\boldsymbol{x}_{(s)}[n]$) to the estimators. Specifically, $\forall s_\star \in \mathcal{S}, s_\star \neq s$:

$$\hat{x}_{s_\star}[n] = f_{s_\star}^{(H_v,N_v)}(\underbrace{\hat{\boldsymbol{x}}_{(s_\star)}[n]}_{\text{replacement}}, \ldots, \boldsymbol{x}_{(s_\star)}[n-L_e]), \quad (8)$$

where the vector $\hat{\boldsymbol{x}}_{(s_\star)}[n]$ collects all existing sensors readings except for $s_\star$ and with sth reading being replaced by $\hat{x}_s[n]$. This logic is intended to replace the input faulty-data with estimates/predictions whose accuracy are better than the input faulty-data (i.e. $x_{(s)}[n]$) to all estimators (except the corresponding sensor s estimator). We highlight that, in all three cases, *no architectural modification* (i.e. varying input size for

[1]In other words, the corresponding estimator is providing better accuracy than the corresponding predictor, i.e. $\phi_{E,s} < \phi_{P,s}$ and $\phi_{E,s} < \tau$.

the estimators and predictors) *is required* for the blocks of the proposed SFDIA method.

Conversely, in the case of *no-fault detected*, this block merely slides the window forward in time to update both $\phi_{P,s}$ and $\phi_{E,s}$ by using the recent residual signals $\boldsymbol{e}_U[n-1]$.

A pseudo-code of the controlling block process is given in Algorithm 1. It is worth remarking that substitution of faulty inputs with either estimated or predicted values maintains estimators and predictors accuracy (by avoiding error propagation) and results in better accommodation performance as well as increased detection rate.

Algorithm 1 Controller

procedure CONTROLLER
  Input: $\boldsymbol{d}$, $\boldsymbol{e}_U$, and x_s for all $s \in \mathcal{S}_U$;
  At starting of each time step n:
  for $s = 1 : N_U$ **do** ▷ `Corresponds to` $s \in \mathcal{S}_U$
    if $d_s[n-1] > \upsilon$ **then** ▷ `Identified faulty`
      if $\phi_{E,s} < \tau$ **then**
        Feed $\hat{x}_s[n-1]$ instead of $x_s[n-1]$ to the prediction block as input;
      if $\phi_{P,s} < \phi_{E,s}$ and $\phi_{P,s} < \tau$ **then**
        Obtain $\tilde{x}_s[n]$ from Eq. (2);
        Feed $\tilde{x}_s[n]$ instead of $x_s[n]$ to the estimation block as input;
      else if $\phi_{E,s} < \tau$ **then**
        Obtain $\hat{x}_s[n]$ from Eq. (1);
        Feed $\hat{x}_s[n]$ instead of $x_s[n]$ to the estimation block as input;
    else ▷ `Identified healthy`
      Update $\phi_{E,s}$, $\phi_{P,s}$ using $\boldsymbol{e}_U[n-1]$;

III. NNs CONFIGURATION

The MLP is a feed-forward layered NN made up of an input layer, an arbitrary number of hidden layers and an output layer, where neurons are interconnected in forward direction from the input to the output layer [47]. The MLP is a suitable NN for regression and classification tasks and is capable to model arbitrary non-linearities while exhibiting fine generalization on unseen data [38], [41]. MLP NNs in the proposed SFDIA architecture are trained using an optimization algorithm [48].

A. Estimators and Predictors

Each MLP-based **estimator** has been implemented with $(N-1)\cdot(L_e+1)$ inputs, H_v hidden layers with N_v neurons each, and a single output. Conversely, each MLP-based **predictor** has been implemented with L_p inputs, H_v hidden layers with N_v neurons each, and a single output. For both the estimators and predictors, the hyperbolic tangent has been selected as the activation function for the hidden layers, while the linear activation function has been selected for the output layer.

Training was accomplished using the Nesterov-accelerated adaptive moment estimation (Nadam) optimization algorithm [49] over real-world data-sets. The mean square error (MSE) loss function was considered as the relevant optimization metric for both the estimators and the predictors. More specifically, the MSE loss for sth estimator and predictor, respectively, is defined as

$$\mathcal{L}_{\mathrm{es},s}(\boldsymbol{\phi}_s) = \frac{1}{w}\sum_{j=0}^{w-1}(\hat{x}_s^j(\boldsymbol{\phi}_s) - x_s^j)^2 \tag{9}$$

$$\mathcal{L}_{\mathrm{pr},s}(\boldsymbol{\varphi}_s) = \frac{1}{w}\sum_{j=0}^{w-1}(\tilde{x}_s^j(\boldsymbol{\varphi}_s) - x_s^j)^2 \tag{10}$$

where w is the number of samples in each batch, $\boldsymbol{\phi}_i$ (resp. $\boldsymbol{\varphi}_i$) represents the vector of trainable parameters of the sth estimator (resp. predictor). Finally, $\hat{x}_s^j$ (resp. $\tilde{x}_s^j$) is the network output associated to the sth estimator (resp. predictor), while x_s^j denotes the true measurement (viz. the labeled sample) of sth sensor.

B. Classifier

The MLP-based **classifier** has been implemented with $2N_U\cdot(L_c+1)$ inputs, H_c hidden layers with N_c neurons each, and N_U outputs. The hyperbolic tangent has been selected as the activation function for the hidden layers, while a logistic (viz. sigmoid) activation function has been selected for each node in the output layer. In order to accomplish both detection & identification tasks, a loss capitalizing *multitask learning* is employed for training the classifier. Specifically, a *weighted sum* of the losses of the N_U binary (fault/no-fault) classification tasks associated with the unreliable sensors is minimized, i.e.

$$\mathcal{L}_{\mathrm{cl}}\left(\boldsymbol{\theta}_{\mathrm{shared}}, \{\boldsymbol{\theta}_s\}_{s=1}^{N_U}\right) \triangleq \sum_{s=1}^{N_U}\lambda_s\,\mathcal{L}_s\left(\boldsymbol{\theta}_{\mathrm{shared}}, \boldsymbol{\theta}_s\right) \tag{11}$$

In the above formula, the weight λ_s indicates the preference level of the sth task (i.e. detection of a fault at sth unreliable sensor). It is worth noticing that the multitask objective function allows the proposed classifier to solve multiple learning tasks *at once* (i.e. via a single NN). Accordingly, in the above expression, $\boldsymbol{\theta}_{\mathrm{shared}}$ represents the vector of *shared* parameters of the MLP common to all the N_U different tasks, whereas $\boldsymbol{\theta}_s$ indicates the vector of parameters which are *task-specific* for sth learning task.

In this work uniform weighting is adopted, i.e. $\lambda_s = 1/N_U$ for $s = 1,\ldots,N_U$, and a binary cross-entropy (BCE) loss function for *all* the N_U binary tasks $\mathcal{L}_1(\cdot),\ldots,\mathcal{L}_{N_U}(\cdot)$. The BCE loss for sth task is formally defined as

$$\begin{aligned}\mathcal{L}_s\left(\boldsymbol{\theta}_{\mathrm{shared}}, \boldsymbol{\theta}_s\right) = -\frac{1}{w}\sum_{j=0}^{w-1}\Big\{&y_s^j\,\ln d_s^j(\boldsymbol{\theta}_{\mathrm{shared}}, \boldsymbol{\theta}_s) + \\ &(1-y_s^j)\ln\left(1-d_s^j(\boldsymbol{\theta}_{\mathrm{shared}}, \boldsymbol{\theta}_s)\right)\Big\}\end{aligned} \tag{12}$$

where w is the number of samples in each batch. Furthermore d_s^j is the entry of classifier output associated to sth sensor, while y_s^j denotes (the $0/1$ representation of) the true fault status (viz. the labeled sample) of sth sensor. The same optimization algorithm (i.e. Nadam) as the estimators/predictors is employed for training the aforementioned MLP-based classifier.

