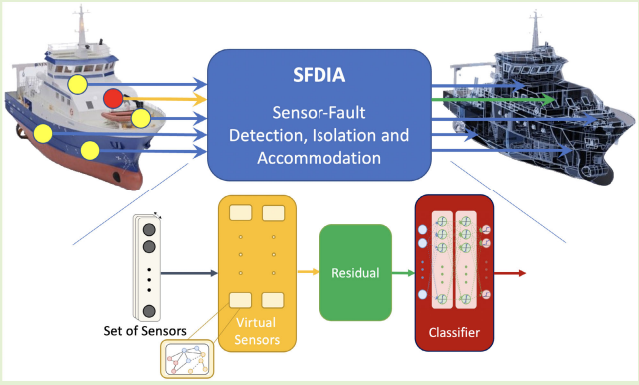
Deep Recurrent Graph Convolutional Architecture for Sensor Fault Detection, Isolation, and Accommodation in Digital Twins

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

***Abstract*—The rapid adoption of Internet-of-Things (IoT) and digital twins (DTs) technologies within industrial environments has highlighted diverse critical issues related to safety and security. Sensor failure is one of the major threats compromising DTs operations. In this article, for the first time, we address the problem of sensor fault detection, isolation, and accommodation (SFDIA) in large-size networked systems. Current available machine-learning solutions are either based on shallow networks unable to capture complex features from input graph data or on deep networks with overshooting complexity in the case of large number of sensors. To overcome these challenges, we propose a new framework for sensor validation based on a deep recurrent graph convolutional architecture which jointly learns a graph structure and models spatio-temporal interdependencies. More specifically, the proposed two-block architecture 1) constructs the virtual sensors in the first block to refurbish anomalous (i.e., faulty) behavior of unreliable sensors and to accommodate the isolated faulty sensors and 2) performs the detection and isolation tasks in the second block by means of a classifier. Extensive analysis on two publicly available datasets demonstrates the superiority of the proposed architecture over existing state-of-the-art solutions.**



***Index Terms*— Digital twin (DT), fault diagnosis, graph learning, Internet of Things (IoT), machine learning, neural networks, sensor validation.**

I. Introduction

UNDER the umbrella of Industry 4.0, digital twins (DTs) have garnered striking interest over the last few years through the process of industry digital transformation [1]. Fundamentally, a DT can be defined as a digital profile that mirrors a physical object or process, i.e., the physical twin (PT), and provides a bidirectional interaction between the physical and digital parts. Leveraging DTs, operators can simulate complex systems behavior, test/predict asset changes in specific scenarios, and remotely control/monitor/steer systems.

The explosion of the Internet of Things (IoT) and industrial IoT (IIoT) largely contributed to the success of DTs by enabling near real-time communications between DT/PT. Hundreds/thousands of sensors distributed all over the PT concertedly gather readings throughout a broad array of dimensions (i.e., features) and send them to the DT in order to run possible simulations, what-if analysis, study-specific scenarios and/or generate possible feed-backs/improvements on the PT. However, faulty data may lead to system instability and eventually jeopardize system reliability with possible noxious outcomes ranging from loss or critical damage to the asset (viz. financial and time loss) and/or environmental hazardous impact to serious injury to people or death in the worst case. Failure sources can be classified into three types [2], [3], [4], [5].

Manuscript received 30 August 2023; accepted 16 October 2023. Date of publication 25 October 2023; date of current version 30 November 2023. This work was supported in part by the Research Council of Norway through the Project SIGNIFY within the IKTPLUSS Framework under Project 311902. The associate editor coordinating the review of this article and approving it for publication was Dr. Peng Wang. *(Corresponding author: Pierluigi Salvo Rossi.)*

Hossein Darvishi was with the Department of Electronic Systems, Norwegian University of Science and Technology, 7491 Trondheim, Norway. He is now with Cognizant AI&A Nordics, Oslo, Norway (e-mail: hossein.darvishi@ntnu.no).

Domenico 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).

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

Digital Object Identifier 10.1109/JSEN.2023.3326096

1) *Hardware/Software Failure*—the sensor itself is inaccurate or faulty due to its bad quality, being out of life time, bad calibration, and/or software failure;
2) *Typical Harsh Environment/Condition*—the system operates under a setting in which sensor survival is difficult and its performance deteriorates rapidly;
3) *Malicious Cyber-Attack*—an attempt is perpetrated to abuse or take advantage of the system functionality.

To ensure the successful rollout of a DT, it is crucial to continuously monitor and regulate received sensory data before feeding the DT with them. In particular, sensor fault diagnosis systems, i.e., systems aiming at *three tasks* of sensor fault detection, isolation, and accommodation (SFDIA) [6], have been wielded to preserve the reliability and robustness of sensor-based systems. Utilizing multiple *physical redundancies* (i.e., three or more identical sensors) for each measuring parameter together with a voting scheme, is a straightforward approach for fault diagnosis [7], [8], but quite expensive. Alternatively, *analytical redundancy* takes advantage of the existing relationships between sensors and have been explored with increasing interest [9], [10], [11], [12].

Relying on available historical data, machine learning and deep learning have been largely adopted for a wide selection of fault diagnosis tasks [12], [13], [14].

Among machine learning approaches, support-vector machines (SVMs) were used to detect faulty sensors in wireless sensor networks (WNSs) but showed limitations with large/complex datasets [13], [15], principal component analysis was employed for IoT sensory systems [16], [17], and dynamic Bayesian networks were tested on intelligent transport systems, suffering on high-dimensional data [14].

Among deep learning approaches, recently different solutions have been explored for fault diagnosis in IoT systems, based on multilayer perceptron (MLP) neural networks (NNs) [9], [18], Recurrent NNs (RNNs) [19], [20], convolutional NNs (CNNs) [21], [22], and autoencoders (AE) [23]. More specifically, a modular-SFDIA (M-SFDIA) architecture based on MLP NNs was proposed in [9] and [24] within the framework of DTs. This architecture presented a layered SFDIA scheme built on a series of MLP estimators providing residual signals along with trustworthy substitutions (i.e., estimations) for faulty data, and a classifier performing detection and identification tasks upon the residual signals. The M-SFDIA architecture was designed to detect, identify, and accommodate only a single faulty sensor at once. An extended version of the M-SFDIA architecture employed a manifold decision logic as well as a controller unit in order to detect, isolate, and accommodate simultaneous faulty sensors and avoid propagation of faults into the architecture [12].

AEs are a type of NNs composed of an encoder and a decoder that can be used to learn compressed representations from input data into a latent-space representation [25], [26]. They are also suitable for fault diagnosis and an AE-based anomaly (viz. fault) detection and accommodation technique was developed for IIoT systems and described in [23].

Although data-driven approaches have received large attention in the recent years since they do not require an explicit formulation of the relationships between sensors (as opposed to model-based approaches, e.g., [27], [28]), they suffer several disadvantages: 1) the performance of basic machine learning methods heavily depends on the nonlinearity, dimensionality, and heterogeneity of the system; 2) shallow NNs suffer weak generalization, i.e., they are unable to properly capture complex features within the data; and 3) the computational complexity increases exponentially with the system network size and usually this is paired with a performance degradation.

DTs are expected to become in the near future more and more accurate counterparts of manifold physical assets, a pervasive monitoring in the physical space (i.e., via a massive number of sensors) becomes a mandatory requirement. However, such massive flow of sensor data can be only considered safe if an appropriate SFDIA is able to keep track of them and provide the digital space with reliable measurements anytime.

Accordingly, in this article, we propose a data-driven-based deep recurrent graph convolutional architecture for SFDIA of *large-scale IoT networks*. Recently, graph NNs (GNN) have gained significant attention as a promising graph-based paradigm to perform fault detection. GNNs are capable to exploit effectively both temporal and spatial correlations among neighboring nodes (sensors) in large-size IoT systems, thus providing excellent accuracy in fault diagnosis. Graph convolutional networks (GCNs) are feed-forward NNs with convolution operation generalized to graphs of arbitrary structure [29]. GCNs have been used successfully for drug synthesis, action recognition, image classification, link prediction, load prediction, and fault diagnosis [30], [31], [32], [33].

Although GNNs has been recently considered for anomaly detection [34], they are mostly unexplored within the SFDIA framework. Some recent works have explored GNN classifiers for fault detection and classification of power transformers [32], graph deviation networks (GDNs) for sensor anomaly detection [35], and adaptive graph convolutional recurrent network (AGCRN) for traffic forecasting [36].

In this work, we exploit the AGCRN in a *denoising configuration* (i.e., reconstruction of data from falsified input) as the building block for developing reliable virtual sensors. This configuration also assists the NN to better explore existing interdependencies among neighboring sensors. Subsequently, the residuals (i.e., the difference between the readings from the real sensors and the virtual sensors) are concatenated and processed by a classifier in order to detect and identify the presence of faulty sensor(s). Furthermore, the virtual readings are employed to accommodate the isolated (viz. identified) faulty sensor(s). Accordingly, the main contributions of this work are summarized as follows.

1) We propose the use of an enhanced GCN, termed AGCRN, to model virtual sensors. The AGCRN captures close-grained spatio-temporal correlations in graph data based on the two modules and a recurrent design.
2) To the best of our knowledge, this article is the first to propose the use of GCN-based design in the SFDIA framework. Our proposed deep recurrent graph convolutional architecture has capabilities for the detection, isolation, and accommodation of unknown fault types without any premodifications.

3) This work is also the first attempt to address and successfully perform all three tasks of detection, isolation, and accommodation of sensor faults within the SFDIA framework within the challenging scenario of large-scale IoT networks.
4) The performance of the proposed approach in terms of mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE) and probabilities of detection, false alarm, and correct identification is evaluated on *two publicly available* datasets [37], [38]. Datasets are falsified with synthetically generated faults (bias and drifts) and compared with alternative state-of-the-art techniques [9], [23], [35], [39]. Synthetically generated bias and drift faults have been considered as they represent hard (i.e., sudden) faults and soft (i.e., gradually appearing) faults, respectively. The datasets and NNs in the proposed architecture are publicly available, which helps reproducibility and further advances on the topic.

The remainder of this article is structured as follows. In Section II, preliminaries on graph convolution and GCNs are presented. Section III describes in detail the proposed SFDIA architecture, while Section IV provides the description of the datasets and the framework for fault generation. The performance comparison of the proposed scheme with other methods via numerical results is illustrated in Section V, followed by concluding remarks and possible future work in Section VI.

Notation: $\boldsymbol{I}_N$ denotes the identity matrix of size N; $\mathbf{1}_{a\times b}$ denotes the matrix of all ones of size $[a \times b]$; $\mathbf{0}_N$ denotes the null vector of length N; $\{\cdot\}^T$ refers to the transpose operator, $[\cdot;\cdot]$ refers to concate operation, $|\cdot|$ indicates the absolute operation, $\odot$ denotes the entry-wise (Hadamard) product, $\otimes$ denotes the tensor product (whose meaning is specified each time is adopted), $*$ denotes the convolution operator, $\|\cdot\|_p$ denotes the p-norm, $\in$ is the set membership, and $\mathcal{O}(\cdot)$ denotes the Landau notation. $\mathcal{U}(a,b)$ (resp. $\mathcal{U}_d(a,b)$) denotes a uniform (resp. discrete-uniform) probability density function (pdf) with support $[a,b]$ (resp. $\{a,a+1,\ldots,b\}$); $\mathcal{N}(b,c)$ (resp. $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$) denotes a Gaussian (resp. multivariate Gaussian) pdf with mean b and variance c (resp. with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$); $\mathcal{B}(p)$ denotes a Bernoulli distribution with parameter p.

II. Preliminary

This section introduces the necessary background on graph signal processing for the proposed SFDIA approach. Specifically, first in Section II-A, the notion of graph convolution is recalled. Then, in Section II-B, the actual implementation of graph convolutional layers is refreshed.

A. Convolution Operation on Graphs

The topological structure of a set of networked sensors can be described as an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E},A)$, where $\mathcal{V}$ is a finite set of N nodes (i.e., sensors), $\mathcal{E}$ is a set of edges that represents the connections between nodes, and $\boldsymbol{A}\in\mathbb{R}^{N\times N}$ is the adjacency matrix describing the connectivity of graph $\mathcal{G}$.