C. Summary of the training phase

The whole training process of the proposed SFDIA architecture is summarized in Algorithm 2. In detail, the estimators and predictors (Sec. III-A) are trained *only* with healthy (fault-free) data, according to the inputs specified via Eqs. (1) and (2), respectively. A similar comment applies to the associated validation set for estimators and predictors.

Conversely, the classifier block (Sec. III-B) is also trained based on *faulty* training data, by including the controller and residual evaluation blocks in an open-loop fashion. Indeed, during the training process, the controller is given the classifier label set (i.e. the binary-valued vector pattern collecting true faulty/healthy condition for all the sensors) as input, in the place of the classifier decision vector. This is to avoid detrimental effects due to training instability of the classifier. However, since perfect identification provided by the label set may lead to overfitting, 25% of controller decisions are randomly dropped out to help the classifier generalize better during the training process. The corresponding validation set for the classifier block includes faulty measurements as well.

Algorithm 2 Training Process

procedure INITIALIZE
  Preparing training set and create a falsified copy;
  Random weights and biases for all networks;
  Set initial value of all other parameters to zero;
procedure ESTIMATORS AND PREDICTORS
  Input: healthy training set; ▷ `Fault free`
  while Epoch number $<$ Max epoch or Validation loss Not triggered **do**
    for each epoch **do**
      Calculate MSE;
      Update weights and biases using Nadam optimization;
      Calculate validation loss;
procedure CLASSIFIER
  Input: Falsified training set;
  while Epoch number $<$ Max epoch or Validation loss Not triggered **do**
    for each epoch **do**
      Obtain $\hat{x}_s$ $\tilde{x}_s$ for all $s \in \mathcal{S}_U$ with respect to the controller mechanism;
      Calculate residual signals;
      Feed residual signals to the classifier;
      Calculate weighted BCE;
      Update weights and biases using Nadam optimization;
      Calculate validation loss;

IV. DATA-SETS AND FAULTS SETUP

A. Data-sets Setup

For the sake of a complete evaluation, three real-world data-sets (similarly as [41]) have been employed to assess the proposed SFDIA architecture. Specifically, the **air quality (AQ)** data-set [42] includes readings from *five* chemical sensors (assumed to be unreliable, namely $N_U = 5$) which are complemented by measurements originating from humidity and temperature sensors (assumed to be reliable, i.e. $N_R = 2$). Such a sensor system is aimed at pollution-level evaluation in an Italian city. The second data-set is related to a **wireless sensor network (WSN)** with *four* unreliable sensors measuring indoor and outdoor humidity and temperature [43]. Labeled anomalies injected into the data-set were omitted and only the temperature readings of the multi-hop section of data-set are considered as unreliable readings ($N_U = 4$, $N_R = 0$) for our analysis. The last data-set includes multiple sensors on a **permanent-magnet synchronous motor (PMSM)** [44], [50]. Among the collected measurements[2], (i) coolant temperature, (ii) voltage and (iii) current (summation of q and d components), (iv) motor speed and (v) torque are included in the unreliable set $\mathcal{S}_U$ (thus $N_U = 5$), whereas the stator yoke temperature is assumed to belong to the reliable set $\mathcal{S}_R$ (thus $N_R = 1$).

Before feeding the data-sets to the proposed architecture, sensors readings in each data-set are normalized using min-max scaling on the training set to avoid polarization during the learning process. Finally, the entire rows containing missing values are ignored from the data-sets. Table I summarizes data-sets description.

TABLE I: Data-sets description. The reliable sensors in each data-set are highlighted in italic.

Data-set	Samples	N_U	N_R	Attributes
AQ	8991	5	2	**Multivariate**, **time-series**; carbon monoxide (CO), non-metanic hydrocarbons (NMH), nitrogen oxides (NO_x), nitrogen dioxide (NO_2) and ozone (O_3) gas concentrations, as well as measurements of *temperature* and *humidity*
WSN	4589	4	0	**Multivariate**, **time-series**; four temperature sensors: two indoor, two outdoor
PMSM	55000	5	1	**Multivariate**, **time-series**; coolant temperature, voltage and current (summation of q and d components), motor speed, torque and *stator yoke* temperature

B. Sensor Faults Modeling

The performance of the proposed SFDIA architecture is evaluated under transient faults.

Also, with the aim of adapting and examining the proposed architecture according to DTs' needs, *four different fault types* with varying severity levels were modeled, as detailed hereinafter. It is worth highlighting that the practice of modeling simulated faults superimposed to real data is a common practice in the evaluation of SFDIA systems (e.g. [12], [29], [39]), as (i) real faulty measurements are sporadic and very hard to obtain and (ii) simulated faults also allow quantifying accommodation performance. This is also to highlight the *generality* of the proposed architecture in accommodating diversified faulty conditions.

[2]The readings were sampled with 1.5 s-intervals and the first 55k readings were picked after sampling.

Bias faults: for each bias fault, a constant bias b injected to the normal operation data-sets for M consecutive samples as follows

$$x_{s,b}[n] = \begin{cases} x_{s,h}[n] + b\,, & 0 \leq n-m < M \\ x_{s,h}[n]\,, & \text{otherwise} \end{cases} \tag{13}$$

where $x_{s,h}[n]$ and $x_{s,b}[n]$ are the healthy and possibly-faulty reading of sensor $s \in \mathcal{S}_U$, respectively, under bias fault. Finally, m denotes the starting time instant of the fault.

Drift faults: as for drift fault, an additive term drifts to bias level b in M samples and remains for K samples ($M > K$), namely:

$$x_{s,d}[n] = \begin{cases} x_{s,h}[n] + \frac{b(n-m+1)}{M}\,, & 0 \leq n-m < M \\ x_{s,h}[n] + b\,, & M \leq n-m < M+K \\ x_{s,h}[n]\,, & \text{otherwise} \end{cases} \tag{14}$$

where $x_{s,d}[n]$ is the possibly-faulty reading of sensor $s \in \mathcal{S}_U$ under drift-type faults.

Noise faults: in the latter case, zero-mean additive Gaussian noise $w[n] \sim \mathcal{N}(0, c)$ is added to the sensor measurement for M consecutive samples, i.e.:

$$x_{s,g}[n] = \begin{cases} x_{s,h}[n] + w[n]\,, & 0 \leq n-m < M \\ x_{s,h}[n]\,, & \text{otherwise} \end{cases} \tag{15}$$

where $x_{s,g}[n]$ is the possibly-faulty reading of sensor $s \in \mathcal{S}_U$ under noise-type faults and c is the variance of the noise.

Freeze faults: for freeze-type faults, sensor output stuck at previous reading for M consecutive samples as follows

$$x_{s,f}[n] = \begin{cases} x_{s,h}[m-1]\,, & 0 \leq n-m < M \\ x_{s,h}[n]\,, & \text{otherwise} \end{cases} \tag{16}$$

where $x_{s,f}[n]$ is the possibly-faulty reading of sensor $s \in \mathcal{S}_U$ under freeze-type faults.

V. Numerical Results

The effectiveness of the proposed architecture for detection, isolation and accommodation of sensor faults has been assessed by means of a comprehensive analysis conducted on the three previously-described real world data-sets. The following section first details the considered system setup and employed parameters, for the sake of reproducibility (Sec. V-A). Then, the working principle of the two relevant SFDIA baselines used for comparison is recalled (Sec. V-B). Finally, the SFDIA performance is reported and discussed (Sec. V-C).

A. System Setup and Parameters

Training and Evaluation Setup: MLP NNs within the proposed architecture were trained using the first 70% and 15% of samples of each data-set as train set and validation set, respectively. The rest ending 15% of samples of each data-set was used as test set for performance evaluation. A validation process based on *early stopping* method [51] was employed during the training phase to avoid over-fitting: the training process was stopped if the loss on the validation set had not decreased for 20 consecutive epochs or if the maximum number of epochs was reached[3].