The *graph Laplacian* $\boldsymbol{L}\in\mathbb{R}^{N\times N}$ is a key operator in graph analysis [40], defined as $\boldsymbol{L}\triangleq(\boldsymbol{D}-\boldsymbol{A})$, namely the difference between the adjacency matrix $\boldsymbol{A}$ and the diagonal degree matrix $\boldsymbol{D}$ (where $d_{ii}=\sum_j a_{ij}$). The *normalized graph Laplacian* matrix $\boldsymbol{L}_{\mathcal{G}}=(\boldsymbol{I}_N-\boldsymbol{D}^{-(1/2)}\boldsymbol{A}\boldsymbol{D}^{-(1/2)})$ is a real symmetric positive semidefinite matrix with a complete set of orthonormal eigenvectors $\{\boldsymbol{u}_i\}_{i=1}^N\in\mathbb{R}^N$ (also known as Fourier functions) associated with real nonnegative eigenvalues $\{\lambda_i\}_{i=1}^N$ representing the frequencies of the graph. Moreover, the graph normalized Laplacian spectrum [41] is contained in the span of $\{\lambda_i\}_{i=1}^N\in[0,2]$. The graph normalized Laplacian is always diagonalizable by the Fourier basis $\boldsymbol{U}=[\boldsymbol{u}_1\ \cdots\ \boldsymbol{u}_N]\in\mathbb{R}^{N\times N}$ i.e., $\boldsymbol{L}_{\mathcal{G}}=\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, where $\boldsymbol{\Lambda}=\mathrm{diag}(\lambda_1,\ldots,\lambda_N)\in\mathbb{R}^{N\times N}$. The *Fourier graph transform* (GFT) $\mathcal{F}$ of a signal $\boldsymbol{x}\in\mathbb{R}^N$ is defined by the Fourier basis $\mathcal{F}(\boldsymbol{x})=\boldsymbol{U}^T\boldsymbol{x}$ and its inverse $\mathcal{F}^{-1}(\boldsymbol{x})=\boldsymbol{U}\mathcal{F}(\boldsymbol{x})$.

Spectral convolution on the graph $\mathcal{G}$ is defined [42] as the signal $\boldsymbol{x}$ filtered by graph filter $g_{\boldsymbol{\theta}}$, i.e.,

$$g_{\boldsymbol{\theta}}*\boldsymbol{x}\triangleq\mathcal{F}^{-1}\left(\mathcal{F}\left(g_{\boldsymbol{\theta}}\right)\odot\mathcal{F}\left(\boldsymbol{x}\right)\right)=\boldsymbol{U}\left(\boldsymbol{U}^T g_{\boldsymbol{\theta}}\odot\boldsymbol{U}^T\boldsymbol{x}\right)$$
$$=\left[\boldsymbol{U}\hat{g}_{\boldsymbol{\theta}}\left(\boldsymbol{\Lambda}\right)\boldsymbol{U}^T\right]\boldsymbol{x} \tag{1}$$

where $\hat{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\triangleq\mathrm{diag}(\boldsymbol{U}^T g_{\boldsymbol{\theta}})$ is the spectral graph filter (in diagonal matrix form) parameterized by $\boldsymbol{\theta}\in\mathbb{R}^N$ in the Fourier domain to avoid the elementwise operation, namely

$$\hat{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})=\begin{bmatrix}\hat{g}_{\theta_1}(\lambda_1) & 0 & \ldots & 0\\ 0 & \hat{g}_{\theta_2}(\lambda_2) & \ldots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \ldots & \hat{g}_{\theta_N}(\lambda_N)\end{bmatrix}. \tag{2}$$

B. Graph Convolutional Neural Network

The defined filtering operation $g_{\boldsymbol{\theta}}*\boldsymbol{x}$ has a *quadratic* computational complexity $\mathcal{O}(N^2)$ due to the matrix multiplication with the Fourier basis $\boldsymbol{U}$ in (1). Also, the eigen decomposition of $\boldsymbol{L}_{\mathcal{G}}$ is required (once) for carrying out spectral convolutions on $\mathcal{G}$. Then, for large graphs (i.e., $N\gg 1$), both these operations can become *computationally expensive*. Equally important, there is *no guarantee of spatial localization* [43] of the graph filter $\hat{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ (i.e., a nonsmooth filter). Spatial decay is an advantageous property to extract multiscale patterns. The graph filter $\hat{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ can become a nonsmooth spectral filter, while smoothness in the frequency domain corresponds to rapid spatial decay in the vertex domain. To overcome both these technical difficulties, the spectral convolution of (1) in GCN layers is obtained by 1) approximating the graph filter via Chebyshev polynomials and 2) then considering a first-order approximation to obtain a linear processing relationship. Further details on these approximation steps are reported in Appendix I.

Capitalizing on the aforementioned result, the GCN layer for a graph signal $\boldsymbol{X}\in\mathbb{R}^{N\times C}$ with C features per node and F filters is formulated as

$$\boldsymbol{Z}=\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{\Theta} \tag{3}$$

where $\boldsymbol{\Theta}\in\mathbb{R}^{C\times F}$ is the learnable-filter parameter matrix (i.e., the GNN weight matrix), the matrix $\widetilde{\boldsymbol{D}}^{-(1/2)}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-(1/2)}$ encodes

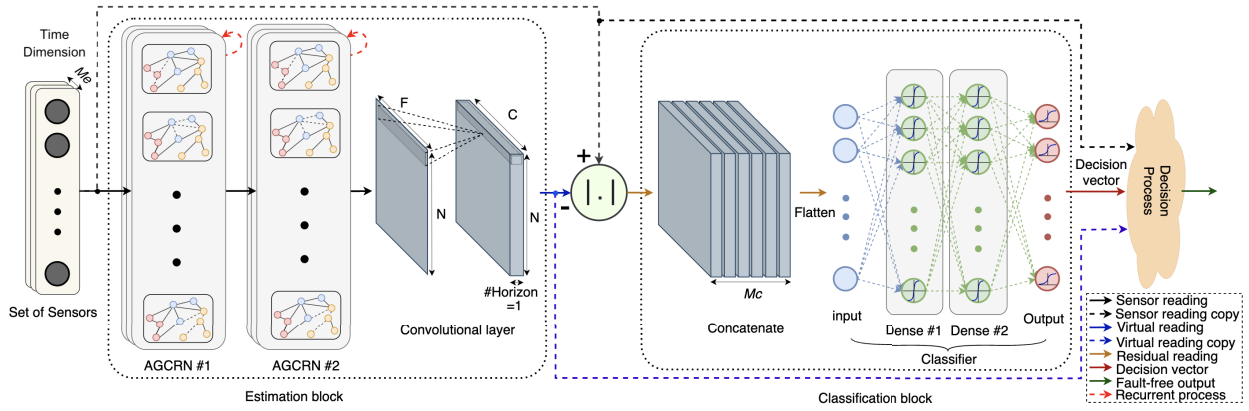


Fig. 1. Block diagram of the conceptual SFDIA framework, which consists of two main blocks: estimation and classification.

the graph structure via a functional which relates to the normalized graph Laplacian, and $\boldsymbol{Z} \in \mathbb{R}^{N\times F}$ is the convolved signal matrix. Accordingly, the computational complexity of the GCN operation is $\mathcal{O}(FC|\mathcal{E}|)$ due to a sparse multiplication (with $\widetilde{\boldsymbol{D}}^{-(1/2)}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-(1/2)}$). Hence, in general, the GCN layer can be expressed in its implicit form as

$$\boldsymbol{Z} = \mathrm{GCN}_{\boldsymbol{\Theta}}(\boldsymbol{X}; \mathcal{G}). \tag{4}$$

The aforementioned layer assumes the knowledge of the graph structure via the matrix $\widetilde{\boldsymbol{D}}^{-(1/2)}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-(1/2)}$.

III. System Model

A. Problem Definition

We consider a large-size sensor system made of N correlated sensors, i.e., $N \gg 1$, each measuring C parameters. The measured parameters of nth sensor at time k are denoted with $\boldsymbol{x}_n[k] \in \mathbb{R}^C$, while the matrix $\boldsymbol{X}[k] = \{\boldsymbol{x}_1[k], \ldots, \boldsymbol{x}_N[k]\}^T \in \mathbb{R}^{N\times C}$ collects the recordings of all the N nodes at same time instant. While observing the stream of measurements $\ldots, \boldsymbol{X}[k-1], \boldsymbol{X}[k], \boldsymbol{X}[k+1], \ldots$, a subset of these sensors (corresponding to the rows of these matrices) may be subject to weak and transient faults. The SFDIA problem consists of: 1) *detecting* fault conditions within the system (i.e., at least one sensor is subject to a fault); 2) *identifying* faulty sensors (i.e., which sensors are currently subject to faults); and 3) *accommodating* faulty measurements, i.e., replacing the faulty streams with estimated measurements via the concept of virtual sensors. Employing SFDIA is necessary to make DTs reliable when operating under faulty conditions.

B. Proposed SFDIA Architecture

As shown in Fig. 1, the proposed SFDIA architecture is made of *two* NN-based blocks: 1) the estimation block and 2) the classification block. Finally, the architecture is topped with 3) a threshold-based decision and accommodation logic. The input to the *estimation block* of the proposed architecture is the set of readings from all the sensors within a sliding window. The estimation block models virtual sensors of *all* the sensors within the system. The *classification block* provides probabilistic predictions on the faulty condition of each sensor on the basis of the residual between each sensor and its corresponding virtual sensor. The *decision process* detects faulty conditions and identifies faulty sensors: once the faulty sensor(s) is (are) identified, the proposed architecture isolates the faulty sensors (i.e., position in failure status) and accommodates their measurements with the associated virtual readings to the DT throughout the process. High-level specification for each block is provided in what follows.

1) Estimation Block: This block aims to model the sensors (i.e., design the virtual sensors) within the system and is composed of a single multidimensional estimator providing the estimates $\hat{\boldsymbol{X}}[k] \in \mathbb{R}^{N\times C}$ of the present readings (at time k). The estimator receives as input a series of previous (time-correlated) sensors readings $\{\boldsymbol{X}[k-1], \ldots, \boldsymbol{X}[k-M_e]\}$ over a window of size M_e (the input of this approach is arranged in a tensor of size $[N, C, M_e]$), a tunable hyperparameter of the proposed estimator. It is designed to capitalize the spatial correlation among sensors via the graph $\mathcal{G}$, i.e.,

$$\hat{\boldsymbol{X}}[k] = \boldsymbol{f}_{\boldsymbol{\varsigma}}(\boldsymbol{X}[k-1], \ldots, \boldsymbol{X}[k-M_e]; \mathcal{G}(\boldsymbol{\varsigma})) \tag{5}$$

where $\boldsymbol{f}_{\boldsymbol{\varsigma}}(\cdot) : \mathbb{R}^{N\times C\times M_e} \to \mathbb{R}^{N\times C}$ denotes the function model of the NN-based estimator which models the current sensors readings and $\boldsymbol{\varsigma}$ collects its trainable parameters. The notation $\mathcal{G}(\boldsymbol{\varsigma})$ in (5) underlines that we aim at learning *also* the graph structure of the system *during the training phase*.

2) Classification Block: As shown in Fig. 1, the classification block is made of an NN-based classifier meant to work in real-time to *detect* fault(s) and also *identify* the faulty sensor(s). To accomplish this task, the classifier relies on the concept of *residuals*, i.e., the absolute difference between sensors reading and their associated virtual reading, namely

$$\boldsymbol{\Delta}[k] = |\hat{\boldsymbol{X}}[k] - \boldsymbol{X}[k]|, \quad \boldsymbol{\Delta}[k] \in \mathbb{R}^{N\times C} \tag{6}$$

where the absolute operation $|\cdot|$ should be interpreted elementwise. We highlight that our novel SFDIA architecture relies on a *decoupled* design between estimation/classification blocks: Thus, other discrepancy measures other than (6) may be adopted without substantial changes in the subsequent layers.