Hyperparameter specification of proposed approach: As in [41], a similar configuration for the classifiers and the estimators was considered. More specifically, estimators and predictors with $H_v = 1$ hidden layer, $N_v = 10$ nodes per hidden layer and $L_v = L_p = 10$, along with a classifier with $H_c = 2$ hidden layers, $N_c = 15$ nodes per hidden layer and $L_c = 10$ were trained. Table II lists MLPs' configurations and corresponding hyper-parameters of the proposed architecture. In addition, the predefined thresholds τ and υ are set to 0.15 and 0.9 for the controller, respectively. The threshold τ needs to be adjusted with respect to the system tolerable level of deviation as well as the estimators/predictors accuracy, whereas threshold υ is selected heuristically according to the system performance on the validation set.

Random generation of synthetic faults: The four types of faults considered in this work are synthetically generated [12], [29], [39] according to the corresponding models detailed in Sec. IV-B on the top of the *real measurement data* described in Sec. IV-A. Unless otherwise stated, the fault absolute level b (with unbiased random positive and negative faults) and noise variance c are assumed uniformly distributed between 0.2 and 0.4 to represent weak fault signals. The fault length (M and K) is also assumed uniformly distributed between 3 and 11 consecutive samples to represent transient faults[4]. It is worth stressing that the uniform distribution choice for the fault level b (resp. the noise variance c) and the fault length (M and K) helps the classifier to generalize better without focusing on a specific fault level/length [37], [39]. To verify the robustness of the proposed architecture against simultaneous faults, *up to three concurrent faulty sensors* were considered for the (fault-)generation process.

Training phase of classifier block: Fig. 6 shows the evolution of the classifier loss function vs. number of epochs (during the training phase) on both training (solid lines) and validation (dashed lines) sets, under *bias* faults. Indeed, validation and training losses under other fault types resemble those shown under bias fault and are thus omitted for brevity. For completeness, both the weighted (multitask) BCE (cf. Eq. (11)) and the per-sensor BCE (cf. Eq. (12)) are reported in Figs. 6a and 6b, respectively. As evident from the curves, the training phase on WSN data-set stops after ≈ 260 epochs ("□" marker) by early-stopping mechanism as the validation loss stops decreasing. Conversely, the training phase on the other two data-sets stops after reaching the maximum number (400) of epochs.

B. Considered Baselines

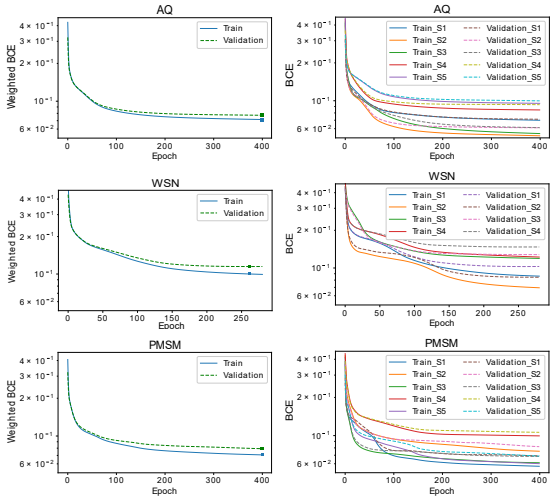
Results of the proposed approach in terms of detection, identification and accommodation performance are compared

[3]We implement the proposed architecture and other baselines using Keras Python API running on TensorFlow version 2.9.2 on MacBook pro M1 CPU 2.1-3.2 GHz with 16 GB memory.

[4]Under freeze fault, the fault length (M) is uniformly distributed between 100 and 400 consecutive samples due to smooth oscillating (WSN and PMSM) data-sets. Smaller fault lengths cause negligible faults on the working data-sets.

TABLE II: Configuration of the proposed architecture.

Parameter	Estimator	Predictor	Classifier
No. of input nodes	$(N-1)\cdot(L_v+1)$	L_p	$2N_U\cdot(L_c+1)$
No. of output nodes	1	1	N_U
No. of hidden layers	1	1	2
No. of nodes per hidden layer	10	10	15
Output activation	Linear	Linear	Sigmoid
Hidden layers activation	Tanh	Tanh	Tanh
Optimizer	Nadam	Nadam	Nadam
Loss function	MSE	MSE	BCE
Maximum epochs	400	400	400
Batch size	20 (50 for PMSM)	20 (50 for PMSM)	200
Learning rate	$4\cdot10^{-4}$ (10^{-3} for PMSM)	$4\cdot10^{-4}$ (10^{-3} for PMSM)	10^{-3}



(a) Weighted loss. (b) Per-sensor loss.

Fig. 6: Training and validation loss of the classifier during the training phase under bias fault.

with *two* state-of-the-art architectures: (i) M-SFDIA [41] and (ii) AE [12].

Similar to the proposed method, our previous M-SFDIA proposal is able to detect and isolate faulty sensors from patterns within the input residual signals. However, solely a bank of estimators is used to derive the residual signals, and to accommodate unreliable sensors in M-SFDIA method. Additionally, the controller block is absent in M-SFDIA. Furthermore, the original M-SFDIA's decision logic was designed to detect, isolate and accommodate only *up to one faulty sensor*. For this reason and for the sake of a fair comparison, the same decision logic as the proposed method was used (see Sec. II-D) to enable the M-SFDIA method to detect, isolate and accommodate multiple sensors simultaneously.

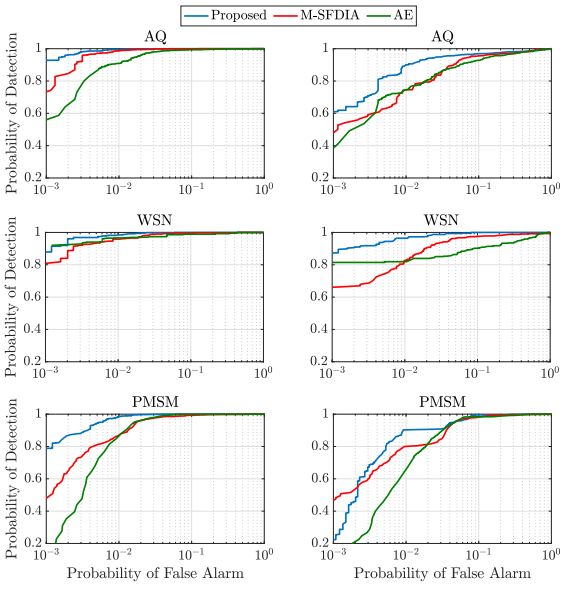
Conversely, the AE-based architecture devised in [12] is based on a *two-stage* approach. Specifically, the first stage is represented by a (standard) AE to learn data correlations among sensors, and detect anomalies (viz. faults) by tracking the MSE between input and output of the AE. As for the accommodation task, a second stage based on a (supervised) denoising AE is then used to clean faulty data. It is worth noticing that the identification task for AE architecture was not addressed in the original work [12]. Indeed, in the aforementioned AE-based method, the overall MSE of input and output (reconstructed) vector of the first AE is compared to a predefined threshold for fault detection only. As opposed to the aforementioned decision logic, herein (for the sake of a fair comparison) the squared error between the corresponding input and output *for each entry* (viz. unreliable sensor) is traced. Then, this error is compared with a predefined threshold σ, enabling the AE method to both detect & identify the faulty sensors[5]. Specifically, similar to the proposed method, $\max_{s=1}^{N_U} e_{AE,s}[n] \gtrless \sigma$ is used for detection, where $e_{AE,s}[n]$ is the squared error for the sth unreliable sensor. Accordingly, for the identification task, the set of identified faulty sensors is obtained as $\mathcal{I}_U \triangleq \{s \in \mathcal{S}_U : e_{AE,s}[n] > \sigma\}$.

C. Performance Analysis and Comparison

Fig. 7 illustrates *fault detection performance* in terms of probability of detection vs. probability of false alarm, i.e. showing the receiver operating characteristic (ROC) curves. In this case, a fault rate[6] $F_R = 0.1$ is considered. Also, ROC performance is reported *separately* for each of the three data-sets and for all four fault typologies considered. It is evident that the proposed architecture outperforms the two baselines for all four fault types. Specifically, the best detection rate is attained on AQ data-set when bias faults are present. Also, for all architectures, detection accuracy under bias faults appears to be generally higher than the other types of faults. Moreover, as can be seen, AE architecture fails to detect freeze faults on the WSN data-set. Indeed, drift and freeze faults are "trickier" to detect since they slowly appear in the system and have a less-appreciable effect on spatio-temporal correlations within the system.

[5]Numerical results (not shown for brevity) based on the original detection logic as [12], namely $\sum_{s=1}^{N_U} e_{AE,s}[n] \gtrless \sigma$ (and a matched identification logic, i.e. $\mathcal{I}_U \triangleq \{s \in \mathcal{S}_U : e_{AE,s}[n] > \sigma/N_U\}$) highlighted worse performance than the considered variant, due to the inability to cope with weak (and transient) faults.

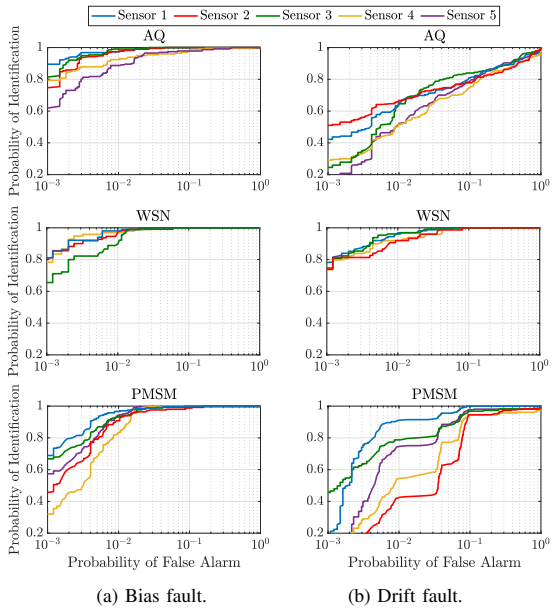
[6]Fault rate refers to the ratio between the number of faulty and non-faulty samples.



(a) Bias fault. (b) Drift fault.

(c) Noise fault. (d) Freeze fault.

Fig. 7: Detection performance in terms of ROC curves for all architectures over different fault types.



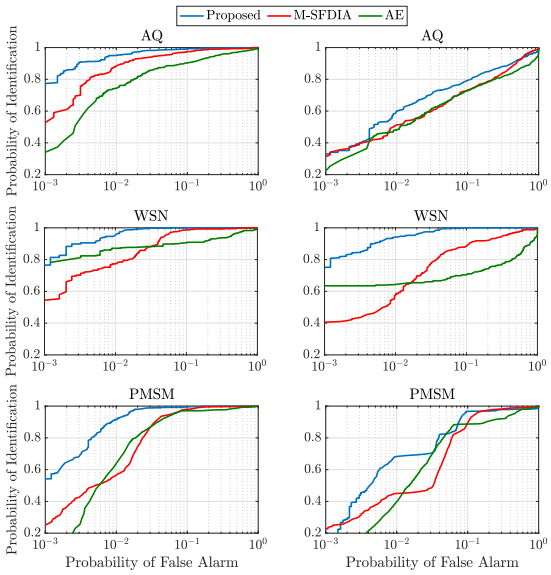
(a) Bias fault. (b) Drift fault.

(c) Noise fault. (d) Freeze fault.

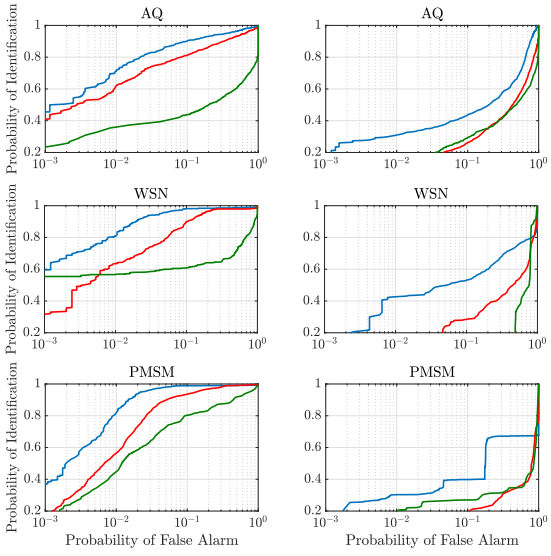
Fig. 8: Identification (isolation) performance in terms of ROC curves for the proposed architecture over different fault types. Sensor numbers refer to sensor indices.

Delving into real-time performance of SFDIA architectures, in Tab. III a *detection delay analysis*[7] for fixed false alarm rate of 10^{-2} is reported. Specifically, the *expected detection delay* is evaluated, defined as the average number of samples needed by an SFDIA architecture to detect a faulty sensor. The latter delay is indeed another important indicator of the SFDIA framework performance, which has a crucial effect on DTs functionality. In the experiments, the fault rate is set to $F_R = 0.5$ to generate a sufficient number of fault events allowing to obtain a reliable estimate of the aforementioned metric. Results highlight that the proposed architecture achieves the *lowest*

[7]Every span of simultaneous faults is considered as a unified fault.



(a) Bias fault. (b) Drift fault.



(c) Noise fault. (d) Freeze fault.

Fig. 9: Averaged identification (isolation) performance in terms of ROC curves for all architectures over different fault types.

detection delay in comparison to the state-of-the-art for all data-sets and fault types considered. Specifically, the average detection delay for the proposed architecture is confined below 1 sample (except for the AQ data-set with drift fault-types), whereas the other two architectures *always require a longer span to detect fault(s) within the system*. The most evident

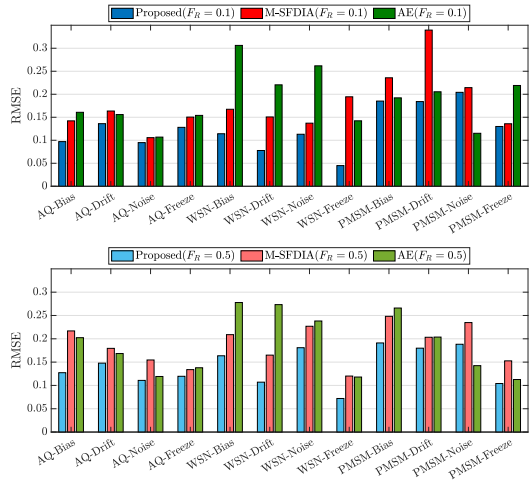


Fig. 10: Comparison of accommodation performance in term of RMSE ($P_f = 10^{-2}$).

performance difference is observed on the PMSM data-set for drift faults (boldface in Tab. III). Indeed, in the latter case, the proposed architecture detects weak faults on average after 0.67 samples whilst MSFDIA and AE architectures take on average 3.40 and 8.30 samples to detect the same faults, respectively. The reported difference corresponds to a *faster detection for our proposal* of more than $5\times$ and $12\times$ than the MSFDIA and AE architectures, respectively. The reduced detection delay of the proposed architecture is mainly due to the *joint* exploitation of estimation and prediction blocks (cf. Sec. II-B), as they provide complementary (residual) information for the classifier.