Ideally, when the nth sensor operates in a faultless fashion at time k, the corresponding residual (i.e., the nth row of $\boldsymbol{\Delta}[k]$) is expected to be zero due to the perfect estimation. Indeed, the nth virtual sensor is designed to satisfy $\hat{\boldsymbol{x}}_n[k] \approx \boldsymbol{x}_n[k]$ in nominal conditions. Yet, in practice, there is always some amount of residual in fault-free conditions since the virtual sensors are

imperfect estimators of real sensors. Accordingly, the proposed classifier collects a concatenated sequence of residuals as inputs, namely $\{\boldsymbol{\Delta}[k], \boldsymbol{\Delta}[k-1], \ldots\}$, and based on the above inputs, a soft decision vector $\boldsymbol{d}[k] = [d_1[k] \ \cdots \ d_N[k]]^T$ is provided as the output at time k. Therein, each element of the decision output $d_n[k] \in [0, 1]$, $n = 1, \ldots, N$, refers to a pseudoprobability of the nth sensor to be faulty at the corresponding time instant. More specifically, the considered classifier is modeled as follows:

$$\boldsymbol{d}[k] = \boldsymbol{h}_{\boldsymbol{\vartheta}}(\boldsymbol{\Delta}[k], \ldots, \boldsymbol{\Delta}[k - M_c + 1]) \tag{7}$$

where $\boldsymbol{h}_{\boldsymbol{\vartheta}}(\cdot) : \mathbb{R}^{N \times C \times M_c} \to \mathbb{R}^N$ denotes the function model of the NN-based classifier and $\boldsymbol{\vartheta}$ collects the classifier trainable parameters. The model accepts residuals using a sliding window mechanism with a memory of size M_c, a tunable hyperparameter of the proposed approach.

3) Decision Process: The value of decision element $d_n[k]$ is assumed to represent *accurately* the architecture confidence in declaring the nth sensor to be faulty at time k, with $d_n[k] = 0$ (resp. $d_n[k] = 1$) being the utmost confidence on declaring the nth sensor nonfaulty (resp. faulty). Consequently, *fault detection* is performed by checking if any entries of the decision vector $\boldsymbol{d}[k]$ exceed a given threshold γ, namely $\max_{n=1}^{N} d_n[k] > \gamma$. Consistently, *fault identification* is based on the set of indices $\mathcal{I} \triangleq \{n \in \{1, \ldots, N\} : d_n[k] > \gamma\}$. Finally, the declared faulty sensors after identification are *accommodated* (viz. isolated and replaced) by their associated virtual sensors in real-time to preserve the DT functionality. More specifically

$$\boldsymbol{x}_s[k] \to \hat{\boldsymbol{x}}_s[k] \quad \forall s \in \mathcal{I} \tag{8}$$

where $\hat{\boldsymbol{x}}_s[k]$ denotes the sth row of $\hat{\boldsymbol{X}}[k]$, i.e., the sth virtual sensor. We underline that the proposed SFDIA architecture runs in "open loop," i.e., accommodated measurements are not fed back into the estimation block. This is to grant decoupled design and avoid complex transients when a fault is detected/identified.

The following subsections describe the NNs (including their training phase) implementing the estimation ($\boldsymbol{f}_{\varsigma}(\cdot)$, Section III-C) and the classification blocks ($\boldsymbol{h}_{\boldsymbol{\vartheta}}(\cdot)$, Section III-D), respectively.

C. NN-Based Estimation

In this work, the AGCRN layer is adopted as the stepping stone for the design of the estimation block (and thus model the whole set of virtual sensors) in the proposed SFDIA architecture [36]. Indeed, AGCRN addresses *three* strict *limitations* of GCNs, via the following *advancements*.

1) Node-Specific Patterns: GCN-based models are designed to effectively capture the shared spatial patterns (i.e., interdependencies) among sensors within the system. Indeed, having shared learnable-filter parameters $\boldsymbol{\Theta} \in \mathbb{R}^{C \times F}$ is quite useful to reduce the number of parameters while remaining on obtaining the prominent shared dependencies among sensors. Except for the shared patterns, the GCN fails to apprehend possible *diversified* node-specific patterns. On the contrary, assigning trainable parameters on each node level (i.e., a tensor $\boldsymbol{\Theta} \in \mathbb{R}^{N \times C \times F}$ with nonparametric dependence) would fit the bill, but unfortunately, drastically increases the network size. Hence, to reach a reasonable compromise, $\boldsymbol{\Theta}$ is factorized as $\boldsymbol{\Theta} = \boldsymbol{E}_g \otimes \boldsymbol{W}_g$, where: 1) $\boldsymbol{E}_g \in \mathbb{R}^{N \times l}$ is a node embedding matrix, where $l \ll N$ is the embedding dimension and 2) $\boldsymbol{W}_g \in \mathbb{R}^{l \times C \times F}$ is a weight pool tensor.[1] Similarly, the additive learnable bias matrix $\boldsymbol{B} \in \mathbb{R}^{N \times F}$ is factorized as $\boldsymbol{B} = \boldsymbol{E}_g \boldsymbol{B}_g$, where $\boldsymbol{B}_g \in \mathbb{R}^{l \times F}$ is the bias pool matrix. Specifically, we have

$$\boldsymbol{Z} = \left(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{X}\right) \otimes \left(\boldsymbol{E}_g \otimes \boldsymbol{W}_g\right) + \boldsymbol{E}_g \boldsymbol{B}_g \tag{9}$$

where the entries of $\boldsymbol{Z}$ are obtained by interpreting the tensor product $\otimes$ as $\{\boldsymbol{Z}\}_{ik} = \sum_j (\widetilde{\boldsymbol{D}}^{-(1/2)} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-(1/2)} \boldsymbol{X})_{ij} (\boldsymbol{E}_g \boldsymbol{W}_g)_{ijk}$, where $(\widetilde{\boldsymbol{D}}^{-(1/2)} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-(1/2)} \boldsymbol{X}) \in \mathbb{R}^{N \times C}$ and $(\boldsymbol{E}_g \boldsymbol{W}_g) \in \mathbb{R}^{N \times C \times F}$.

2) Learned Adjacency Matrix: The graph convolution operation [i.e., (3)] is completely dependent on the predefined adjacency matrix $\widetilde{\boldsymbol{A}}$ (as $\widetilde{\boldsymbol{D}}$ can be readily obtained from the former) to capture the spatial dependencies. The adjacency matrix is usually obtained by utilizing (intuitive) notions of similarity and/or distance functions [44], [45]. Unfortunately, the predefined graph generated in the aforementioned fashion is unable to contain domain knowledge of spatial dependencies and, equally important, is not related to the considered task. To this end, AGCRN learns the spatial dependencies by introducing an *embedding matrix* $\boldsymbol{E}_a \in \mathbb{R}^{N \times l_a}$. Such a matrix, together with its transpose $\boldsymbol{E}_a^T$, is used to learn *directly* the matrix $\widetilde{\boldsymbol{D}}^{-(1/2)} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-(1/2)}$, collecting the (graph-based) spatial information leveraged by a GCN layer, based on the equation

$$\widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-\frac{1}{2}} = \text{softmax}\left(\text{ReLU}\left(\boldsymbol{E}_a \boldsymbol{E}_a^T\right)\right) \tag{10}$$

where l_a is the node embedding dimension, ReLU$(\cdot)$ function is used to force a nonnegative matrix and softmax$(\cdot)$ function is utilized to normalize columnwise the final adaptive matrix.

3) Complex Temporal Correlations: AGCRN integrates the (node-specific and graph-adaptive) GCN layer with the concept of gated recurrent unit (GRU) [36] to capture also the complex and long-term temporal correlations. Hence, the AGCRN layer relies on (matrix-valued) *update* and *reset* gates to keep track of time dependence. As a result of the integration, the update equations for the two aforementioned gates also include a *spatial processing* which accounts for the graph structure. Such processing utilizes node-specific patterns and a learned adjacency matrix, both depending on *same embedding matrix* $\boldsymbol{E}$ to limit the complexity. Finally, the AGCRN output at time k is represented by the matrix $\boldsymbol{H}[k] \in \mathbb{R}^{N \times F}$, which is obtained as the weighted sum of its previous version ($\boldsymbol{H}[k-1]$) and a candidate activation matrix ($\hat{\boldsymbol{H}}[k]$). The weights are provided by the update gate, whereas the candidate activation matrix is obtained with an analogous spatial processing and leveraging the reset gate outcome. Details regarding the constitutive equations of the AGCRN layer are reported in Appendix II.

[1] We underline that in such case $\boldsymbol{\Theta}$ is obtained as a matrix-tensor multiplication between $\boldsymbol{E}_g$ and $\boldsymbol{W}_g$ by contracting the second dimension of the matrix with the first dimension of the tensor according to Einstein notation, i.e., $\{\boldsymbol{\Theta}\}_{ik\ell} = \sum_j \{\boldsymbol{E}_g\}_{i,j} \{\boldsymbol{W}_g\}_{j,k,\ell}$.

To achieve accurate virtual sensors, the corresponding NN-based estimator $\boldsymbol{f}_{\boldsymbol{\varsigma}}(\cdot)$ is thus composed as follows. We stack two AGCRN layers to be able to extract complex spatial and temporal correlations among sensors. The output (viz. input) of the first (resp. second) ACGRN layer is the evolution of the state matrix $\boldsymbol{H}_1[k-M_e], \ldots, \boldsymbol{H}_1[k]$. Conversely, from the output of the second AGCRN, we extract only the most recent form of the state matrix, i.e., $\boldsymbol{H}_2[k]$ (a usual practice when stacking multiple recurrent layers). A 2-D convolutional layer[2] is then applied at the output of the second AGCRN layer to directly project the representation from $\boldsymbol{H}_2[k] \in \mathbb{R}^{N\times F}$ (number of sensors by number of AGCRN features) to obtain the estimate $\hat{\boldsymbol{X}}[k] \in \mathbb{R}^{N\times C}$ (number of sensors by number of node parameters).

The MAE loss function is utilized to train the estimator, i.e.,

$$\mathcal{L}_{\text{est}}(\boldsymbol{\varsigma}) = \frac{1}{w}\sum_{j=0}^{w-1}\left\|\hat{\boldsymbol{X}}_j(\boldsymbol{\varsigma}) - \boldsymbol{X}_j\right\|_1 \tag{11}$$

where w is the number of samples in each batch, $\boldsymbol{X}_j$ denotes the fault-free readings of sensors and $\hat{\boldsymbol{X}}_j(\boldsymbol{\varsigma})$ refers to the corresponding estimate. The estimation block is learned in the so-called *denoising configuration*: the NN is then trained to predict the fault-free $\boldsymbol{X}[k]$ even in the presence of faulty sensors. Such configuration helps the model learn the latent representation of data and make a robust recovery of the clean original data. Finally, the adaptive moment estimation (Adam) optimization algorithm [46] is used to optimize the above loss.

The training process of the NN-based estimation block is summarized in the upper part of Algorithm 1 ("*procedure* ESTIMATOR").

D. NN-Based Classification

In this work, a deep feed-forward (viz. MLP) classifier was selected to implement the mapping $\boldsymbol{h}_{\boldsymbol{\vartheta}}(\cdot)$. Specifically, the input tensor $\{\boldsymbol{\Delta}[k], \ldots, \boldsymbol{\Delta}[k-M_c+1]\}$ is flattened into a single vector with NCM_c entries. The considered MLP is made of $H_c = 2$ hidden layers, each with $N_c = 2N$ neurons, where N denotes the number of sensors. Hyperbolic tangent activation function [i.e., $\tanh(\cdot)$] is applied to each neuron in both hidden layers. Finally, the MLP network is terminated with N neuron outputs with sigmoid activation function [i.e., $\sigma(\cdot)$] to provide a pseudoprobability output within $[0, 1]$. The N outputs are the entries of the soft decision vector $\boldsymbol{d}[k]$.

To train the classifier and make it able to implement both detection and identification tasks, a loss capitalizing *multitask learning* is employed. In the following, each learning task is associated with the classification of the operating condition for the corresponding sensor. Specifically, a *weighted sum* of the losses of the N binary (fault/no-fault) detection tasks associated with the unreliable sensors is minimized, i.e.,

$$\mathcal{L}_{\text{cl}}\left(\boldsymbol{\vartheta}_{\text{shared}}, \{\boldsymbol{\vartheta}_n\}_{n=1}^{N}\right) \triangleq \sum_{n=1}^{N}\lambda_n\,\mathcal{L}_n\left(\boldsymbol{\vartheta}_{\text{shared}}, \boldsymbol{\vartheta}_n\right) \tag{12}$$

[2]1-D convolutional layer and linear layers were observed to perform considerably worse than the 2-D convolutional layer.

Algorithm 1 Training Process of the Proposed SFDIA Architecture

procedure INITIALIZE
  Split train and validation sets;
  Generate the faults and add to train and validation sets;
  Random initialization of weights and biases for both networks;
procedure ESTIMATOR ▷ Denoising
  Input: falsified train set;
  Label: fault free train set;
  while Epoch number < Max epoch OR Max validation patience not reached **do**
    for Each epoch **do**
      Calculate MAE loss function;
      Update weights and biases using Adam optimization;
      Calculate validation loss;
      Update validation patience and epoch number;
procedure CLASSIFIER
  Compute virtual sensors readings of train and validation sets using Estimator;
  Compute residual signals of train and validation sets;
  Input: residual signals of train set;
  Label: the 0/1 representation of the true fault status;
  while Epoch number < Max epoch OR Max validation patience not reached **do**
    for Each epoch **do**
      Calculate BCE loss function;
      Update weights and biases using Adam optimization;
      Calculate validation loss;
      Update validation patience and epoch number;

where $\boldsymbol{\vartheta} = \{\boldsymbol{\vartheta}_{\text{shared}}, \boldsymbol{\vartheta}_1, \ldots \boldsymbol{\vartheta}_N\}$. In the above formula, the weight λ_n indicates the preference level of the nth task (i.e., detection of a fault at nth 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, $\boldsymbol{\vartheta}_{\text{shared}}$ represents the vector of *shared* parameters of the MLP common to all the N different tasks (i.e., those corresponding to the $H_c = 2$ hidden layers), whereas $\boldsymbol{\vartheta}_n$ indicates the vector of parameters which are *task-specific* for nth learning task (i.e., those corresponding to the nth output).

In this work, uniform weighting is adopted ($\lambda_n = 1/N$) and a binary cross-entropy (BCE) loss function for *all* the N binary tasks $\mathcal{L}_1(\cdot), \ldots, \mathcal{L}_N(\cdot)$ is used, i.e.,

$$\begin{aligned}\mathcal{L}_n^{\text{BCE}}&\left(\boldsymbol{\vartheta}_{\text{shared}}, \boldsymbol{\vartheta}_n\right)\\ &= -\frac{1}{w}\sum_{j=0}^{w-1}\Big\{y_n^j\,\ln d_n^j\left(\boldsymbol{\vartheta}_{\text{shared}}, \boldsymbol{\vartheta}_n\right)\\ &\qquad + (1-y_n^j)\ln\left(1-d_n^j\left(\boldsymbol{\vartheta}_{\text{shared}}, \boldsymbol{\vartheta}_n\right)\right)\Big\}\end{aligned} \tag{13}$$

TABLE I
SUMMARY STATISTICS OF DATASETS

Dataset	Time Range	Readings	# Sensors	# Edges	Data Range	Median	# Samples	Sample Rate
PeMSD8	01/07/2016 - 31/08/2016	Traffic Flow	170	277	0 - 1147	215	17856	5 minutes
Water Tank	—	Pressure	100	388	0 - 100	52	26998	—

where y_n^j is the 0/1 representation of the true (i.e., labeled) fault status and d_n^j denotes the entry of classifier output of nth sensor. Finally, w is the number of samples in each batch.

We underline that the overall loss is minimized by leveraging Nesterov-accelerated adaptive moment estimation (Nadam) optimization algorithm [47]. The training process of the NN-based classification block is summarized in the lower part of Algorithm 1 ("*procedure* CLASSIFIER").

IV. Data Description

In this study, we use *hybrid datasets*, i.e., datasets made of real-world measurements with synthetically generated faults superimposed. We point out that injecting synthetically generated fault into datasets with real-world measurements is a common practice in the evaluation of SFDIA systems (e.g., [13], [23], [48]) to assess the performance via a complete statistical analysis given the possibility of generating a large variation of faulty conditions[3] and, equally important, assessing accommodation performance. Details about the hybrid datasets used in this work are provided in the following. Also, we recall that datasets are presented in the form of time series which are the inputs to the considered algorithms.[4]

A. Measurements

As for the real-world measurements, we use two datasets, which are publicly available and have been widely used in previous research [50], [51]. Table I summarizes the statistics of each dataset. Sensors in both datasets collect a single parameter (flow rate or pressure), i.e., $C = 1$. Their detailed description follows.

1) PeMSD8: This real-world dataset contains traffic data collected in San Bernardino, CA, USA, from July to August 2016 [37]. The traffic data consist of all detector-based point data captured by the California Department of Transportation (CalTrans) performance measurement system[5] (PeMS). The traffic flow is collected by $N = 170$ sensors on eight roads in San Bernardino with a time interval of 5 min.

2) Water Tank: This dataset[6] collects measurements from a network of $N = 100$ water tanks connected through pipelines [38]. Tanks pressure is measured using pressure sensors to indicate the level of water in the tanks. When a tank's water level goes below a certain threshold, tank starts to refill until it is full. The flow rate between two connected tanks is a nonlinear function of the pressure and distance between the tanks. We use the first three measurements' logs of the dataset which roughly contains 27k samples.

[3]It is worth noticing that real faults are sporadic and hard to collect. Some efforts toward the collection of real-world datasets for SFDIA are discussed in [49] with focus on Carbon Capture and Storage.

[4]The considered GSP-based approaches do not require side information related to the sensor topology, they extract the spatial information directly from the time series.

[5]https://pems.dot.ca.gov/

[6]https://github.com/IndustrialNetwork/GraphDataset

B. Sensor Faults

As for synthetically generated sensor faults, we consider *bias*, *drift*, and *noise* fault types. Bias fault represents sudden faults, while drift fault well represents gradually appearing faults. Finally, noise faults well represent sensors subject to external disturbances.

1) Bias Fault: This fault type manifests as an additive constant vector $\boldsymbol{b} \in \mathbb{R}^C$ inserted to the normal operation of generic nth sensor for M consecutive samples, i.e.,

$$\boldsymbol{x}_n^b[k] = \begin{cases} \boldsymbol{x}_n[k] + \boldsymbol{b}, & 0 \le k - m < M \\ \boldsymbol{x}_n[k], & \text{otherwise} \end{cases} \tag{14}$$

where m denotes the starting time instant of the fault, $\boldsymbol{x}_n[k]$ and $\boldsymbol{x}_n^b[k]$ are the normalized healthy and possibly bias-faulty readings at time step k, respectively.

2) Drift Fault: For this type of fault, an additive term drifts gradually to the bias level vector $\boldsymbol{b}$ in M samples and then remains at the same value for K samples ($M > K$), namely

$$\boldsymbol{x}_n^d[k] = \begin{cases} \boldsymbol{x}_n[k] + \dfrac{(k-m+1)}{M}\boldsymbol{b}, & 0 \le k - m < M \\ \boldsymbol{x}_n[k] + \boldsymbol{b}, & M \le k - m < M + K \\ \boldsymbol{x}_n[k], & \text{otherwise} \end{cases} \tag{15}$$

where $\boldsymbol{x}_n^d[k]$ is the possibly drift-faulty readings at time k.

3) Noise Fault: This fault type is also considered to evaluate the performance of the proposed architecture in unseen fault scenarios. Specifically, a zero-mean additive Gaussian noise vector $\boldsymbol{w}[k] \sim \mathcal{N}(\boldsymbol{0}_C, \sigma_g^2 \boldsymbol{I}_C)$ is added for M consecutive samples, i.e.,

$$\boldsymbol{x}_n^g[k] = \begin{cases} \boldsymbol{x}_n[k] + \boldsymbol{w}[k], & 0 \le k - m < M \\ \boldsymbol{x}_n[k], & \text{otherwise} \end{cases} \tag{16}$$

where $\boldsymbol{x}_n^g[k]$ is the possibly noise-faulty readings at time step k and σ_g^2 represents the noise variance.

In this study, we train and test our proposed architecture under synthetically generated *bias* and *drift* fault types. Indeed, evaluation of the proposed architecture under these two fault types helps to highlight the proposed architecture *generality* in accommodating diversified faulty conditions. Additionally, in Section V-B, we will consider *noise faults* as an *unseen* fault type (i.e., not used during training) to test/evaluate the performance *robustness* of our proposal.

V. Numerical Results

In this section, we present the results of comprehensive experiments to demonstrate the effectiveness of the proposed SFDIA architecture. Also, the proposed architecture is compared with several state-of-the-art methods.[7]

[7]We modified the configuration of some baselines for sake of fair comparison (see Table II).

TABLE II
SETUP PARAMETERS OF THE PROPOSED ARCHITECTURE AND THE OTHER BASELINES. THE MAXIMUM NUMBER OF EPOCHS AND EARLY STOPPING PATIENCE FOR ALL ARCHITECTURE ARE SET TO 400 AND 15, RESPECTIVELY. WE RECALL THAT THE ARCHITECTURAL SPECIFICATION OF FCC AND THE ESTIMATOR BLOCK OF M-SFDIA/OM-SFDIA REFERS TO A SINGLE VIRTUAL SENSOR

Parameter	AE	FCC	M-SFDIA / OM-SFDIA		GDN	Proposed	
	AE & DAE		Est.	Clf.		AGCRN	Clf.
# Input Size	$[N \cdot M_e]$	$[N-1]$	$[(N-1)\cdot M_e]$ / $[N-1, M_e]$	$N \cdot M_c$ / $[N, M_c]$	$[N, M_e]$	$[N, M_e]$	$[N \cdot M_c]$
# Output Size	$N \cdot M_e$	1	1	N	N	N	N
# Hidden Layers	5	8	1	2 / 1	2 GNN & 1 Dense	2 AGCRN & 1 Conv2d	2
# Nodes per Hidden Layer	$(9; 6; 3; 6; 9)\cdot 10^2$	1	10	N / 15	128	64 (# GRU)	$2\cdot N$
Output Activation	Linear	Bipolar-Sigmoid	Linear	Sigmoid	Linear	Linear	Sigmoid
Hidden Layer Activation	Sigmoid	Bipolar-Sigmoid	Tanh	Tanh	Softmax & (Leaky)ReLU	Sigmoid & Tanh	Tanh
Optimizer/Learning Alg.	RMSprop	Neuron-by-Neuron	Nadam	Nadam	Adam	Adam	Adam
Loss Function	MSE	Sum Squared Error	MSE	BCE	MSE	MAE	BCE
Batch Size	20	20	20	100 / 200	128	64	32
Learning Rate	10^{-3}	10^{-3}	$4\cdot 10^{-4}$	10^{-3}	10^{-3}	$3\cdot 10^{-3}$	$2\cdot 10^{-4}$

A. Experimental Setup

1) Preprocessing: We split each dataset into train, validation, and test sets with a split ratio of 60% : 20% : 20%, respectively. We perform a data standardization before training the networks. We normalize data to the span of [0, 1] to avoid polarization during the training phase using a min-max scaling.

2) Baselines: We compare our SFDIA architecture against *five* state-of-the-art SFDIA and anomaly detection methods.

1) *AE*-based architecture addressed in [23] is a *two-stage* approach of a (standard) AE to learn data patterns among sensors for fault detection, and a denoising AE (DAE) to clean (accommodate) faulty data. The identification task for AE architecture was not advised in the original work. For the sake of fair comparison and to enable this architecture to perform identification task, we changed its decision logic. Here, we tracked the squared error (namely $\boldsymbol{e}_{AE,s}$) between the respective input and output of each sensor s. Later, this error is compared with a predefined threshold δ, enabling the AE method to both detect and identify the faulty sensors.[8] The input of this approach is $\boldsymbol{X}[k-1], \ldots, \boldsymbol{X}[k-M_e]$, arranged in a single vector of length $N \cdot M_e$ (i.e., different sensors at different time steps are flattened together).
2) *Fully connected cascade* (FCC) NN is used in a modular architecture to model the virtual sensors [39]. FCC NN is chosen instead of the popular MLP NN due to efficiency in terms of number of neurons and input size required for the SFDIA problem. Fault detection and identification are performed by evaluating the residual between each sensor and the corresponding FCC NN estimate. The input of this approach is $\boldsymbol{X}[k-1]$ (i.e., time dependence is not exploited), arranged in a vector of length N.
3) *M-SFDIA* proposal is a modular architecture which is able to detect and isolate *only* a single faulty sensor at a time [9]. Nonetheless, for the sake of a fair comparison, we used our decision logic (see Section III-B.3) to enable the M-SFDIA architecture to detect, isolate, and accommodate multiple simultaneously faulty sensors. The input of this approach is $\boldsymbol{X}[k-1], \ldots, \boldsymbol{X}[k-M_e]$, arranged in a vector of length $N \cdot M_e$ (i.e., different sensors at different time steps are flattened together).
4) *Optimized M-SFDIA* (OM-SFDIA) is an optimized class of M-SFDIA architecture which selects the best configuration of NN modules for SFDIA among various variants [20]. OM-SFDIA is enhanced to handle more complex spatio-temporal patterns in the data by using GRU model as virtual estimators and a CNN model for the classifier. The input of this approach is $\boldsymbol{X}[k-1], \ldots, \boldsymbol{X}[k-M_e]$, arranged in a matrix of size $[N, M_e]$.
5) *GDN* is a novel attention-based approach, which detects anomalies from a learned graph of relationships between sensors [35]. The GDN method was merely designed for anomaly detection purposes. We used our decision logic (see Section III-B.3) upon the GDN graph attention-based output to enable this method to detect and isolate multiple simultaneously faulty sensors. The input of this approach is $\boldsymbol{X}[k-1], \ldots, \boldsymbol{X}[k-M_e]$, arranged in a matrix of size $[N, M_e]$ (i.e., similar to our proposal).