TABLE III: Detection delay Analysis. Results refer to bias and drift faults and are in the format avg. ($\pm$ std.) delayed samples obtained for a fault rate $F_R = 0.5$.

Data-set	Fault type	Proposed	M-SFDIA	AE
AQ	Bias	0.06 ($\pm$ 0.30)	0.39 ($\pm$ 1.11)	0.50 ($\pm$ 1.33)
	Drift	1.77 ($\pm$ 1.69)	2.33 ($\pm$ 2.09)	4.04 ($\pm$ 2.84)
WSN	Bias	0.28 ($\pm$ 0.91)	0.84 ($\pm$ 1.47)	0.31 ($\pm$ 1.08)
	Drift	0.72 ($\pm$ 1.12)	2.12 ($\pm$ 2.14)	2.73 ($\pm$ 2.12)
PMSM	Bias	0.10 ($\pm$ 0.53)	1.24 ($\pm$ 1.94)	3.61 ($\pm$ 3.62)
	Drift	**0.67 ($\pm$ 1.21)**	**3.40 ($\pm$ 2.68)**	**8.30 ($\pm$ 4.85)**

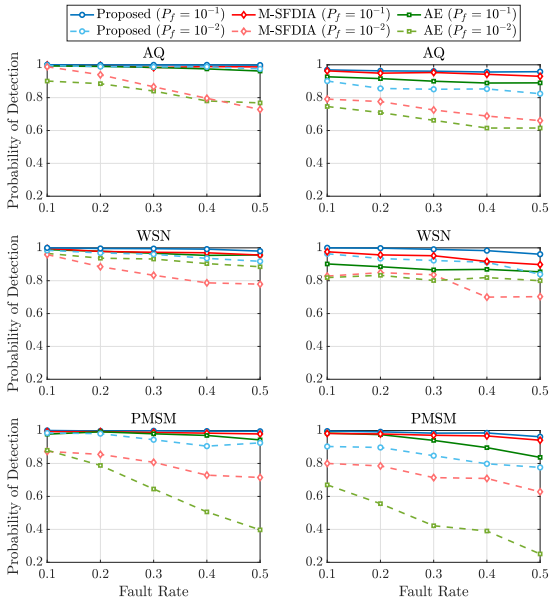
Fig. 8 shows the *identification performance* from the individual sensor perspective for the proposed architecture under different fault types. The probability of identification refers to the probability that SFDIA architecture correctly isolates the corresponding faulty sensor(s), where the averaged value is the average probability of identification over all unreliable sensors in each data-set. Apparently, different sensors undergo different performances, mostly depending on the level of spatio-temporal correlation (implicitly) providing the available redundant information within the system. The corresponding

sensor-averaged identification performance (under the same fault rate) is depicted in Fig. 9. Here in Fig. 8 and 9, the proposed architecture performs even better over other methods since it manages to reduce fault propagation within the architecture itself and avoid functionality degradation using the controlling block. Replacing faulty sensors with their estimates or predictions by the controller provides the classifier with easier interpretative residual signals.

The *accommodation performance* in terms of root mean square error (RMSE) is shown in Fig. 10, where fault rates $F_R \in \{0.1, 0.5\}$ are considered. Herein the term *error* means the difference between sensor healthy values before adding the fault and the accommodated values provided by the SFDIA architecture (or the original values, in the case of an undetected/unidentified fault). First of all, it is apparent that the proposed architecture outperforms the M-SFDIA architecture by presenting more accurate replacements for faulty data. The reason is that the proposed architecture relies on a combined estimator/predictor pair for each sensor and a controller block to continuously improve the accommodation performance by modifying their inputs based on the decision vector obtained from the classifier in a closed-loop fashion. Conversely, the M-SFDIA architecture does not take advantage of these excessive data. Finally, the proposed architecture outperforms AE-based SFDIA on all the three available data-sets (except for PMSM-Noise), with the higher improvement (viz. RMSE reduction) in the case of WSN data-set.

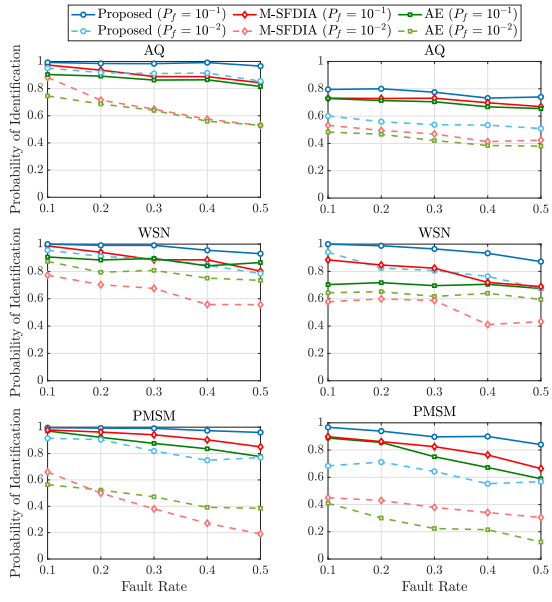
The rest of analysis specifically focuses on bias and drift faults as they well represent sudden (*hard*) faults and slowly appearing (*soft*) faults, respectively. The *impact of different fault rates on the detection and (averaged) isolation performance* is assessed in Figs. 11 and 12, respectively. In the above cases, two relevant false-alarm probability values are considered, namely $P_f = 10^{-1}$ and $P_f = 10^{-2}$. As expected, both detection and identification results reveal that higher fault rates have a negative impact on the architecture overall performance, as well as the considered baselines. Still, while the proposed architecture is capable to preserve its detection and isolation performance by incurring a milder detection/identification loss, both AE and M-SFDIA architectures exhibit a higher degradation with the fault rate. This outcome is mostly due to the estimators and predictors limiting the impact of fault propagation within the proposed architecture. For instance, referring to PMSM data-set, drift faults and $P_f = 10^{-2}$, a (harsh) fault-rate condition equal to 0.3 leads to a detection probability ≈ 0.4 (resp ≈ 0.7) for AE (resp. M-SFDIA). This corresponds to a 30% (resp. 10%) decrement w.r.t. a fault-rate scenario equal to 0.1. On the contrary, our architecture attains a detection probability ≈ 0.85 in the same harsh condition, with a corresponding degradation (w.r.t. fault-rate equal 0.1) equal to 0.05.

Figs. 13 and 14 compare *the performance trend of different architectures under versus the fault level b*. Clearly, detection and isolation performance for all architectures under strong faults are higher than the case of weaker faults. However, this in turn motivates the importance of developing techniques suited for weak faults. Results demonstrate a clear advantage of the proposed architecture over other architectures for different fault levels, with performance improvement being



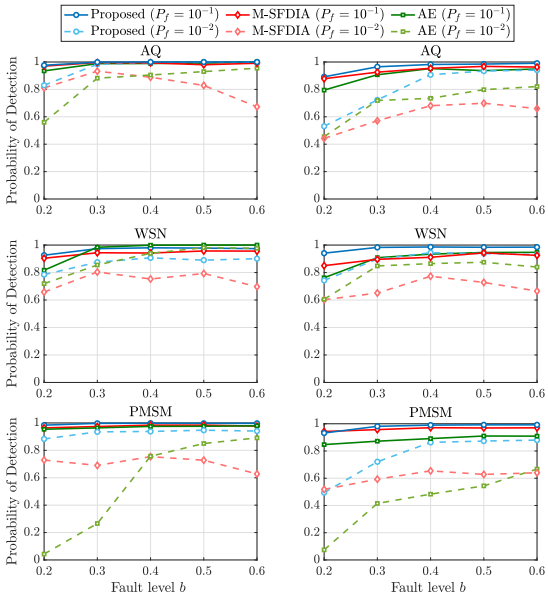
(a) Bias fault. (b) Drift fault.

Fig. 11: Impact of different fault rate on the detection accuracy.