We stress that all the aforementioned baselines (except for FCC, which neglects time dependence in its formulation [39]) leverage *the same exact input as the proposed SFDIA architecture.* Furthermore, we remark that, given the modularity of FCC, M-SFDIA, and OM-SFDIA, a *dedicated* NN is trained for each virtual sensor. Accordingly, the actual input for nth virtual sensor NN is obtained by replacing $\boldsymbol{X}[k-m]$ with $\boldsymbol{X}_{-n}[k-m]$ (namely the same vector after removing the contribution from nth sensor) in these architectures while retaining the flattening rationale described. Finally, we also compared the proposed architecture with a SVM-based classifier [13]. Surprisingly, the SVM method entirely failed in detecting the faults on both datasets, thus we do not report those performance in the following.

3) Parameters Settings and Implementation Details: The selected sliding window size for the estimator input is

[8] Similar to our proposed method, $\max_{s=1}^{N} \boldsymbol{e}_{AE,s}[n] \gtrless \delta$ is used for detection, and for the identification, the set of identified faulty sensors is obtained as $\mathcal{I}_U \triangleq \{s \in \{1, \ldots, N\} : \boldsymbol{e}_{AE,s}[n] > \delta\}$. It is worth noting that numerical results (not shown for brevity) based on the original detection logic as [23] showed worse performance than the considered variant due to the inability to cope with weak (and transient) faults.

$M_e = 12$ and for the classifier input is $M_c = 6$ based on a search over the grid $\{3, 6, 9, 12, 15\}$. We also choose some hyperparameters of the proposed architecture and other baselines using a grid search. To avoid models overfitting, we use early-stopping criteria to train models, i.e., we stop the training process once the validation loss stops decreasing for 15 consecutive epochs or maximum number of epochs achieved. More details about setup parameters of the proposed architecture and the baselines can be found on Table II.

We implement our architecture using Pytorch version 1.12.0 on MacBook pro M1 CPU 2.1–3.2 GHz with 16 GB memory, GDN architecture using Pytorch Geometric Library [52] version 2.0.4, and other baselines using Keras Python API running on TensorFlow version 2.9.2.

4) Fault Generation: To represent weak faults, the fault level b (note that $\boldsymbol{b}$ reduces to a scalar since $C = 1$) is generated as $b = (2s_b - 1) \cdot a_b$ where $s_b \sim \mathcal{B}(1/2)$ and $a_b \sim \mathcal{U}(0.2, 0.4)$. The variance for noise faults is similarly generated as $\sigma_g^2 \sim \mathcal{U}(0.2, 0.4)$. The fault lengths are specified via the parameters M and K [see (14), (15) and (16)], which are assumed uniformly distributed as $M, K \sim \mathcal{U}_d(3, 11)$ to represent *transient faults*. The random distribution of faults helps both the estimator and classifier to better generalize during the training phase and prevents focusing on a specific fault level/length. In the fault generation process, *up to* $P = 3$ *concurrent faulty sensors were considered* to evaluate the robustness of the proposed architecture against simultaneous faults in terms of detection and identification. Finally, we consider a fault rate (ratio between the number of faulty and nonfaulty samples in the datasets) $F_R \approx 10\%$. It is worth mentioning that the parameters are selected to show the potentiality of handling multiple faults, however, a complete analysis about the impact of the fault ratio on the performance is beyond the scope of the article.

B. Results and Analysis

Virtual sensors with embedding dimensions of $l \in \{1, 2, 3, 4, 5\}$ have been trained and compared. In detail, we utilized common metrics of MAE, RMSE, and MAPE to measure the performance of virtual sensors in Fig. 2 for each dataset. The dimension of the node embedding directly impacts the learned graph property. Results highlight that the proposed architecture maintains relatively stable performance for all the tested embedding dimensions, which illustrates the robustness of the proposed architecture. In general, larger embedding dimensions would relatively improve the performance, while excessively increasing the embedding dimension would inflate the number of the trainable parameter and causes overfitting which deteriorates the performance. The node embedding dimension $l = 2$ is considered acceptable to balance the architecture's performance and complexity on both datasets, thus in the following, we will refer to this specific configuration.

Virtual sensors performance of our proposal versus other baselines are reported in Table III for both the considered datasets. Overall, our proposed method clearly marks the best performance in all metrics as summarized in Table III. Underlined values refer to the best-performing baselines on each metric. We can observe that both the graph-based methods,

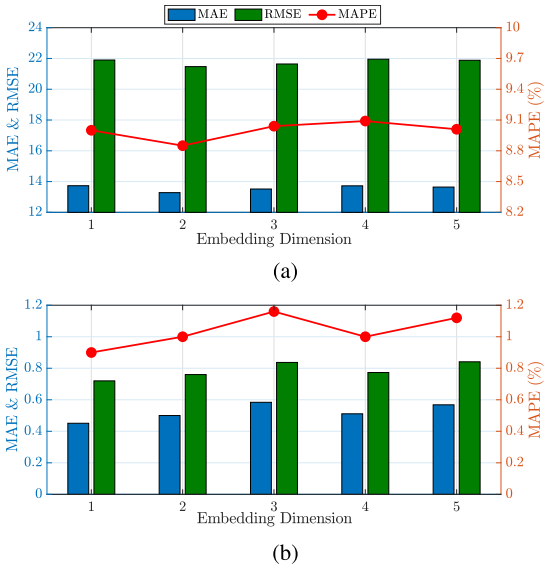


Fig. 2. Virtual sensors performance sensitivity to different embedding dimensions l in terms of MAE, RMSE, and MAPE. (a) PeMSD8. (b) Water Tank.

TABLE III
VIRTUAL SENSORS PERFORMANCE

Datasets	Models	Metrics		
		MAE	RMSE	MAPE [%]
PeMSD8	AE	26.52	38.99	12.76
	FCC	20.64	33.81	14.41
	M-SFDIA	20.63	33.83	19.76
	OM-SFDIA	18.84	33.35	14.64
	GDN	24.24	36.57	14.68
	Proposed	**13.28**	**21.47**	**8.85**
	Gain [%]	+29.51	+36.50	+30.64
Water Tank	AE	12.37	16.38	24.57
	FCC	15.50	20.00	30.77
	M-SFDIA	11.98	16.68	23.91
	OM-SFDIA	12.47	16.67	24.91
	GDN	6.63	9.12	14.35
	Proposed	**0.50**	**0.76**	**1.00**
	Gain [%]	+92.25	+91.67	+93.03

GDN method and our proposal, outperform other baselines on the Water Tank dataset, which indicates the effectiveness of GCN-based architecture in capturing spatial correlations. Nevertheless, the proposed architecture illustrates significant improvements (i.e., above 90% over GDN), thanks to its recurrent design. Better virtual sensors also imply higher *accommodation performance*, since these virtual measurements replace the real faulty sensors measurements upon classifier identification. There are no existing baselines that are as stable as our proposal. For instance, AE illustrates reasonably low MAPE on the PeMSD8 dataset, but failed on other metrics and the other dataset. On the contrary, our proposal shows *reliable estimations in all cases*.

Focusing our investigation toward *detection and identification* (viz. isolation) performance, in Fig. 3, we report the receiver operating characteristic (ROC) curves, which depicts

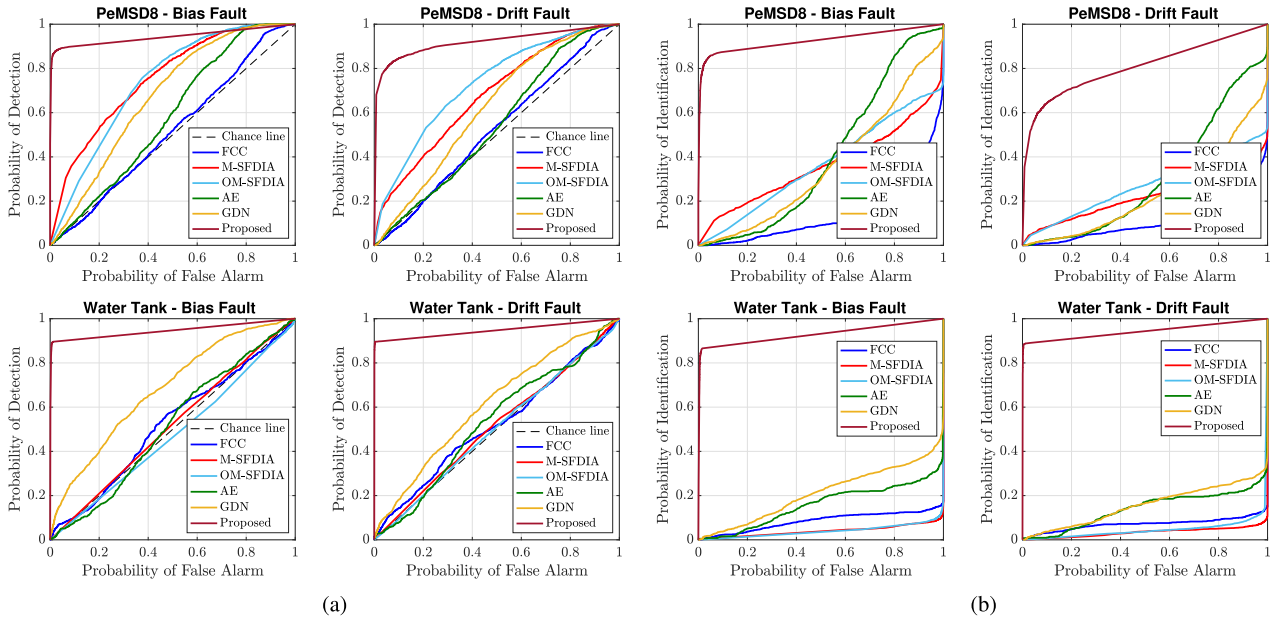


Fig. 3. ROC curves on two benchmark datasets. (a) and (b) Detection performance, where the chance line is included, and the identification performance, respectively. Curves closer to the top-left corner indicate better performance (the closer the curve to the chance line, the less accurate the detection performance).

the probability of (a) detection and (b) identification vs. the probability of false alarm. The probability of identification indicates the probability that SFDIA architecture correctly identifies the actual faulty sensor(s). Hence, the probability of identification is *upper bounded* by the corresponding detection probability, given the same false-alarm rate value. The proposed architecture significantly outperforms all the baseline methods on both datasets, demonstrating its capability to detect and isolate sensor faults on graph data. The main reason behind this is that our proposal captures the faults and sensors' patterns by jointly utilizing spatio-temporal correlations due to its graph convolutional and recurrent design. The AE, M-SFDIA, OM-SFDIA, and FCC methods, regardless of their approved performance on small-size sensory networks [9], [20], [23], [39], show relatively poor detection performance because these methods have limited capability to discriminate faults from high-dimensional graphs. We notice that all the baseline methods almost failed to identify the correct faulty sensor(s), while the proposed architecture identifies faulty sensor(s) with bold confidence on both datasets. Moreover, our proposal obtains more considerable performance gains on detection and identification in the water tank dataset compared to which in the PeMSD8 dataset. We observe that the Water Tank dataset has a more spatial connection degree (# edges $=$ 388 and # sensors $=$ 100) than the PeMSD8 (# edges $=$ 277 and # sensors $=$ 170) dataset, which may lead to stronger spatial correlations.