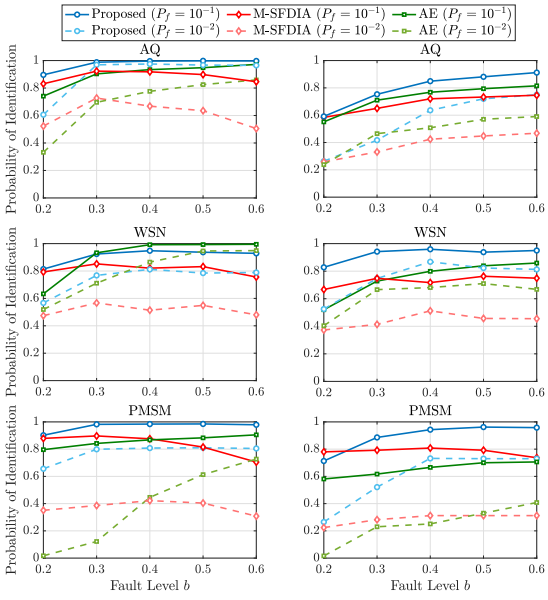
(a) Bias fault. (b) Drift fault.

Fig. 12: Impact of different fault rates on the averaged identification accuracy.



(a) Bias Fault (b) Drift Fault

Fig. 13: Impact of different fault level (b) on the detection accuracy.



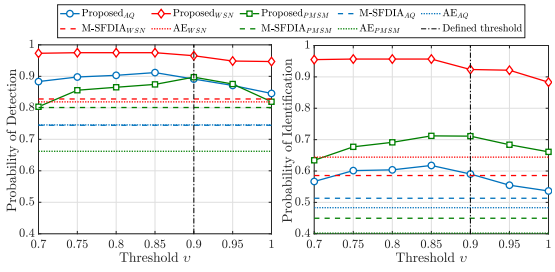
(a) Bias fault. (b) Drift fault.

Fig. 14: Impact of different fault level (b) on the averaged identification accuracy.

extremely evident under weak faults. For instance, referring to the case of bias faults with $b = 0.2$ on the PMSM data-set and assuming $P_f = 10^{-2}$, the proposed architecture achieves correct-identification probability of 0.9 while the AE architecture is below 0.1. The AE architecture mostly exploits change detection in the correlation structure of the signals and weak faults might have a negligible impact from this perspective. Conversely, the combined use of estimators, predictors and residual processing employed by the proposed architecture is able to detect & isolate these "low-observable" faults. Moreover, as the fault level increases, the proposed architecture is overtaking the M-SFDIA architecture since the proposed method mitigates propagation of strong faults within the architecture by means of the controller block.

To deepen the investigation of the controller block, a *sensitivity analysis* was also performed, focusing on detection and identification performance of the proposed architecture, by varying the threshold υ during the test phase. More specifically, Fig. 15 shows the detection and identification performance of the proposed method with respect to the threshold υ. To better apprehend the impact of the threshold υ, the detection and identification performance of the state-of-the-art counterparts were reported as a lower bound. Results highlight quite smooth performance trends on the three data-sets with respect to the threshold υ. Interestingly, predefined threshold $\upsilon = 0.9$ based on the validation set is pretty near to the optimum value on the test set.

Finally, to have a finer-grained view of the three architectures for detection & isolation tasks, Fig. 16 reports their



(a) Detection Sensitivity (b) Identification Sensitivity

Fig. 15: Impact of threshold (υ) on the detection and identification accuracy ($P_f = 10^{-2}$). Threshold $\upsilon = 1$ associated to a zero-effect of the controller (i.e. off-circuit controller).

decision outcomes for a time-segment long 50 samples taken from the PMSM data-set under bias fault ($P_f = 10^{-2}$). Specifically, for each time index n, "$\circ$" symbol denotes the actual (true) faulty sensors, whereas "None" is used in the case of a healthy system. Then, for each architecture, the miss-detected faults (denoted with red "$*$" symbol) and the false-alarms (i.e. sensors erroneously declared as faulty by the architecture when the system is healthy, with blue "$\times$" symbol) are highlighted. Finally, when each SFDIA architecture declares a detection, the corresponding identified faults are reported with a green "$+$" symbol. The most eminent point in Fig. 16 is that, by resorting to the proposed architecture, *only one fault remained*

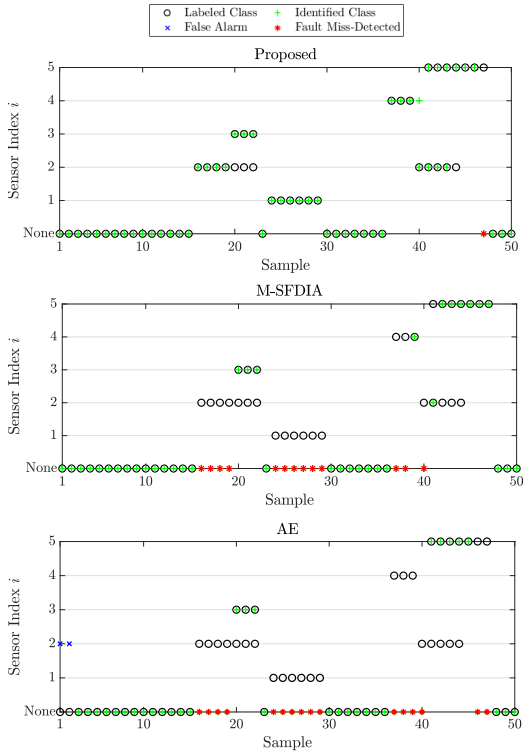


Fig. 16: Visualization of fault classification for all architectures on PMSM data-set.

undetected whereas M-SFDIA and AE architectures miss-detected 13 and 16 out of 24 faulty samples, respectively. As mentioned earlier, the proposed architecture attains better prompt detection & identification performance with respect to its counterparts. For instance, according to Fig. 16, the latter two architectures were only capable to identify only one faulty sensor for the given snapshot when simultaneous faults occurred, while the proposed architecture successfully identified most of them.

D. Complexity Analysis

As the final stage of the numerical comparison, the proposed approach is compared with the considered baselines in terms of the relevant computational complexity involved, by looking at both the (i) training and (ii) operational (testing) phases.

Regarding the *training phase*, the number of trainable parameters associated with each architecture is summarized in Tab. IV. Trainable parameters refer to weights and biases of each NN to be learned during the training phase (through stochastic gradient descent by resorting to the back-propagation technique) in the architecture. Clearly, the number of trainable parameters grows with the complexity of the (sensor) system to be accommodated, with the higher complexity associated with AQ data-set on all three architectures. Also, the info in the table highlights that the proposed architecture has a *comparable complexity with M-SFDIA while enjoying shorter training times than the considered AE*. Furthermore, thanks to the modularity granted by the proposed approach, different blocks of the considered architecture (e.g. estimators and predictors) could be trained in a parallel fashion on distributed (e.g. cloud) architectures.

Regarding the *testing phase*, the assumption of an equal number of hidden layers ($H_v = H_c = H_J$), time delays ($L_v = L_p = L_c = L_J$) and nodes per hidden layer ($N_v = N_c = N_J$) is made, as considered in [41], where index J refers to the joint value. Additionally, the impact of the activation functions is neglected (for simplicity). Accordingly, the computational complexity of the operational phase is analyzed in terms of the well-known big-$\mathcal{O}$ (Landau's) notation. First, it is worth recalling that the computational complexity of M-SFDIA approximately equals $\mathcal{O}(L_J N_U^2 N_J + L_J N_R N_U N_J + H_J N_U N_J^2)$ for one input sample [41]. Furthermore, the complexity cost of each predictor in the proposed architecture is approximately $\mathcal{O}(L_J N_J)$. Accordingly, the overall computational complexity of the proposed architecture approximately equals the M-SFDIA architecture. Indeed, the complexity is mainly dominated by the computational cost of the estimators and of the classifier, which is almost equal in both architectures [41]. Indeed, the impact of the residual and controller block operations is negligible in the overall cost.

TABLE IV: Number of NNs' trainable parameters.

Data-set	N_U	N_R	**Proposed**			**M-SFDIA**		**AE**	
			Est.	Pre.	Clf.	Est.	Clf.	AE	Denoising-AE
AQ	5	2	681	121	1985	681	1160	16945	16945
			In total = 5995			4565		33890	
WSN	4	0	351	121	1639	351	979	5470	5470
			In total = 3527			2383		10940	
PMSM	5	1	571	121	1985	571	1160	12250	12250
			In total = 5445			4015		24500	

For instance, for AQ data-set the computational complexity for estimators block is $\mathcal{O}(3510)$, for the classifier block[8] is $\mathcal{O}(1740)$ and for the predictors block is $\mathcal{O}(550)$. This results in a total computational complexity of $\mathcal{O}(5{\cdot}10^3)$ and $\mathcal{O}(6{\cdot}10^3)$ for M-SFDIA and the proposed architecture, respectively. Still, the proposed architecture attains substantially higher overall SFDIA performance at the expense of a manageably higher complexity (see Sec. V-C).

Conversely, for AE-based architecture (which is made of two similar AEs with a three-fold compression factor [12]), the computational complexity is $\mathcal{O}((452/81) \cdot (L_J \cdot (N_U + N_R))^2)$. Accordingly, in the peculiar case of AQ data-set, the computational complexity of the aforementioned architecture is approximately $\mathcal{O}((452/81) \cdot (10 \cdot (5+2))^2) \approx \mathcal{O}(3 \cdot 10^4)$ for one input sample. As a result, the complexity of the AE-based architecture appears to be considerably higher than that incurred by the proposed approach.

[8]The computational complexity for the classifier in M-SFDIA architecture is $\mathcal{O}(990)$.

VI. Conclusion

This article presented a four-layer architecture for SFDIA based on MLP NNs. Our contribution represents a stepping stone towards the development of (modular) DTs based on sensor systems/networks in IoT contexts. The (four) designed layers consist of estimation&prediction, residual, classification and controlling blocks. The classifier block at the heart of the architecture is in charge of detecting and identifying faulty sensors based on residual signals provided by estimators and predictors. Moreover, a controlling block is placed to track the classifier's decision output in order to boost overall system performance. This is accomplished by stopping fault propagation chain at the first layer by modifying estimators and predictors inputs with respect to the classifier's decision.

The proposed method was trained and tested on *three* real-world and publicly-available data-sets (i.e. [41], [42], [50]) for the sake of a complete and reproducible assessment. For the sake of generalization, four types of faults were considered in this study: bias, drift, noise and freeze. The proposed architecture yielded notably higher detection and isolation performance compared to the state-of-art M-SFDIA [41] and AE [12] architectures, for *all* four fault types. Moreover, the proposed architecture was shown to enjoy robustness against different fault rates while other architectures' performances were affected considerably.

Future works will focus on (i) the study of DTs for sensors operating under channel uncertainty, (ii) the design of SFDIA architectures which scale well with the number of sensors, (iii) the investigation of reinforcement-learning algorithms for optimized controller design and (iv) the application of explainable artificial-intelligence algorithms' in interpreting (and improving) the proposed SFDIA approach. Finally, more sophisticated NN approaches (e.g. convolutional NNs, RNNs) for each SFDIA module we will be also investigated with the intent of improving detection, identification and accommodation performance under specific circumstances while meeting the operational deployment constraints.

References

[1] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21 980–22 012, 2020.

[2] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[3] L. Chettri and R. Bera, "A comprehensive survey on internet of things (iot) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[4] S. Cui, F. Farha, H. Ning, Z. Zhou, F. Shi, and M. Daneshmand, "A survey on the bottleneck between applications exploding and user requirements in iot," *IEEE Internet of Things Journal*, pp. 1–1, 2021.

[5] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.

[6] J. Zhang, L. Li, G. Lin, D. Fang, Y. Tai, and J. Huang, "Cyber resilience in healthcare digital twin on lung cancer," *IEEE Access*, vol. 8, pp. 201 900–201 913, 2020.

[7] A. Francisco, N. Mohammadi, and J. E. Taylor, "Smart city digital twin–enabled energy management: Toward real-time urban building energy benchmarking," *Journal of Management in Engineering*, vol. 36, no. 2, p. 04019045, 2020.

[8] N. Mohammadi and J. E. Taylor, "Smart city digital twins," in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–5.

[9] A. Mahapatro and P. M. Khilar, "Fault diagnosis in wireless sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2000–2026, 2013.

[10] Z. Yang, N. Meratnia, and P. Havinga, "An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine," in *IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2008, pp. 151–156.

[11] X. Luo, Y. Li, X. Wang, and X. Guan, "Interval observer-based detection and localization against false data injection attack in smart grids," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 657–671, 2021.

[12] M. M. N. Aboelwafa, K. G. Seddik, M. H. Eldefrawy, Y. Gadallah, and M. Gidlund, "A machine-learning-based technique for false data injection attacks detection in industrial IoT," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8462–8471, 2020.

[13] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava, "Sensor network data fault types," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, Jun. 2009.

[14] T. Muhammed and R. A. Shaikh, "An analysis of fault detection strategies in wireless sensor networks," *Elsevier Journal of Network and Computer Applications*, vol. 78, pp. 267 – 287, 2017.

[15] Y. Jiang, S. Yin, J. Dong, and O. Kaynak, "A review on soft sensors for monitoring, control and optimization of industrial processes," *IEEE Sensors Journal*, 2020.

[16] M. M. Gharamaleki and S. Babaie, "A new distributed fault detection method for wireless sensor networks," *IEEE Systems Journal*, vol. 14, no. 4, pp. 4883–4890, 2020.

[17] E. Dubrova, "Hardware redundancy," in *Fault-Tolerant Design*. Springer, 2013, pp. 5–86.

[18] A. A. Amin and K. Mahmood-Ul-Hasan, "Advanced fault tolerant air-fuel ratio control of internal combustion gas engine for sensor and actuator faults," *IEEE Access*, vol. 7, pp. 17 634–17 643, 2019.

[19] S. Yin, B. Xiao, S. X. Ding, and D. Zhou, "A review on recent development of spacecraft attitude fault tolerant control system," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3311–3320, 2016.

[20] P. M. Papadopoulos, L. Hadjidemetriou, E. Kyriakides, and M. M. Polycarpou, "Robust fault detection, isolation, and accommodation of current sensors in grid side converters," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 2852–2861, 2017.

[21] J. Loy-Benitez, Q. Li, K. Nam, and C. Yoo, "Sustainable subway indoor air quality monitoring and fault-tolerant ventilation control using a sparse autoencoder-driven sensor self-validation," *Elsevier Sustainable Cities and Society*, vol. 52, p. 101847, 2020.

[22] G. Campa, M. Thiagarajan, M. Krishnamurty, M. R. Napolitano, and M. Gautam, "A neural network based sensor validation scheme for heavy-duty diesel engines," *ASME Journal of dynamic systems, measurement, and control*, vol. 130, no. 2, 2008.

[23] M. Ruba, R. O. Nemes, S. M. Ciornei, and C. Martis, "Simple and robust current sensor fault detection and compensation method for 3-phase inverters," *IEEE Access*, vol. 8, pp. 34 820–34 832, 2020.

[24] S. K. Kommuri, S. B. Lee, and K. C. Veluvolu, "Robust sensors-fault-tolerance with sliding mode estimation and control for PMSM drives," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 17–28, 2018.

[25] C. Sun and Y. Lin, "Adaptive output feedback compensation for a class of nonlinear systems with actuator and sensor failures," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–10, 2021.

[26] J. Zhang, S. Li, and Z. Xiang, "Adaptive fuzzy output feedback event-triggered control for a class of switched nonlinear systems with sensor failures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5336–5346, 2020.

[27] C. Lo, J. P. Lynch, and M. Liu, "Distributed reference-free fault detection method for autonomous wireless sensor networks," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 2009–2019, 2013.

[28] P. Liu, Y. Zhang, H. Wu, and T. Fu, "Optimization of Edge-PLC-Based fault diagnosis with random forest in Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9664–9674, 2020.

[29] S. Zidi, T. Moulahi, and B. Alaya, "Fault detection in wireless sensor networks through SVM classifier," *IEEE Sensors Journal*, vol. 18, no. 1, pp. 340–347, 2018.

[30] Guo Su, D. Fang, Sun Jian, and Li Fengmei, "Sensor fault detection with online sparse least squares support vector machine," in *Proceedings of the 32nd Chinese Control Conference*, 2013, pp. 6220–6224.

[31] S. Mandal, B. Santhi, S. Sridhar, K. Vinolia, and P. Swaminathan, "Nuclear power plant thermocouple sensor-fault detection and classification using deep learning and generalized likelihood ratio test," *IEEE Transactions on Nuclear Science*, vol. 64, no. 6, pp. 1526–1534, 2017.

[32] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. Salvo Rossi, "A data-driven architecture for sensor validation based on neural networks," in *IEEE Sensors Conference*, 2020, pp. 1–4.

[33] J. Gao, J. Wang, P. Zhong, and H. Wang, "On threshold-free error detection for industrial wireless sensor networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2199–2209, 2018.

[34] K. Jeong, S. B. Choi, and H. Choi, "Sensor fault detection and isolation using a support vector machine for vehicle suspension systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3852–3863, 2020.

[35] H. Zhao, "Neural component analysis for fault detection," *Chemometrics and Intelligent Laboratory Systems*, vol. 176, 12 2017.

[36] M. Elnour, N. Meskin, and M. Al-Naemi, "Sensor data validation and fault diagnosis using auto-associative neural network for HVAC systems," *Journal of Building Engineering*, vol. 27, p. 100935, 2020.

[37] S. Hussain, M. Mokhtar, and J. M. Howe, "Sensor failure detection, identification, and accommodation using fully connected cascade neural network," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 3, pp. 1683–1692, 2015.

[38] F. Balzano, M. L. Fravolini, M. R. Napolitano, S. d'Urso, M. Crispoltoni, and G. del Core, "Air data sensor fault detection with an augmented floating limiter," *Hindawi International Journal of Aerospace Engineering*, 2018.

[39] D. Haldimann, M. Guerriero, Y. Maret, N. Bonavita, G. Ciarlo, and M. Sabbadin, "A scalable algorithm for identifying multiple-sensor faults using disentangled RNNs," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2020.

[40] H. Zhang, Q. Zhang, J. Liu, and H. Guo, "Fault detection and repairing for intelligent connected vehicles based on dynamic Bayesian network model," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2431–2440, 2018.

[41] H. Darvishi, D. Ciuonzo, E. R. Eide, and P. Salvo Rossi, "Sensor-fault detection, isolation and accommodation for digital twins via modular data-driven architecture," *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4827–4838, February 2021.

[42] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008.

[43] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie, and M. Palaniswami, "Labelled data collection for anomaly detection in wireless sensor networks," in *6th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2010, pp. 269–274.

[44] W. Kirchgässner, O. Wallscheid, and J. Böcker, "Deep residual convolutional and recurrent neural networks for temperature estimation in permanent magnet synchronous motors," in *IEEE International Electric Machines & Drives Conference (IEMDC)*, 2019, pp. 1439–1446.

[45] H. Darvishi, D. Ciuonzo, and P. Salvo Rossi, "Real-time sensor fault detection, isolation and accommodation for industrial Digital Twins," in *IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2021.

[46] P. Tchakoua, R. Wamkeue, M. Ouhrouche, F. Slaoui-Hasnaoui, T. A. Tameghe, and G. Ekemb, "Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges," *MDPI Energies*, vol. 7, no. 4, pp. 1–36, April 2014.

[47] F. Rosenblatt, *The perceptron: a theory of statistical separability in cognitive systems (Project Para)*. Cornell Aeronautical Laboratory, 1958.

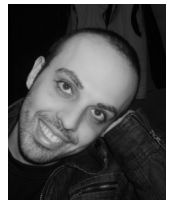
[48] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[49] T. Dozat, "Incorporating Nesterov momentum into Adam," in *International Conference on Learning Representations (ICLR)*, 2016.

[50] W. Kirchgässner, O. Wallscheid, and J. Böcker, "Empirical evaluation of exponentially weighted moving averages for simple linear thermal modeling of permanent magnet synchronous machines," in *IEEE 28th International Symposium on Industrial Electronics (ISIE)*, 2019, pp. 318–323.

[51] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

Hossein Darvishi (GS'20) received the B.Sc. degree from the Kermanshah University of Technology, Iran, and the M.Sc. degree (*ranked first*) in telecommunications engineering from K.N. Toosi University of Technology, Iran, in 2016 and 2018, respectively. Since 2020, he has been pursuing the Ph.D. degree in Electronics and Telecommunications with the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. He is currently a visiting researcher at the Signal Processing Laboratory (LTS4) in the Electrical Engineering Institute of the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. He is affiliated with the Nordic Industrial IoT Hub (HI2OT) and the SIGNIFY project (NTNU and SINTEF's research and connectivity program in sensor validation solutions for digital twins of safety-critical systems). He is a Reviewer for reputable journals including IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, and IEEE SENSORS JOURNAL. His research interests include statistical signal processing, machine learning, Internet of Things and wireless sensor networks.



Domenico Ciuonzo (S'11-M'14-SM'16) is an Assistant Professor at University of Napoli Federico II. He holds a Ph.D. from the University of Campania "L. Vanvitelli", Italy. Since 2011, he has been holding several visiting researcher appointments. He is currently a Technical Editor for the IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS and an Executive Editor for the IEEE COMMUNICATIONS LETTERS. He is the recipient of two Best Paper awards (IEEE ICCCS 2019 and Elsevier Computer Networks 2020), the 2019 Exceptional Service award from IEEE AESS, the 2020 Early-Career Technical Achievement award from IEEE SENSORS COUNCIL for sensor networks/systems and the 2021 Early-Career Award from IEEE AESS for contributions to decentralized inference and sensor fusion in networked sensor systems. His research interests include data fusion, statistical signal processing, wireless sensor networks, the Internet of Things and machine learning.

Pierluigi Salvo Rossi (SM'11) was born in Naples, Italy, in 1977. He received the Dr.Eng. degree (*summa cum laude*) in telecommunications engineering and the Ph.D. degree in computer engineering from the University of Naples "Federico II", Italy, in 2002 and 2005, respectively. He is currently a Full Professor and the Deputy Head with the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. He is also a part-time Research Scientist with the Department of Gas Technology, SINTEF Energy Research, Norway.

Previously, he worked with the University of Naples "Federico II", Italy, with the Second University of Naples, Italy, with NTNU, Norway, and with Kongsberg Digital AS, Norway. He held visiting appointments with Drexel University, USA, with Lund University, Sweden, with NTNU, Norway, and with Uppsala University, Sweden.

His research interests fall within the areas of communication theory, data fusion, machine learning, and signal processing. Prof. Salvo Rossi was awarded as an Exemplary Senior Editor of the IEEE COMMUNICATIONS LETTERS in 2018. He is (or has been) in the Editorial Board of the IEEE SENSORS JOURNAL, the IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, the IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, the IEEE COMMUNICATIONS LETTERS and the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.