Delving into the detection and identification performance of the proposed architecture, Fig. 4 presents the details of its decision behavior on both datasets *for a selected threshold* γ. Furthermore, we considered a single simultaneous fault in the present analysis, namely $P = 1$. The reason was to investigate the identification performance in a *classification form*, so as to discriminate among $(N + 1)$ classes, corresponding to a fault event on each of the N sensors or no fault condition. This assumption allows assessing whether there are relevant miss-identification patterns (i.e., a correct fault detection event is declared, but the latter is associated with the wrong sensor). There are several algorithms for threshold selection, subject to different criteria. We choose Youden index J, which selects the threshold γ subject to maximization of the vertical distance between the chance line and the detection probability P_d point on the ROC curve [53], i.e.,

$$J = \max_{\gamma}(P_d - P_f) \tag{17}$$

where P_f is the probability of false alarm. Details of the selected probabilities of detection and false alarm using the Youden index are shown in Fig. 4(a). From inspection of the corresponding confusion matrices, although the SFDIA architecture miss-detects some faulty sensors, i.e., $\approx$ 23% of faults in the worst case on PeMSD8 under drift fault, it identifies the actual faulty sensor (viz. class) with excellent precision upon detection [see Fig. 4(b)].

As a complementary analysis over unsupervised performance of the proposed architecture, in Figs. 5 and 6, we analyze the proposed architecture performance in situations that the architecture is not specifically trained for (i.e., without any supervision). In Fig. 5, we trained both NNs in our SFDIA architecture on bias fault type, while tested its (a) detection and (b) identification performance on *unseen* drift and noise fault types. Surprisingly, both detection and identification performance on unseen fault types are relatively close to the performance on trained fault type. This is basically because the proposed technique models the virtual sensors in the denoising configuration, and this helps the estimator to focus on sensors' interdependencies and sensor-specific patterns rather than focusing on fault type patterns. Moreover, perfect virtual sensors result in interpretable residual signals which further help the classifier to easier differ between faulty

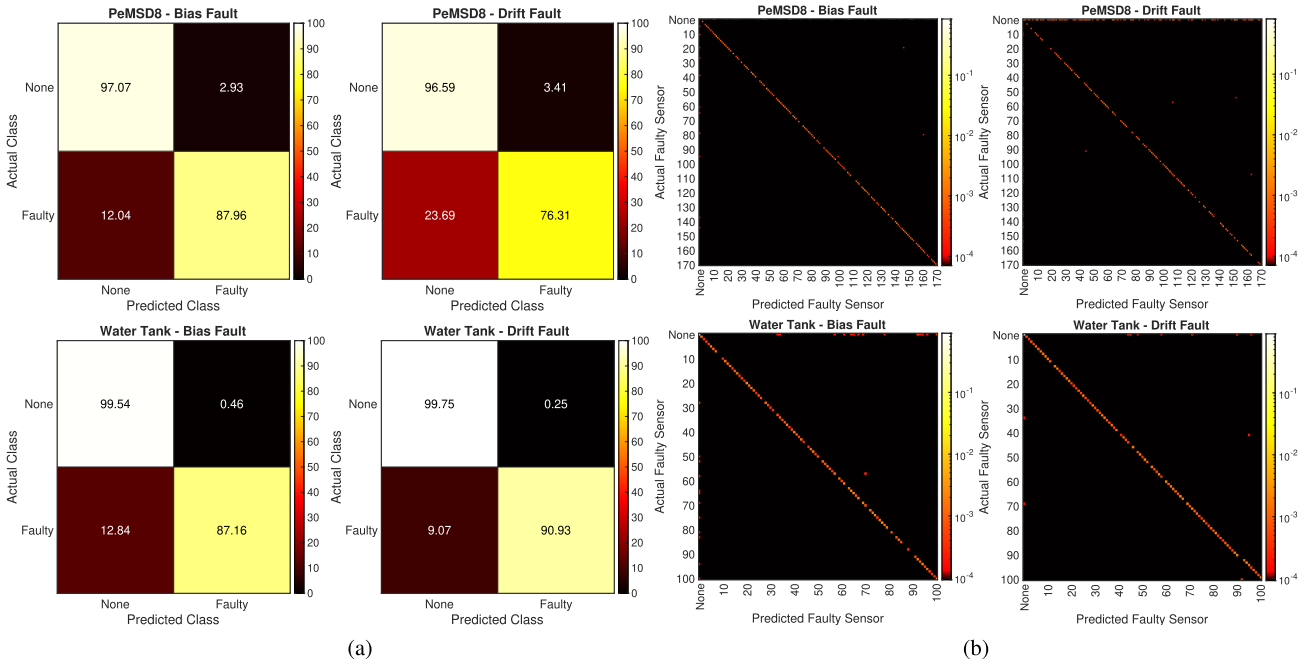


Fig. 4. Confusion matrices of the proposed architecture. Note that the log scale is used in (b) to evidence small probabilities. *y*- and *x*-axis numbers in (b) refer to the corresponding sensors in each dataset. "None" label refers to no-faulty sensor class. (a) Detection. (b) Classification.

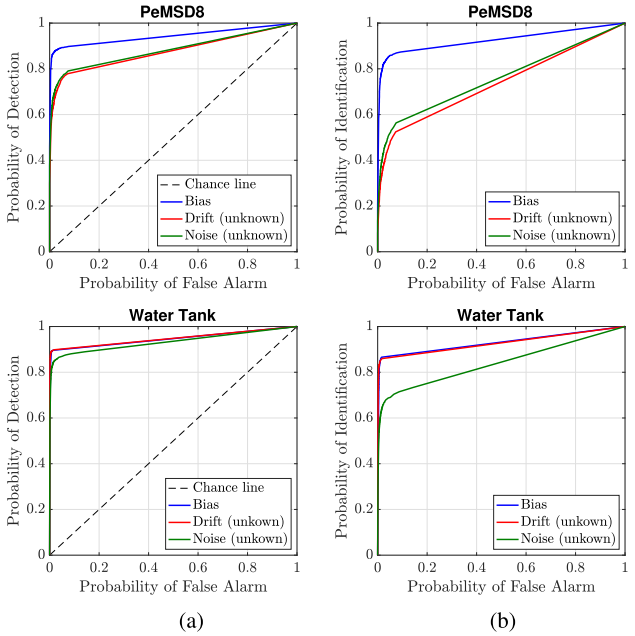


Fig. 5. Unsupervised (a) detection and (b) identification performance of the proposed architecture over unknown fault types (i.e., drift and noise fault) in terms of ROC curves. Both estimator and classifier are trained over the bias fault type.

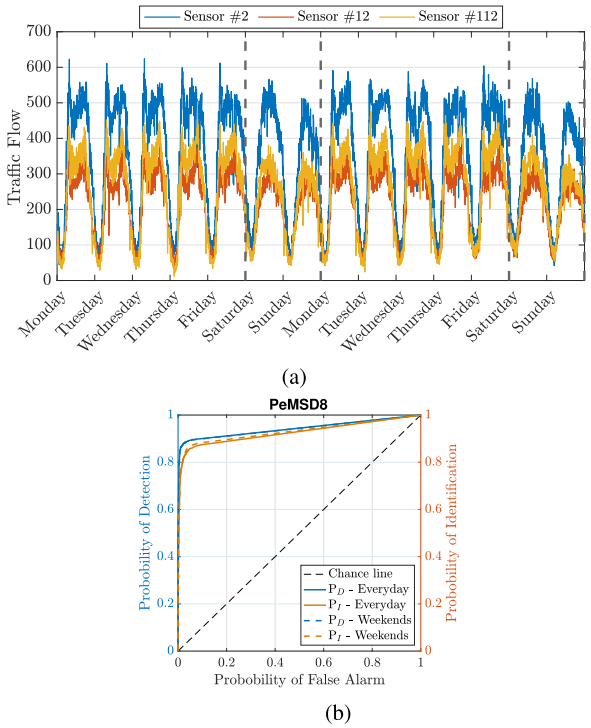


Fig. 6. (a) Traffic flow over three zones (i.e., captured by three sensors) over two weeks starting from the 11th of July, 2016 on the PeMSD8 dataset. (b) Detection and identification performance of the proposed architecture trained over bias fault on weekdays of the PeMSD8 dataset and tested on weekends in terms of ROC curves. Black dash lines in (a) highlight the weekends.

and nonfaulty residual patterns. To further investigate the proposed architecture capabilities in unseen (i.e., unsupervised) scenarios, Fig. 6 shows the results related to training the SFDIA architecture on weekdays and testing it on weekends for the PeMSD8 dataset. Fig. 6(a) visualizes how the patterns differ on weekdays compared to those on the weekends. It is apparent that three randomly chosen sensors record lower traffic flows on the corresponding streets during weekends and slightly shifted rush (peak) hours with respect to the patterns on weekdays. Although, despite obvious changes in the patterns between weekdays and weekends, the proposed architecture exhibits stable detection and identification performance in comparison to the case that the proposed architecture trained on both weekdays and weekends. Again, this is due to

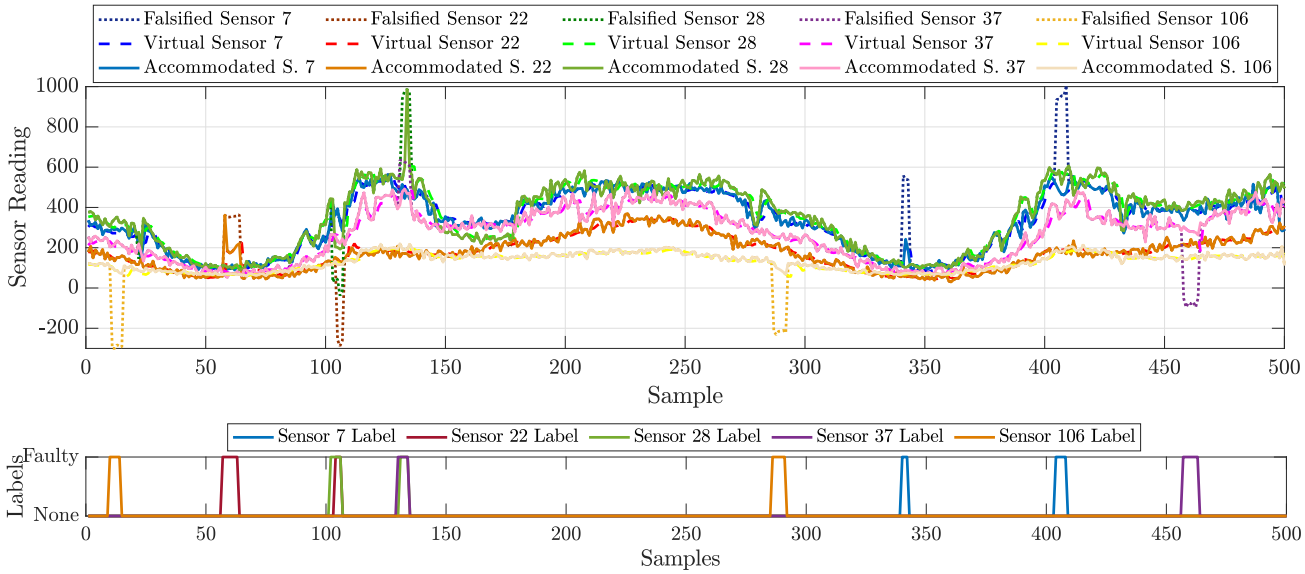


Fig. 7. Output at different stages of the proposed SFDIA architecture on five selected sensors (no. 7, 22, 28, 37, and 106) of PeMSD8 dataset with bias faults ($P_f = 10^{-2}$) for approximately two days of the test set.

TABLE IV
NUMBER OF NNS' TRAINABLE PARAMETERS

Dataset	AE		FCC	M-SFDIA / OM-SFDIA		GDN	Proposed	
	AE	DAE		Est.	Clf.		AGCRN	Clf.
PeMSD8	5.11M	5.11M	261K	3.45M / 925K	521K / 174K	25K	75K	521K
	In total = 10.23M			3.97M / 1.1M			596K	
Water	3.60M	3.60M	91K	1.19M / 334K	180K / 60K	16K	75K	180K
Tank	In total = 7.21M			1.37M / 394K			255K	

TABLE V
TRAINING AND INFERENCE TIME FOR DIFFERENT BASELINES. THE TRAINING TIME (MINUTES) IS FOR THE OVERALL TRAINING PHASE, WHILE THE INFERENCE TIME (MILLISECONDS) IS PER INPUT SAMPLE

Dataset	Op. Phase	AE	FCC	M-SFDIA	OM-SFDIA	GDN	Proposed
PeMSD8	Train	8min	48min	52min	903min	33min	440min
	Inference	1ms	23ms	29ms	100ms	3ms	11ms
Water	Train	7min	26min	40min	387min	29min	344min
Tank	Inference	$<$ 1ms	12ms	9ms	20ms	2ms	7ms

capability of the proposed architecture to learn fine-grained general spatio-temporal inter-dependencies between sensors.

To visually assess the accommodation capabilities of our proposal, Fig. 7 shows the output of the proposed SFDIA architecture for almost two days of the test set. More specifically, the bottom plot of the figure monitors the faults on five selected sensors on the PeMSD8 dataset in the case of bias faults. The proposed architecture successfully detects and identifies all faults in the corresponding sensors with virtually no delay in the system. Results indeed highlight only two miss detected points (i.e., false negatives) at samples 58 and 134 on the sensors 22 and 28, respectively. As evident from the top plot of Fig. 7, after fault identification, the proposed architecture accommodates isolated faulty data with their estimates from virtual sensors to ensure the fault-free performance of the system. In this snapshot, we have two overlapped (i.e., simultaneous) faulty sensor incidents for the samples $\in (100, 150)$. Nonetheless, the proposed architecture manages to handle this scenario and successfully identifies and accommodates the faulty samples.

Finally, to investigate the training complexity of the proposed architecture, in Table IV, we report the number of trainable parameters of our proposal in comparison to the baselines. Results highlight the intermediate complexity of the proposed architecture. Although the proposed architecture has much more trainable parameters in comparison to the GDN method, our architecture is capable to perform all three tasks of the SFDIA framework. Interestingly, the AE deep network and M-SFDIA methods have a considerably high number of trainable parameters, thus their instability in detecting faults to some extent. It also reveals the exponential complexity increment of the deep learning solutions over large-size networked IoT systems. The training complexity of the proposed architecture is balanced with respect to its perfect SFDIA capabilities. Table V presents a comparative analysis of the speed metrics across different baselines in terms of both *training* and *inference times*. Training time is the time taken by a model to train on a dataset, and the inference time denotes the time taken for the forward propagation of a single sample. Notably, our proposed architecture exhibits a relatively extended training time when contrasted with the baseline models, which can be attributed to its recurrent design. However, it is important to underscore that the longer training time does not pose a significant impediment in practical scenarios (training can happen on powerful cloud-based servers, which effectively mitigates the impact of this temporal aspect). Conversely, inference time assumes greater importance as it reflects the amount of time it takes for a model to process new data and make a prediction/decision. As evident from Table V, the proposed architecture demonstrates an inference

time of approximately 10 ms to process the input sample on the given hardware, which underline its suitability for real-time processing and decision-making tasks.

VI. Conclusion and Future Work

We tackled the SFDIA problem of large-size networked IoT systems via a deep learning approach. Our work represents an opening gate toward transferring reliable data into DTs of large-size sensor systems/networks. In this article, we proposed a *two-block* architecture for SFDIA framework. In the first block, an estimator models virtual sensors (by capitalizing both spatial and temporal dependence via graph-convolutional recurrent layers) and provides replacements for the identified faulty sensors within the system. In the second block, a lightweight (MLP-based) classifier detects and identifies the faulty sensors based on residuals. We provided a wide numerical analysis for comprehensive evaluation and comparison of the proposed architecture with other state-of-the-art methods, and also pointed out the unexplored groundwork for SFDIA advances on the large-scale networked IoT systems. The numerical analysis of the proposed SFDIA architecture highlighted performance gains ranging approximately from 30% to 90% (in terms of MAE, RMSE, and MAPE) in the case of virtual sensor design over the state of art. Equally important, results concerning detection and identification rates highlighted improvements larger than 40% (for a $P_f = 10^{-1}$) over existing SFDIA architectures for *both* the datasets when *both* the fault types considered were superimposed. It is worth highlighting that the denoising design and the classification upon the residuals empower the proposed architecture to maintain its performance under unseen fault types. Accordingly, our proposal attained larger than 80% detection (resp. 70% identification) probability for a $P_f = 10^{-1}$ in the case of (unseen) drift and noise faults on the Water Tank dataset.

Future work will focus on two main aspects: 1) including dynamic risk analysis in the design of IIoT systems in order to meet safety requirements when deploying DTs for safety-critical applications and 2) investigating nonstationary scenarios and the impact of diversity and redundancy in the graph.

Appendix I
Spectral Convolution in GCN Layers

This appendix provides the details of how the spectral convolution of (1) in GCNs can be approximated as a linear processing of the input via a matrix which encodes the graph structure. This is accomplished by considering a Chebyshev polynomial approximation for the spectral graph convolution and then truncating this series to the first order.

A. Chebyshev Polynomial Approximation

First, to tackle the localization problem, $\hat{g}_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ can be approximated by a truncated expansion up to order K of Chebyshev polynomials [54] $\{T_k(x)\}_{k=0}^{K}$, namely $\hat{g}_{\boldsymbol{\theta}'}(\boldsymbol{\Lambda}) \approx \sum_{k=0}^{K} \theta'_k \boldsymbol{T}_k(\widetilde{\boldsymbol{\Lambda}})$. In the latter approximation, $\widetilde{\boldsymbol{\Lambda}}$ denotes the rescaled matrix $\widetilde{\boldsymbol{\Lambda}} \triangleq (2\boldsymbol{\Lambda}/\lambda_{\max} - \boldsymbol{I}_N)$ (where $\lambda_{\max}$ denotes the largest eigenvalue of $\boldsymbol{L}_{\mathcal{G}}$), while θ'_k represents the kth Chebyshev coefficient ($\boldsymbol{\theta}' \in \mathbb{R}^{K+1}$). The Chebyshev polynomials can be efficiently computed via the recurrence relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0(x) = 1$ and $T_1(x) = x$. Although the graph filter is now K-localized with respect to the Kth-order polynomials of the Laplacian, the learning complexity is still not addressed because of the multiplication of the eigenvector matrix $\boldsymbol{U}$. A solution to this problem is to directly learn the function of the normalized Laplacian $g_{\boldsymbol{\theta}'}(\boldsymbol{L}_{\mathcal{G}})$ [42]. Indeed, exploiting $(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T)^k = \boldsymbol{U}\boldsymbol{\Lambda}^k\boldsymbol{U}^T$, the equality $\boldsymbol{U}\boldsymbol{T}_k(\widetilde{\boldsymbol{\Lambda}})\boldsymbol{U}^T = \boldsymbol{T}_k(\widetilde{\boldsymbol{L}})$ holds, where $\widetilde{\boldsymbol{L}} \triangleq (2\boldsymbol{L}_{\mathcal{G}}/\lambda_{\max} - \boldsymbol{I}_N)$. Accordingly, graph convolution can be approximated as

$$g_{\boldsymbol{\theta}'} * \boldsymbol{x} \approx \sum_{k=0}^{K} \theta'_k \left[\boldsymbol{U}\,\boldsymbol{T}_k(\widetilde{\boldsymbol{\Lambda}})\boldsymbol{U}^T\right]\boldsymbol{x} = \sum_{k=0}^{K} \theta'_k \boldsymbol{T}_k(\widetilde{\boldsymbol{L}})\,\boldsymbol{x} \tag{18}$$

where $\boldsymbol{T}_k(\widetilde{\boldsymbol{L}}) \in \mathbb{R}^{N\times N}$ is the Kth-order Chebyshev polynomial. The filtering operation is reduced to $\mathcal{O}(K|\mathcal{E}|)$ operations.

B. Linear Formulation of GCN

With first-order approximation (one-hop localization, i.e., $K = 1$) of (18) and further assuming[9] $\lambda_{\max} \approx 2$ and $\theta = \theta'_0 = -\theta'_1$, a layer-wise *linear* convolution operation can be defined to create a graph-based CNN model, i.e.,

$$g_\theta * \boldsymbol{x} = \theta\left(\boldsymbol{I}_N + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right)\boldsymbol{x} \to \theta\left(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\right)\boldsymbol{x} \tag{19}$$

with $\widetilde{\boldsymbol{A}} \triangleq \boldsymbol{A} + \boldsymbol{I}_N$ and $\widetilde{d}_{ii} \triangleq \sum_j \widetilde{a}_{ij}$. The last expression means that the matrix operation has been replaced with the so-called *re-normalization trick* [55].

Appendix II
Detailed Structure of AGCRN Layer

This appendix explains how the typical structure of a GRU unit is integrated with the spatial processing of GCN layers (exploiting 1) node-specific patterns and 2) a learned adjacency matrix). Such integration defines the AGCRN layer. More specifically, the AGCRN layer is formally defined as

$$\begin{aligned}
\hat{\boldsymbol{A}} &= \text{softmax}(\text{ReLU}(\boldsymbol{E}\boldsymbol{E}^T))\\
\boldsymbol{Z}[k] &= \sigma\left(\hat{\boldsymbol{A}}[\boldsymbol{X}[k]; \boldsymbol{H}[k-1]] \otimes (\boldsymbol{E}\otimes\boldsymbol{W}_z) + \boldsymbol{E}\boldsymbol{B}_z\right)\\
\boldsymbol{R}[k] &= \sigma\left(\hat{\boldsymbol{A}}[\boldsymbol{X}[k]; \boldsymbol{H}[k-1]] \otimes (\boldsymbol{E}\otimes\boldsymbol{W}_r) + \boldsymbol{E}\boldsymbol{B}_r\right)\\
\hat{\boldsymbol{H}}[k] &= \tanh\Big(\hat{\boldsymbol{A}}[\boldsymbol{X}[k]; \boldsymbol{R}[k]\odot\boldsymbol{H}[k-1]]\\
&\qquad\otimes\left(\boldsymbol{E}\otimes\boldsymbol{W}_{\hat{h}}\right) + \boldsymbol{E}\boldsymbol{B}_{\hat{h}}\Big)\\
\boldsymbol{H}[k] &= \boldsymbol{Z}[k]\odot\boldsymbol{H}[k-1] + (\boldsymbol{1}_{N\times F} - \boldsymbol{Z}[k])\odot\hat{\boldsymbol{H}}[k]
\end{aligned} \tag{20}$$

where $\boldsymbol{E} \in \mathbb{R}^{N\times l}$, $\boldsymbol{W}_z \in \mathbb{R}^{l\times(C+F)\times F}$, $\boldsymbol{W}_r \in \mathbb{R}^{l\times(C+F)\times F}$, $\boldsymbol{W}_{\hat{h}} \in \mathbb{R}^{l\times(C+F)\times F}$, $\boldsymbol{B}_z \in \mathbb{R}^{l\times F}$, $\boldsymbol{B}_r \in \mathbb{R}^{l\times F}$ and $\boldsymbol{B}_{\hat{h}} \in \mathbb{R}^{l\times F}$ are the trainable parameters of the AGCRN.

In the GRU-inspired layer of (20), the matrices $\boldsymbol{Z}[\cdot] \in \mathbb{R}^{N\times F}$, $\boldsymbol{R}[\cdot] \in \mathbb{R}^{N\times F}$, and $\hat{\boldsymbol{H}}[\cdot] \in \mathbb{R}^{N\times F}$ represent the

[9]These assumptions are made to constrain the number of trainable parameters (viz. reduce number of operations) and to address overfitting. Change in scale can be adapted by NN in the training phase.

update gate, the *reset gate*, and *candidate activation matrix*, respectively. It is worth noticing that in the above equations, the tensor products $\boldsymbol{E} \otimes \boldsymbol{W}_z$, $\boldsymbol{E} \otimes \boldsymbol{W}_r$, and $\boldsymbol{E} \otimes \boldsymbol{W}_{\hat{h}}$ have analogous meaning as $\boldsymbol{E}_g \otimes \boldsymbol{W}_g$ in (9). The same reasoning applies for the products $\hat{\boldsymbol{A}}[\boldsymbol{X}[k]; \boldsymbol{H}[k-1]] \otimes (\boldsymbol{E} \otimes \boldsymbol{W}_z)$, $\hat{\boldsymbol{A}}[\boldsymbol{X}[k]; \boldsymbol{H}[k-1]] \otimes (\boldsymbol{E} \otimes \boldsymbol{W}_r)$ and $\hat{\boldsymbol{A}}[\boldsymbol{X}[k]; \boldsymbol{R}[k] \odot \boldsymbol{H}[k-1]] \otimes (\boldsymbol{E} \otimes \boldsymbol{W}_{\hat{h}})$ when compared with $(\widetilde{\boldsymbol{D}}^{-(1/2)} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-(1/2)} \boldsymbol{X}) \otimes (\boldsymbol{E}_g \otimes \boldsymbol{W}_g)$ in the same (9). The functions $\sigma(\cdot)$ and $\tanh(\cdot)$ are the (entry-wise) sigmoid and hyperbolic tangent activations, respectively. Furthermore, the matrix $\hat{\boldsymbol{A}} \in \mathbb{R}^{N \times N}$ is the (estimated) pseudo-Laplacian of graph $\mathcal{G}$. In order to reduce the number of learning parameters, AGCRN unifies the embedding matrices associated with node-specific patterns and (adaptive) graph structure (i.e., $\boldsymbol{E}_g$ and $\boldsymbol{E}_a$) into a *single embedding matrix* $\boldsymbol{E}$.

Finally, the AGCRN output at time k is represented by the matrix $\boldsymbol{H}[k] \in \mathbb{R}^{N \times F}$, which is obtained as the *weighted sum* of its previous version ($\boldsymbol{H}[k-1]$) and the candidate activation matrix ($\hat{\boldsymbol{H}}[k]$). The weights are provided by the update gate at the same time step ($\boldsymbol{Z}[k]$), whereas the candidate activation matrix is obtained with an analogous spatial processing and leveraging the reset gate outcome at the same time ($\boldsymbol{R}[k]$).

Acknowledgment

The authors would like to thank Pascal Frossard, École Polytechnique Fédérale de Lausanne, Switzerland, and Antonio Ortega, University of Southern California, Los Angeles, CA, USA, for useful discussions during the development of the work.

References

[1] A. Rasheed, O. San, and T. Kvamsdal, "Digital twin: Values, challenges and enablers from a modeling perspective," *IEEE Access*, vol. 8, pp. 21980–22012, 2020.

[2] Z. Yang, N. Meratnia, and P. Havinga, "An online outlier detection technique for wireless sensor networks using unsupervised quarter-sphere support vector machine," in *Proc. Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process.*, Dec. 2008, pp. 151–156.

[3] 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 Things J.*, vol. 8, no. 2, pp. 657–671, Jan. 2021.

[4] Z. Zhang, A. Mehmood, L. Shu, Z. Huo, Y. Zhang, and M. Mukherjee, "A survey on fault diagnosis in wireless sensor networks," *IEEE Access*, vol. 6, pp. 11349–11364, 2018.

[5] A. Mahapatro and P. M. Khilar, "Fault diagnosis in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2000–2026, 4th Quart., 2013.

[6] R. Isermann, *Fault-Diagnosis Systems: An Introduction From Fault Detection to Fault Tolerance*. Cham, Switzerland: Springer, 2005.

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

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

[9] 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 J.*, vol. 21, no. 4, pp. 4827–4838, Feb. 2021.

[10] 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 Trans. Ind. Appl.*, vol. 53, no. 3, pp. 2852–2861, May 2017.

[11] H. Zhao, "Neural component analysis for fault detection," *Chemometric Intell. Lab. Syst.*, vol. 176, pp. 11–21, May 2018.

[12] H. Darvishi, D. Ciuonzo, and P. Salvo Rossi, "A machine-learning architecture for sensor fault detection, isolation, and accommodation in digital twins," *IEEE Sensors J.*, vol. 23, no. 3, pp. 2522–2538, Feb. 2023.

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

[14] 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 Things J.*, vol. 5, no. 4, pp. 2431–2440, Aug. 2018.

[15] G. Su, D. Fang, S. Jian, and L. Fengmei, "Sensor fault detection with online sparse least squares support vector machine," in *Proc. 32nd Chin. Control Conf.*, Jul. 2013, pp. 6220–6224.

[16] T. Yu, X. Wang, and A. Shami, "Recursive principal component analysis-based data outlier detection and sensor data aggregation in IoT systems," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2207–2216, Dec. 2017.

[17] T.-B. Dang, D.-T. Le, T.-D. Nguyen, M. Kim, and H. Choo, "Monotone split and conquer for anomaly detection in IoT sensory data," *IEEE Internet Things J.*, vol. 8, no. 20, pp. 15468–15485, Oct. 2021.

[18] G. Campa, M. Thiagarajan, M. Krishnamurty, M. R. Napolitano, and M. Gautam, "A neural network based sensor validation scheme for heavy-duty diesel engines," *J. Dyn. Syst., Meas., Control*, vol. 130, no. 2, pp. 1–12, Mar. 2008.

[19] M. Alrifaey, W. H. Lim, and C. K. Ang, "A novel deep learning framework based RNN-SAE for fault detection of electrical gas generator," *IEEE Access*, vol. 9, pp. 21433–21442, 2021.

[20] H. Darvishi, D. Ciuonzo, and P. Salvo Rossi, "Exploring a modular architecture for sensor validation in digital twins," in *Proc. IEEE Sensors*, Oct. 2022, pp. 1–4.

[21] Z. Zhi, L. Liu, D. Liu, and C. Hu, "Fault detection of the harmonic reducer based on CNN-LSTM with a novel denoising algorithm," *IEEE Sensors J.*, vol. 22, no. 3, pp. 2572–2581, Feb. 2022.

[22] F. van Wyk, Y. Wang, A. Khojandi, and N. Masoud, "Real-time sensor anomaly detection and identification in automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1264–1276, Mar. 2020.

[23] 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 Things J.*, vol. 7, no. 9, pp. 8462–8471, Sep. 2020.

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

[25] M. Elnour, N. Meskin, and M. Al-Naemi, "Sensor data validation and fault diagnosis using auto-associative neural network for HVAC systems," *J. Building Eng.*, vol. 27, Jan. 2020, Art. no. 100935.

[26] 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," *Sustain. Cities Soc.*, vol. 52, Jan. 2020, Art. no. 101847.

[27] B. Pourbabaee, N. Meskin, and K. Khorasani, "Sensor fault detection, isolation, and identification using multiple-model-based hybrid Kalman filter for gas turbine engines," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 4, pp. 1184–1200, Jul. 2016.

[28] I. Samy, I. Postlethwaite, and D.-W. Gu, "Detection and accommodation of sensor faults in UAVs—A comparison of NN and EKF based approaches," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Dec. 2010, pp. 4365–4372.

[29] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.

[30] D. Hong, L. Gao, J. Yao, B. Zhang, A. Plaza, and J. Chanussot, "Graph convolutional networks for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 7, pp. 5966–5978, Jul. 2021.

[31] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with multi-stream adaptive graph convolutional networks," *IEEE Trans. Image Process.*, vol. 29, pp. 9532–9545, 2020.

[32] W. Liao, D. Yang, Y. Wang, and X. Ren, "Fault diagnosis of power transformers using graph convolutional network," *CSEE J. Power Energy Syst.*, vol. 7, no. 2, pp. 241–249, Mar. 2021.

[33] Z. Yan, J. Ge, Y. Wu, L. Li, and T. Li, "Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1040–1057, Jun. 2020.

[34] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9214–9231, Jun. 2022.

[35] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 5, pp. 4027–4035.

[36] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, "Adaptive graph convolutional recurrent network for traffic forecasting," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2020, pp. 1–12.

[37] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial–temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI*, vol. 33, 2019, pp. 922–929.

[38] H. Khorasgani, A. Hasanzadeh, A. Farahat, and C. Gupta, "Fault detection and isolation in industrial networks using graph convolutional neural networks," in *Proc. IEEE Int. Conf. Prognostics Health Manage. (ICPHM)*, Jun. 2019, pp. 1–7.

[39] S. Hussain, M. Mokhtar, and J. M. Howe, "Sensor failure detection, identification, and accommodation using fully connected cascade neural network," *IEEE Trans. Ind. Electron.*, vol. 62, no. 3, pp. 1683–1692, Mar. 2015.

[40] A. Ortega, *Introduction to Graph Signal Processing*. Cambridge, U.K.: Cambridge Univ. Press, 2022.

[41] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[42] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 3844–3852.

[43] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*.

[44] L. Bai, L. Yao, S. S. Kanhere, X. Wang, and Q. Z. Sheng, "STG2Seq: Spatial–temporal graph to sequence model for multi-step passenger demand forecasting," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 1981–1987.

[45] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–16.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–13.

[47] T. Dozat, "Incorporating Nesterov momentum into Adam," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–6.

[48] 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 Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 3, pp. 1093–1106, Mar. 2022.

[49] A. Chawla et al., "IoT-based monitoring in carbon capture and storage systems," *IEEE Internet Things Mag.*, vol. 5, no. 4, pp. 106–111, Dec. 2022.

[50] S. Guo, Y. Lin, H. Wan, X. Li, and G. Cong, "Learning dynamics and heterogeneity of spatial–temporal graph data for traffic forecasting," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 11, pp. 5415–5428, Nov. 2022.

[51] K. Guo, Y. Hu, Z. Qian, Y. Sun, J. Gao, and B. Yin, "Dynamic graph convolution network for traffic forecasting based on latent network of Laplace matrix estimation," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 1009–1018, Feb. 2022.

[52] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 1–8.

[53] W. J. Youden, "Index for rating diagnostic tests," *Cancer*, vol. 3, no. 1, pp. 32–35, 1950.

[54] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Appl. Comput. Harmon. Anal.*, vol. 30, no. 2, pp. 129–150, Mar. 2011.

[55] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.

Hossein Darvishi (Graduate Student Member, IEEE) received the B.Sc. degree from the Kermanshah University of Technology, Kermanshah, Iran, in 2016, and the M.Sc. (Hons.) degree in telecommunications engineering from the K. N. Toosi University of Technology, Tehran, Iran, 2018, and the Ph.D. degree in electronics and telecommunications from the Department of Electronic Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2023.

He has also worked as a Visiting Researcher at the Signal Processing Laboratory (LTS4), Electrical Engineering Institute of the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland. He is now a Senior Data Scientist at Cognizant AI&A Nordics, Oslo, Norway. He is with the Nordic Industrial IoT Hub (HI2OT) and the SIGNIFY Project, which is a research and connectivity program in sensor validation solutions for digital twins of safety-critical systems, jointly conducted by NTNU and SINTEF. His research interests include statistical signal processing, machine learning, Internet of Things, and wireless sensor networks.

Dr. Darvishi is a reviewer for reputable journals including IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and IEEE SENSORS JOURNAL.

Domenico Ciuonzo (Senior Member, IEEE) received the Ph.D. degree from the University of Campania "L. Vanvitelli", Naples, Italy, in 2013.

He is a Tenured-Track Assistant Professor at the University of Napoli Federico II, Naples. Since 2011, he has been holding several visiting researcher appointments (NATO CMRE, UConn, NTNU, and CTTC). His research interests include data fusion, statistical signal processing, wireless sensor networks, Internet of Things (IoT), and machine learning.

Dr. Ciuonzo was a recipient of the two Best Paper Awards from IEEE ICCCS in 2019 and Elsevier Computer Networks in 2020, the 2019 Exceptional Service Award from IEEE Aerospace and Electronic Systems Society (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.

Pierluigi Salvo Rossi (Senior Member, IEEE) was born in Naples, Italy, in 1977. He received the Dr.Eng. (summa cum laude) degree in telecommunications engineering and the Ph.D. degree in computer engineering from the University of Naples "Federico II", Naples, 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, Trondheim. Previously, he worked with the University of Naples "Federico II"; the Second University of Naples, Naples; NTNU; and Kongsberg Digital AS, Horten, Norway. He held a visiting appointments with Drexel University, Philadelphia, PA, USA; Lund University, Lund, Sweden; NTNU; and Uppsala University, Uppsala, 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 IEEE COMMUNICATIONS LETTERS in 2018. He is (or has been) in the Editorial Board of IEEE SENSORS JOURNAL, IEEE OPEN JOURNAL OF THE COMMUNICATIONS SOCIETY, IEEE TRANSACTIONS ON SIGNAL AND INFORMATION PROCESSING OVER NETWORKS, IEEE COMMUNICATIONS LETTERS, and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS.